

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—

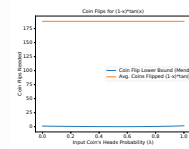
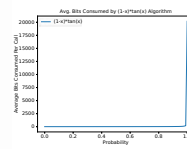
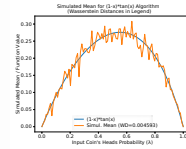
- an ϵ of $1 - (x + c) * 1.001$ was used (or 0.0001 if ϵ would be greater than 1), and
- an ϵ of $(x - c) * 0.9995$ for the subtraction variants.

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.

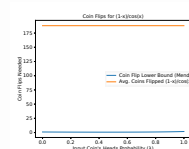
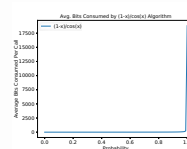
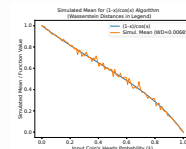
0.1 The Charts

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

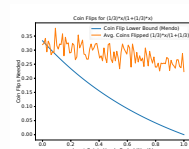
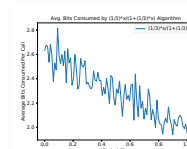
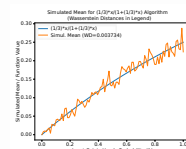
$$(1-x)*\tan(x)$$



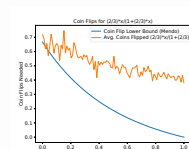
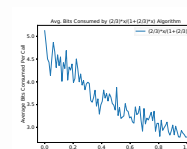
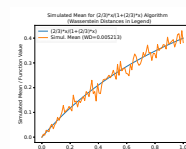
$$(1-x)/\cos(x)$$



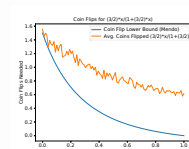
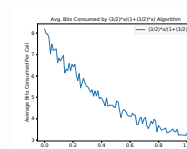
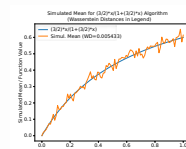
$$(1/3)*x/(1+(1/3)*x)$$



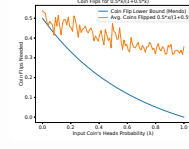
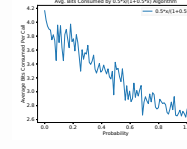
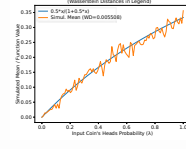
$$(2/3)*x/(1+(2/3)*x)$$



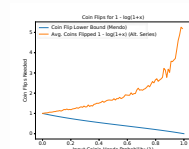
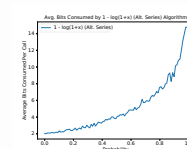
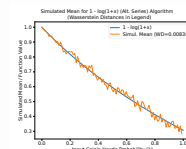
$$(3/2)*x/(1+(3/2)*x)$$



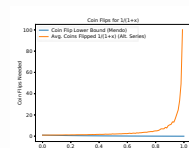
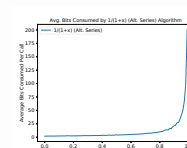
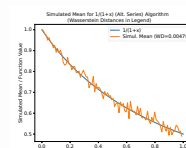
$$0.5*x/(1+0.5*x)$$



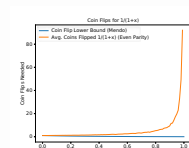
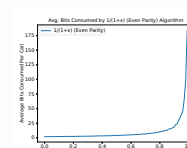
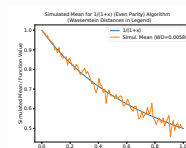
$$1 - \ln(1+x) \text{ (Alt. Series)}$$



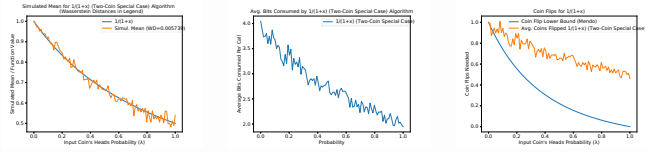
$$1/(1+x) \text{ (Alt. Series)}$$



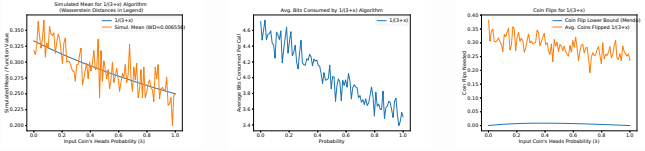
$$1/(1+x) \text{ (Even Parity)}$$



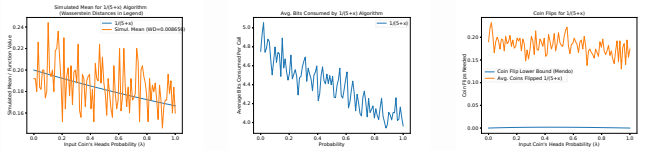
1/(1+x) (Two-Coin Special Case)



