

## # 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  $\lambda$  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  $\lambda$  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  $\lambda$ , the entire data point for that  $\lambda$  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)[<sup>1</sup>]. Note that some functions require a growing number of coin flips as  $\lambda$  approaches 0 or 1. Note that for the 2014, 2016, and 2019 algorithms—

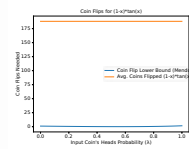
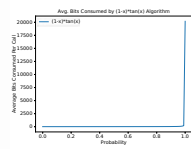
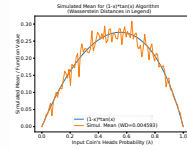
- an  $\epsilon$  of  $1 - (x + c) * 1.001$  was used (or 0.0001 if  $\epsilon$  would be greater than 1), and
- an  $\epsilon$  of  $(x - c) * 0.9995$  for the subtraction variants.

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

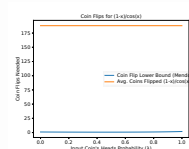
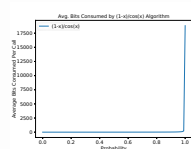
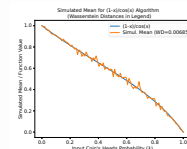
## 0.1 The Charts

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

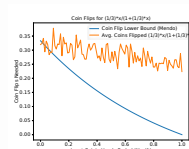
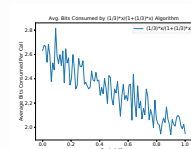
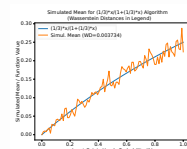
$$(1-x)*\tan(x)$$



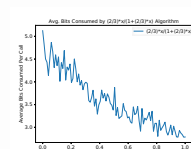
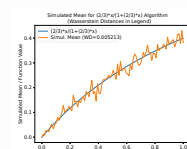
$$(1-x)/\cos(x)$$



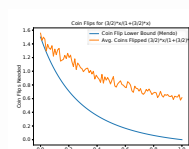
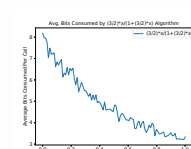
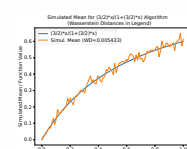
$$(1/3)*x/(1+(1/3)*x)$$



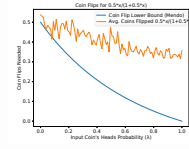
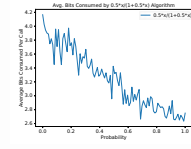
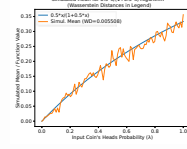
$$(2/3)*x/(1+(2/3)*x)$$



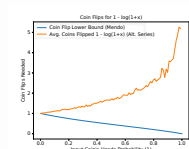
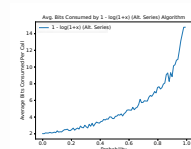
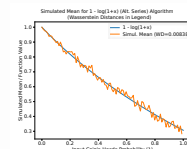
$$(3/2)*x/(1+(3/2)*x)$$



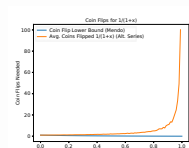
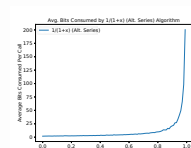
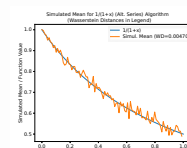
$$0.5*x/(1+0.5*x)$$



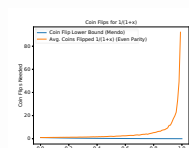
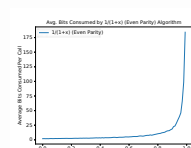
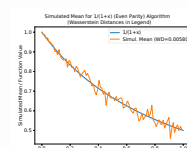
$$1 - \ln(1+x) \text{ (Alt. Series)}$$



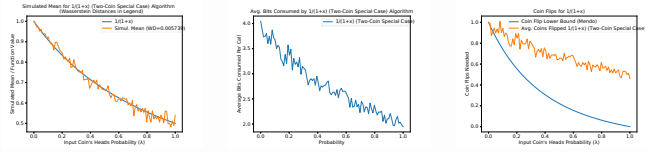
$$1/(1+x) \text{ (Alt. Series)}$$



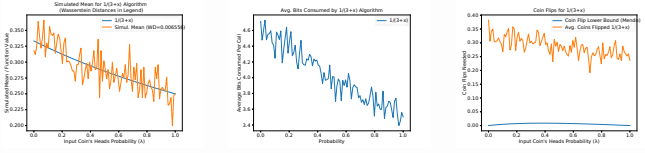
$$1/(1+x) \text{ (Even Parity)}$$



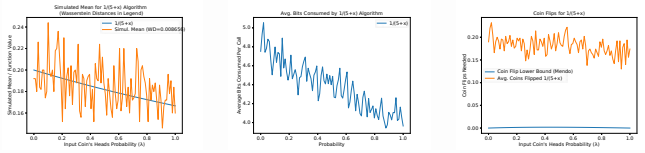
1/(1+x) (Two-Coin Special Case)