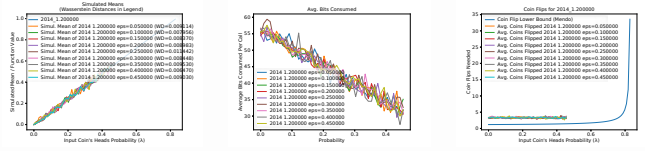
1/(3+x)



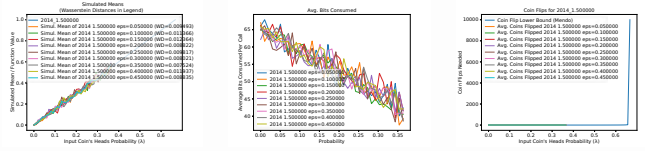
1/(5+x)



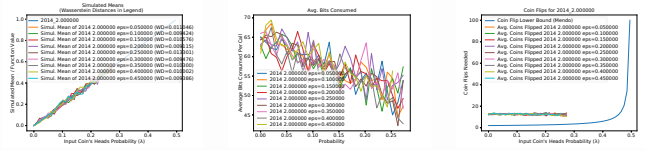
2014 1.200000
eps=0.050000



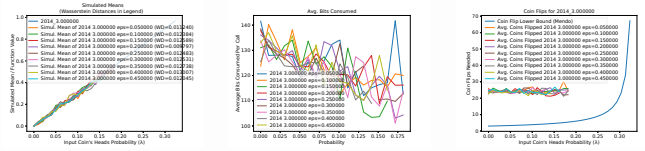
2014 1.500000
eps=0.050000



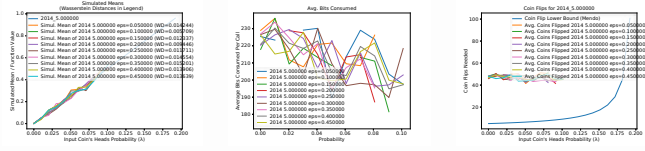
2014 2.000000
eps=0.050000



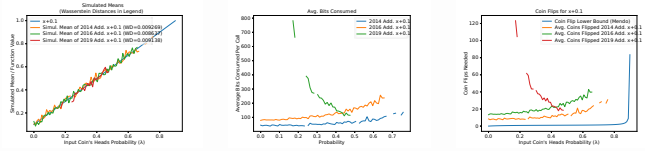
2014 3.000000
eps=0.050000



2014 5.000000
eps=0.050000



2014 Add. x+0.1



[illegible]

Figure 1 consists of three subplots comparing the proposed method (CUP Filter) with state-of-the-art methods (CUP Filter Lower Bound, CUP Filter, and CUP Filter Upper Bound) across different years (2014, 2015, 2016, 2017, 2018, 2019) and AUC values (0.5 to 0.9).

Top Left Plot: Stratified Mean AUC (std) vs AUC

This plot shows the stratified mean AUC (std) for different methods and years. The x-axis represents the AUC value (0.5 to 0.9), and the y-axis represents the stratified mean AUC (std) (0.5 to 0.9). The legend indicates the following methods and years:

- 2014, AUC=0.5
- 2015, AUC=0.5
- 2016, AUC=0.5
- 2017, AUC=0.5
- 2018, AUC=0.5
- 2019, AUC=0.5

Top Right Plot: Average ROC Curves (std) vs AUC

This plot shows the average ROC curves (std) for different methods and years. The x-axis represents the AUC value (0.5 to 0.9), and the y-axis represents the average ROC curves (std) (0.5 to 0.9). The legend indicates the following methods and years:

- 2014, AUC=0.5
- 2015, AUC=0.5
- 2016, AUC=0.5
- 2017, AUC=0.5
- 2018, AUC=0.5
- 2019, AUC=0.5

Bottom Plot: CUP Filter for AUC=0.5 vs CUP Filter for AUC=0.5

This plot shows the CUP Filter for AUC=0.5 for different methods and years. The x-axis represents the CUP Filter for AUC=0.5 (0.5 to 0.9), and the y-axis represents the CUP Filter for AUC=0.5 (0.5 to 0.9). The legend indicates the following methods and years:

- CUP Filter Lower Bound
- CUP Filter
- CUP Filter Upper Bound
- 2014, AUC=0.5
- 2015, AUC=0.5
- 2016, AUC=0.5
- 2017, AUC=0.5
- 2018, AUC=0.5
- 2019, AUC=0.5

[illegible][illegible]

Figure 1 consists of three subplots. Subplot (a) is a scatter plot titled 'Scatter Plot (Covariate)' showing 'Predicted Mean (Covariate)' on the y-axis versus 'Input Covariate Mean Probability (C)' on the x-axis. Both axes range from 0.0 to 0.5. Data points are colored by year: 2014 (blue), 2015 (orange), 2016 (green), 2017 (red), 2018 (purple), 2019 (brown), 2020 (pink), and 2021 (grey). A diagonal line represents the ideal prediction. Subplot (b) is a line plot titled 'Avg. Likelihood Ratio (ALR)' showing 'Average Likelihood Ratio' on the y-axis (0.0 to 1.0) versus 'Probability' on the x-axis (0.0 to 0.5). It includes a legend for '2014 Lin. $\chi^2=0$ ', '2015 Lin. $\chi^2=0$ ', '2016 Lin. $\chi^2=0$ ', '2017 Lin. $\chi^2=0$ ', '2018 Lin. $\chi^2=0$ ', '2019 Lin. $\chi^2=0$ ', '2020 Lin. $\chi^2=0$ ', and '2021 Mean $\chi^2=0.0099$ '. Subplot (c) is a line plot titled 'Cost Ratio for $\chi^2=0$ ' showing 'Cost Ratio' on the y-axis (0 to 100) versus 'Input Covariate Mean Probability (C)' on the x-axis (0.0 to 0.5). It includes a legend for '2014 Lin. $\chi^2=0$ ', '2015 Lin. $\chi^2=0$ ', '2016 Lin. $\chi^2=0$ ', '2017 Lin. $\chi^2=0$ ', '2018 Lin. $\chi^2=0$ ', '2019 Lin. $\chi^2=0$ ', '2020 Lin. $\chi^2=0$ ', and '2021 Mean $\chi^2=0.0099$ $\chi^2=0$ '.

Figure 1 consists of two subplots, (a) and (b), illustrating the performance of the proposed model.

Subplot (a) is a Receiver Operating Characteristic (ROC) curve. The x-axis is labeled "Input Coir's Weak Probability (λ)" and ranges from 0.00 to 0.15. The y-axis is labeled "Detection Rate (True Positive Rate)" and ranges from 0.00 to 1.00. The plot shows several curves for different models:

- Proposed (blue line): Shows the highest performance, reaching a True Positive Rate of 1.00 at a False Positive Rate of approximately 0.05.
- Proposed + λ (red line): Similar performance to the proposed model.
- Proposed + λ + λ^2 (green line): Slightly lower performance than the proposed model.
- Proposed + λ + λ^2 + λ^3 (orange line): Lower performance.
- Proposed + λ + λ^2 + λ^3 + λ^4 (purple line): Lowest performance among the models shown.
- Proposed + λ + λ^2 + λ^3 + λ^4 + λ^5 (brown line): Lowest performance.

Subplot (b) is a line graph showing the Average ROC Curve for the proposed model across different probabilities. The x-axis is labeled "Probability" and ranges from 0.00 to 0.15. The y-axis is labeled "Average ROC Curve" and ranges from 0.00 to 1.00. The plot shows several curves for different models:

- Proposed (blue line): Shows the highest performance, reaching an Average ROC Curve of 1.00 at a Probability of approximately 0.05.
- Proposed + λ (red line): Similar performance to the proposed model.
- Proposed + λ + λ^2 (green line): Slightly lower performance than the proposed model.
- Proposed + λ + λ^2 + λ^3 (orange line): Lower performance.
- Proposed + λ + λ^2 + λ^3 + λ^4 (purple line): Lowest performance among the models shown.
- Proposed + λ + λ^2 + λ^3 + λ^4 + λ^5 (brown line): Lowest performance.

[illegible]

Figure 1 consists of three subplots labeled (a), (b), and (c), each showing performance metrics versus Input Correlation (ρ) ranging from 0.0 to 1.0.