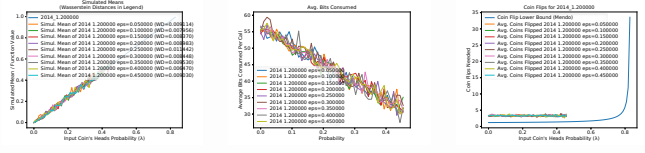
1/(3+x)



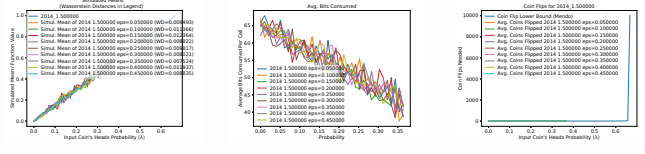
1/(5+x)



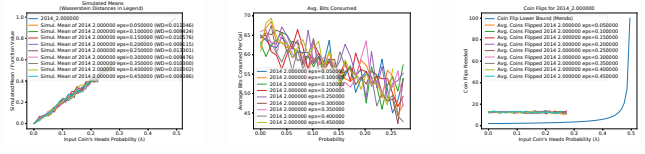
2014 1.200000  
eps=0.050000



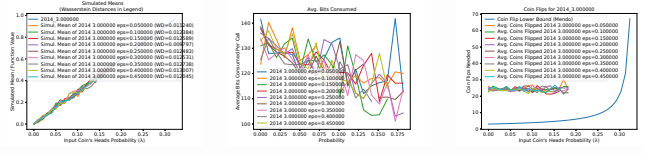
2014 1.500000  
eps=0.050000



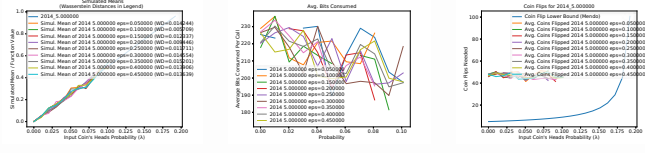
2014 2.000000  
eps=0.050000



2014 3.000000  
eps=0.050000



2014 5.000000  
eps=0.050000



2014 Add. x+0.1

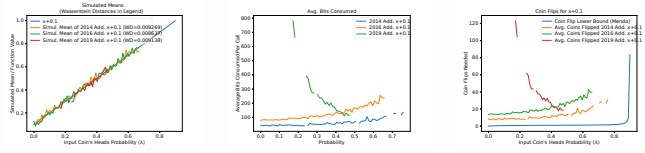


Figure 1 consists of three subplots. The left subplot is a scatter plot titled 'Dispersed Values (Observations minus Logarithm)' showing 'True Value (mm)' on the y-axis versus 'Predicted Value (mm)' on the x-axis, both ranging from 0.0 to 0.8. It includes data points for 2014 (blue), 2015 (green), and 2016 (orange), along with a red line representing the proposed method. The middle subplot is titled 'Avg. RMSE Calculated' and shows 'Average RMSE Calculated (mm)' on the y-axis (0 to 10) versus 'Normalized Error' on the x-axis (0.0 to 0.6). It compares the proposed method (red line) with other methods (blue, green, orange lines). The right subplot is titled 'Cum. Error for 2014-2016' and shows 'Cumulative Error (mm)' on the y-axis (0 to 80) versus 'Normalized Error' on the x-axis (0.0 to 0.6). It compares the proposed method (red line) with other methods (blue, green, orange lines).

[illegible]

Figure 1 consists of three subplots. Subplot (a) is a scatter plot titled 'Scatter Plot of True vs. Predicted mean (F1 score)' showing the relationship between the input mean F1 score and the predicted mean F1 score for 10 datasets. The data points are tightly clustered along the diagonal line, indicating high predictive accuracy. Subplot (b) is titled 'Avg. ROC Curves' and shows the average ROC curves for 10 datasets. The curves are generally above the diagonal, indicating good performance. Subplot (c) is titled 'Avg. ROC Curves' and shows the average ROC curves for 10 datasets, with a zoomed-in view of the low-probability region (0.0 to 0.2 on the x-axis). The curves are generally above the diagonal, indicating good performance.

[illegible]

Figure 1 consists of three subplots labeled (a), (b), and (c), each showing performance metrics against the 'Input CDR's mean Probability (x)' on the x-axis, ranging from 0.0 to 0.5.

Subplot (a) is titled 'Calibration Error (underestimation or overestimation)'. The y-axis is 'Dry peak ratio (underest.)' ranging from 0.0 to 1.0. It shows four lines: 'Proposed' (blue), 'LSE' (green), 'LSE with  $\sigma$ ' (orange), and 'LSE with  $\sigma$  and  $\mu$ ' (purple). The 'Proposed' line is a diagonal line from (0,0) to (1,1). The other lines show varying degrees of underestimation, with 'LSE with  $\sigma$  and  $\mu$ ' showing the most significant underestimation at higher probabilities.