Subplot (a) shows the Standard Error of the Mean (SEM) on the y-axis (0.00 to 0.02) against Input Correlation (ρ) on the x-axis (0.0 to 1.0). The legend includes:

- Prop. (red line with circles)
- 256B Lin. $\rho=0$ (0.002-0.00346) (blue line)
- Control (green line)
- Prop. of case linear $\rho=0$ (0.002-0.01347) (orange line)
- 256B Lin. $\rho=0.5$ (0.002-0.00346) (purple line)
- Control (green line)
- Prop. of case linear $\rho=0.5$ (0.002-0.01347) (orange line)

 All lines show a decreasing trend in SEM as ρ increases.

Subplot (b) shows Average Bit Count on the y-axis (0 to 250) against Input Correlation (ρ) on the x-axis (0.0 to 1.0). The legend includes:

- Prop. (red line with circles)
- 256B Lin. $\rho=0$ (blue line)
- Prop. of case linear $\rho=0$ (orange line)
- 256B Lin. $\rho=0.5$ (purple line)
- Prop. of case linear $\rho=0.5$ (orange line)

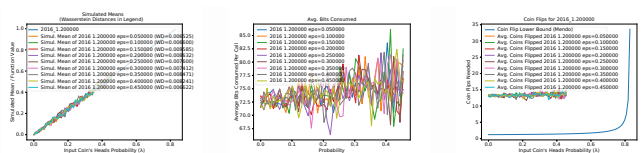
 The bit count generally increases with ρ , with the 256B Lin. $\rho=0.5$ series showing the highest values.

Subplot (c) shows Cost on the y-axis (0 to 200) against Input Correlation (ρ) on the x-axis (0.0 to 1.0). The legend includes:

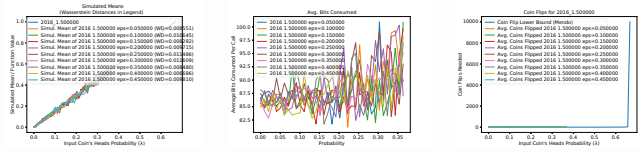
- Prop. (red line with circles)
- 256B Lin. $\rho=0$ (blue line)
- Prop. of case linear $\rho=0$ (orange line)
- 256B Lin. $\rho=0.5$ (purple line)
- Prop. of case linear $\rho=0.5$ (orange line)

 The cost generally decreases as ρ increases, with the 256B Lin. $\rho=0.5$ series showing the highest values.

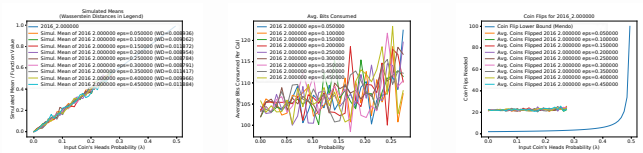
```
2016 1.200000
eps=0.050000
```



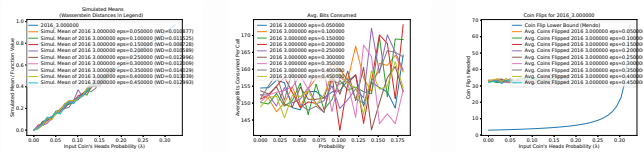
```
2016 1.500000
eps=0.050000
```



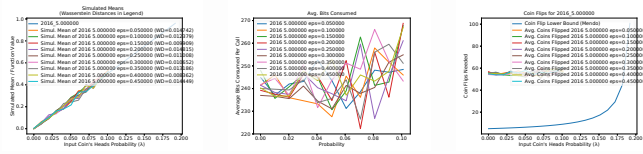
```
2016 2.000000
eps=0.050000
```



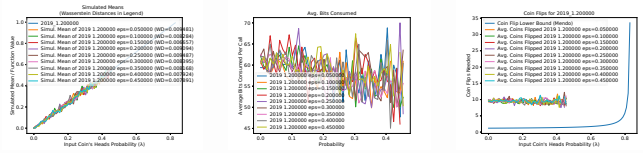
```
2016 3.000000
eps=0.050000
```



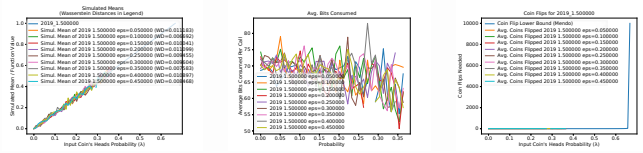
```
2016 5.000000
eps=0.050000
```



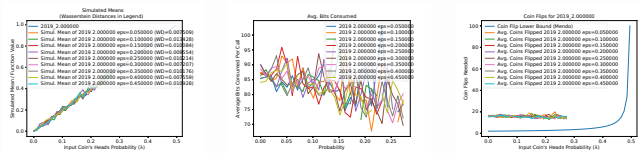
```
2019 1.200000
eps=0.050000
```



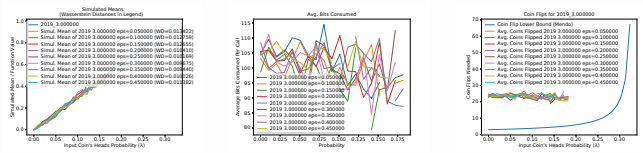
```
2019 1.500000
eps=0.050000
```



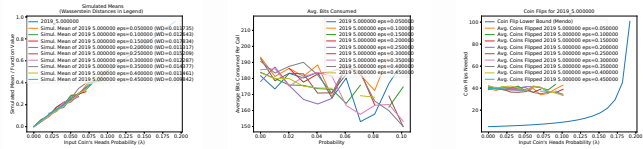
```
2019 2.000000
eps=0.050000
```



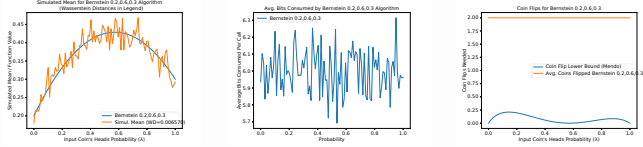
```
2019 3.000000
eps=0.050000
```



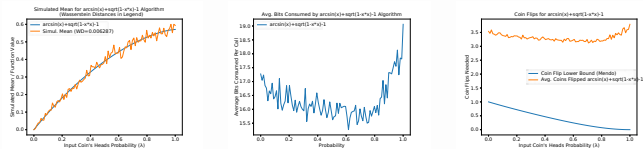
2019 5.000000
eps=0.050000



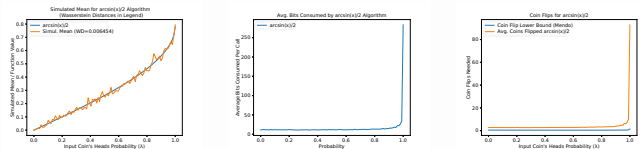
Bernstein
0.2,0.6,0.3



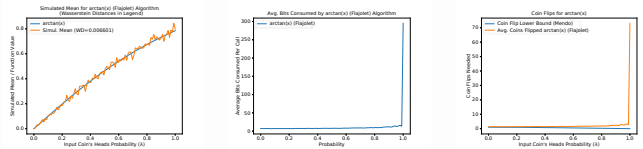
$\arcsin(x) + \sqrt{1-x^2} - 1$



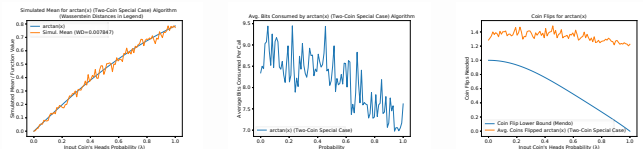
$\arcsin(x)/2$



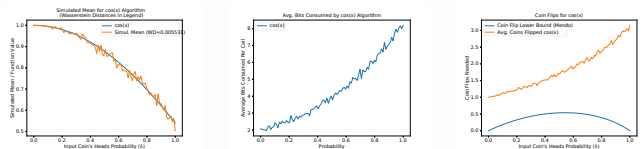
$\arctan(x)$
(Flajolet)



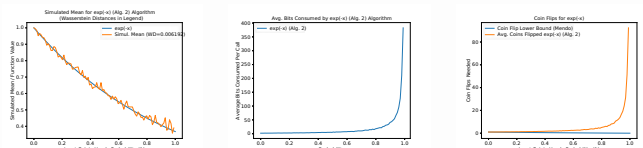
$\arctan(x)$ (Two-Coin Special Case)



$\cos(x)$



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



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

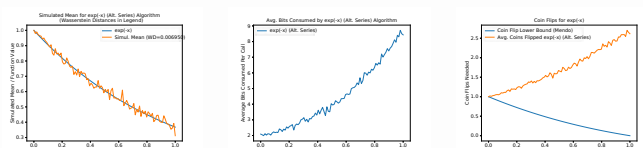


Figure 1 consists of three subplots labeled (a), (b), and (c), each showing the performance of the proposed algorithm as a function of the logarithmic scale of the sample size (ranging from 0.0 to 1.0).

- (a) Standard Error (SE) vs. Logarithmic Scale of Sample Size:** The y-axis is 'Standard Error (SE) (Std)' ranging from 0.0 to 1.0. The x-axis is 'Logarithmic Scale of Sample Size' ranging from 0.0 to 1.0. The plot shows a decreasing trend. The 'Proposed Method (20000-100000)' is represented by a blue line with circular markers, and the 'Exact SE (20000-100000)' is represented by a solid black line. The proposed method closely follows the exact SE.
- (b) Average Coverage Rate vs. Logarithmic Scale of Sample Size:** The y-axis is 'Average Coverage Rate' ranging from 0.000 to 0.040. The x-axis is 'Logarithmic Scale of Sample Size' ranging from 0.0 to 1.0. The plot shows that the 'Proposed Method (20000-100000)' (blue line with circular markers) maintains a coverage rate near 0.000, while the 'Exact Coverage Rate' (solid black line) increases sharply to approximately 0.035 as the sample size increases.
- (c) CPU Time vs. Logarithmic Scale of Sample Size:** The y-axis is 'CPU Time (Sec)' ranging from 0 to 250. The x-axis is 'Logarithmic Scale of Sample Size' ranging from 0.0 to 1.0. The plot shows that the 'Proposed Method (20000-100000)' (blue line with circular markers) has a CPU time that increases sharply to approximately 250 seconds as the sample size increases, while the 'Exact Coverage Rate' (solid black line) remains near 0 seconds.

Figure 10 consists of three subplots comparing the proposed algorithm (blue line) with the state-of-the-art algorithm (orange line).

- Top Plot:** Titled "Empirical Mean for $\text{exp}(1/2) \times 10$ Algorithm". The y-axis is "Standard Error of $\text{exp}(1/2) \times 10$ " (0.0 to 1.0) and the x-axis is "Signal-to-Noise Ratio (dB)" (-10 to 10). The blue line is consistently lower than the orange line, indicating a smaller standard error.
- Bottom Left Plot:** Titled "Avg. Bits Consumed for $\text{exp}(1/2) \times 10$ Algorithm". The y-axis is "Number of Bits Consumed (bits)" (0 to 10) and the x-axis is "Probability" (0.0 to 1.0). The blue line is consistently lower than the orange line, indicating fewer bits consumed.
- Bottom Right Plot:** Titled "Gap Error for $\text{exp}(1/2) \times 10$ ". The y-axis is "Gap Error (bits)" (0.0 to 2.0) and the x-axis is "Signal-to-Noise Ratio (dB)" (-10 to 10). The blue line is consistently lower than the orange line, indicating a smaller gap error.

Figure 1 consists of three subplots comparing the proposed algorithm with the Pigeon-inspired algorithm. The left subplot shows the 'Serialized Mean for $\log_2(x+1)$ (Pigeon)' and 'Serialized Mean for $\log_2(x+1)$ (Proposed)' algorithms. The middle subplot shows the 'log2(x+1) (Pigeon)' and 'log2(x+1) (Proposed)' algorithms. The right subplot shows the 'Cost Ratio for $\log_2(x+1)$ ' for 'Cost Ratio Lower Bound' and 'Avg. Cost Pigeon log2(x+1) (Pigeon)'. The proposed algorithm consistently shows lower values than the Pigeon-inspired algorithm across all three metrics.

Figure 1 consists of three subplots labeled (a), (b), and (c), each showing performance metrics for the proposed algorithm across different numbers of nodes (0.0 to 1.0).

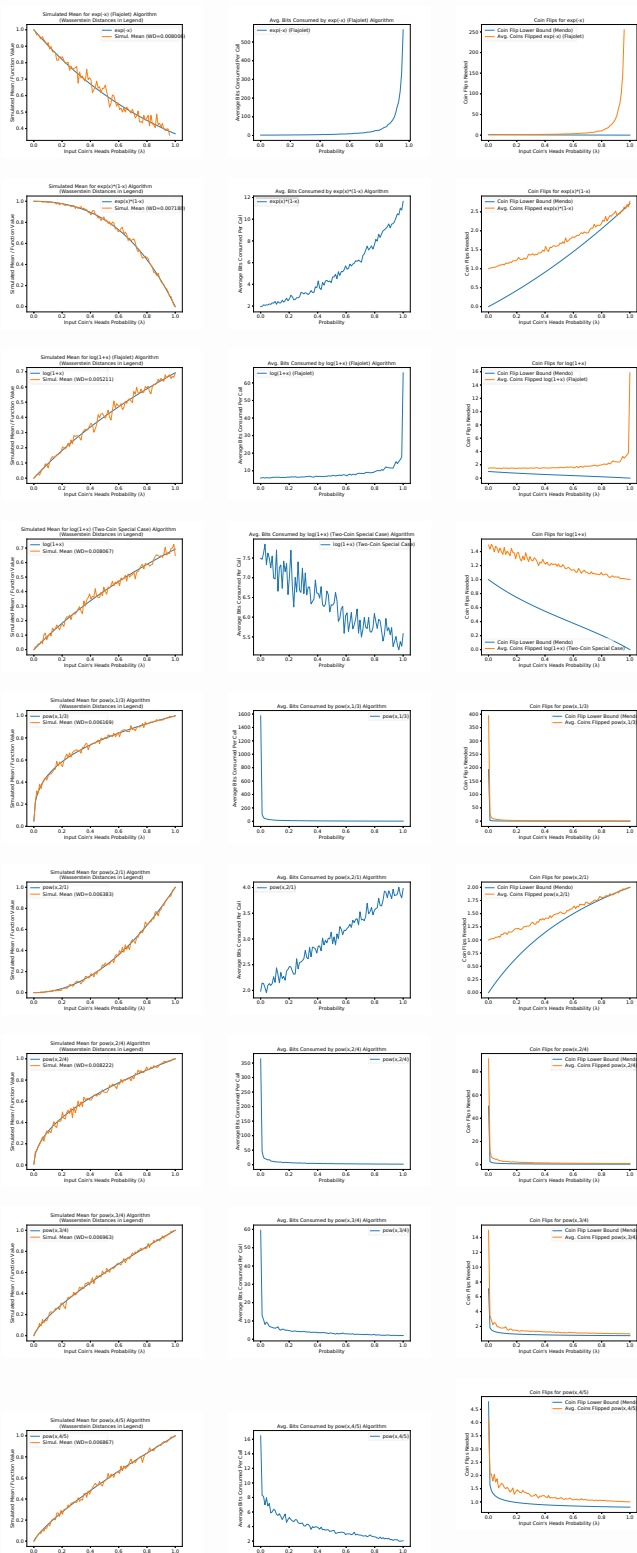
- (a) Scalability:** The y-axis is 'Disabled flow capacity (kA)' ranging from 0.0 to 1.0. The x-axis is 'Number of nodes' ranging from 0.0 to 1.0. The plot shows two curves: 'points 1/20' (blue line with circles) and 'points 1/200' (orange line with circles). Both curves start at (0,0) and increase monotonically, with 'points 1/200' reaching a higher capacity than 'points 1/20'.
- (b) Convergence:** The y-axis is 'Average flow capacity (kA)' ranging from 0.0 to 1000. The x-axis is 'Iteration number' ranging from 0.0 to 1.0. The plot shows two curves: 'points 1/20' (blue line with circles) and 'points 1/200' (orange line with circles). Both curves start at approximately 1000 and drop sharply to near zero by iteration 0.2, remaining stable thereafter.
- (c) Error:** The y-axis is 'Error (kA)' ranging from 0 to 400. The x-axis is 'Iteration number' ranging from 0.0 to 1.0. The plot shows two curves: 'points 1/20' (blue line with circles) and 'points 1/200' (orange line with circles). Both curves start at approximately 400 and drop sharply to near zero by iteration 0.2, remaining stable thereafter.

Figure 1 consists of three subplots. Subplot (a) is titled 'Simulated Error for p=0.1: 20 Algorithms' and shows 'Standard Error of the Mean (SEM)' on the y-axis (0.0 to 1.8) versus 'Signal-to-Noise Ratio (SNR)' on the x-axis (0.0 to 1.0). It compares 'p=0.1: 201' (blue line) and 'p=0.1: Mean (0.00000000)' (orange line). Subplot (b) is titled 'Avg. RMSE Computed by p=0.1: 20 Algorithms' and shows 'Average RMSE Computed (RMSE)' on the y-axis (0.0 to 6.0) versus 'SNR' on the x-axis (0.0 to 1.0). It compares 'p=0.1: 201' (blue line) and 'p=0.1: Mean (0.00000000)' (orange line). Subplot (c) is titled 'Case Files for p=0.1: 20 Algorithms' and shows 'Case Files' on the y-axis (0.000 to 2.000) versus 'Signal-to-Noise Ratio (SNR)' on the x-axis (0.0 to 1.0). It compares 'Case Files (User Requested)' (blue line) and 'Avg. Case Files (p=0.1: 201)' (orange line).

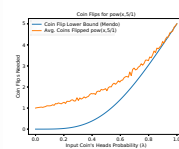
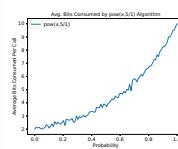
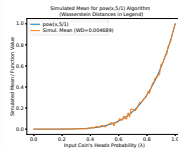
Figure 1 consists of four subplots showing the performance of the proposed 3-W algorithm across different input confidence mean probabilities (X-axis, 0.0 to 1.0).

- Top-left plot:** Stratified Mean for p=0.5, 3-W algorithm. The Y-axis is 'Stratified Mean for p=0.5, 3-W algorithm' (0.0 to 1.0). It compares 'Proposed 3-W algorithm' (blue line with circles) and 'Service Mean (MD=0.0386)' (orange line with circles). Both lines show a similar upward trend, starting near 0.0 and reaching 1.0 at X=1.0.
- Top-right plot:** Avg. Size Computed by p=0.5, 3-W algorithm. The Y-axis is 'Avg. Size Computed by p=0.5, 3-W algorithm' (0.0 to 1.0). The 'p=0.5, 3-W' (blue line) starts at 1.0 for X=0.0 and drops sharply to near 0.0 by X=0.1, remaining there until X=1.0.
- Bottom-left plot:** Mean R1s Computed by p=0.5, 3-W algorithm. The Y-axis is 'Mean R1s Computed by p=0.5, 3-W algorithm' (0.0 to 1.0). The 'p=0.5, 3-W' (blue line) starts at 1.0 for X=0.0 and drops sharply to near 0.0 by X=0.1, remaining there until X=1.0.
- Bottom-right plot:** Conf. Hits for p=0.5, 3-W algorithm. The Y-axis is 'Conf. Hits for p=0.5, 3-W algorithm' (0.0 to 1.0). It compares 'Proposed 3-W algorithm' (blue line with circles) and 'Conf. Hits for p=0.5, 3-W algorithm' (orange line with circles). Both lines show a sharp drop from 1.0 at X=0.0 to near 0.0 by X=0.1, remaining there until X=1.0.

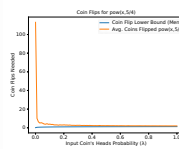
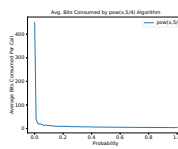
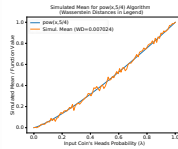
Figure 1 consists of three subplots. Subplot (a) is a line graph titled 'Simulated Mean for penta-K5 Algorithm' showing 'Simulated Mean for penta-K5 Algorithm' on the y-axis (ranging from 0.0 to 1.0) versus 'Input Data Probability (p)' on the x-axis (ranging from 0.0 to 1.0). It includes a legend for 'Simulated Mean for penta-K5 Algorithm' (blue line) and 'Simulated Mean for penta-K5 Algorithm' (orange line). Subplot (b) is a line graph titled 'Avg. Bits Consumed by penta-K5 Algorithm' showing 'Avg. Bits Consumed by penta-K5 Algorithm' on the y-axis (ranging from 0.0 to 1.0) versus 'Input Data Probability (p)' on the x-axis (ranging from 0.0 to 1.0). It includes a legend for 'penta-K5' (blue line). Subplot (c) is a line graph titled 'Avg. Bits Consumed by penta-K5 Algorithm' showing 'Avg. Bits Consumed by penta-K5 Algorithm' on the y-axis (ranging from 0.0 to 1.0) versus 'Input Data Probability (p)' on the x-axis (ranging from 0.0 to 1.0). It includes a legend for 'penta-K5' (blue line) and 'penta-K5' (orange line).



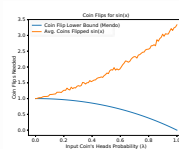
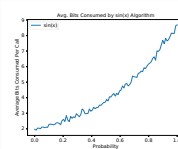
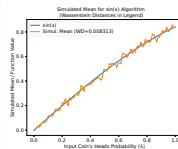
pow(x,5/1)



pow(x,5/4)



sin(x)



sqrt(x)