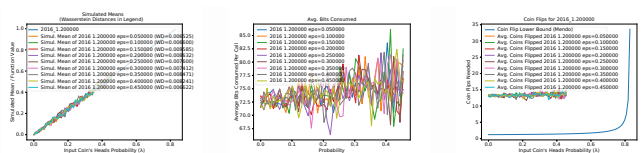
Subplot (b) is titled 'Avg. Bias Coefficient'. The y-axis is 'Average Bias Coefficient' ranging from 0.00 to 0.04. It shows four lines: 'Proposed' (blue), 'LSE' (green), 'LSE with  $\sigma$ ' (orange), and 'LSE with  $\sigma$  and  $\mu$ ' (purple). The 'Proposed' line is a diagonal line from (0,0) to (1,1). The other lines show varying degrees of bias, with 'LSE with  $\sigma$  and  $\mu$ ' showing the highest bias at higher probabilities.

Subplot (c) is titled 'Gain Ratio for  $\sigma$  and  $\mu$ '. The y-axis is 'Gain Ratio' ranging from 0 to 180. It shows four lines: 'Proposed' (blue), 'LSE' (green), 'LSE with  $\sigma$ ' (orange), and 'LSE with  $\sigma$  and  $\mu$ ' (purple). The 'Proposed' line is a diagonal line from (0,0) to (1,1). The other lines show varying degrees of gain, with 'LSE with  $\sigma$  and  $\mu$ ' showing the highest gain at higher probabilities.

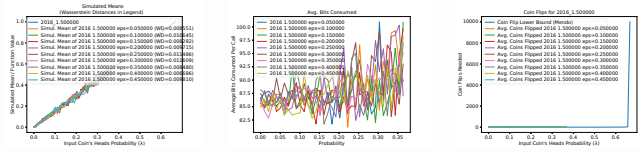
Figure 10 consists of two plots. The left plot shows the average number of clusters (log scale) versus input data size (log scale) for various datasets. The right plot shows the average number of clusters (log scale) versus input data size (log scale) for various datasets, with a legend indicating the datasets used.

[illegible]

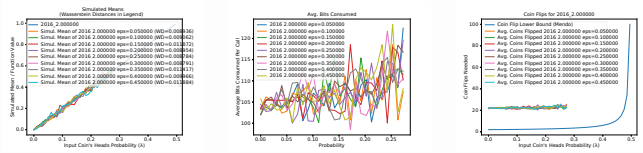
```
2016 1.200000
eps=0.050000
```



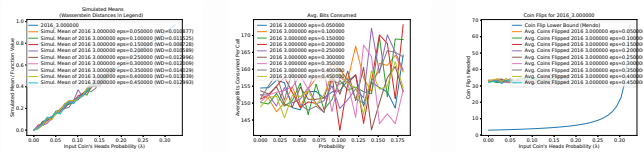
```
2016 1.500000
eps=0.050000
```



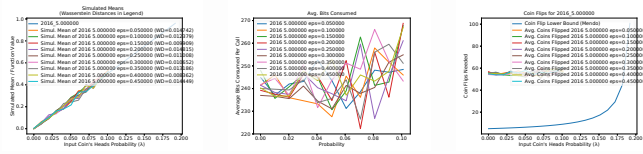
```
2016 2.000000
eps=0.050000
```



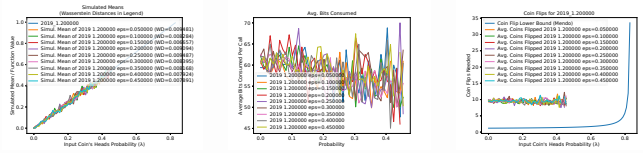
```
2016 3.000000
eps=0.050000
```



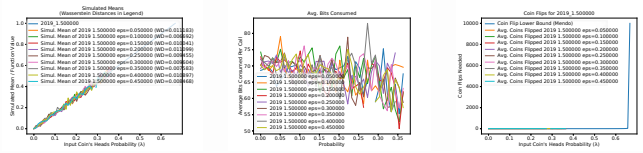
```
2016 5.000000
eps=0.050000
```



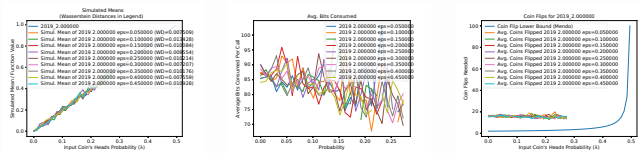
```
2019 1.200000
eps=0.050000
```



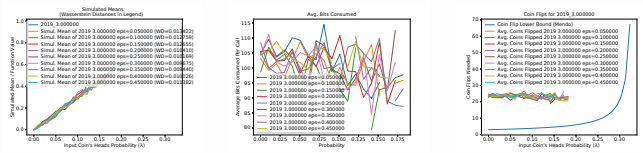
```
2019 1.500000
eps=0.050000
```



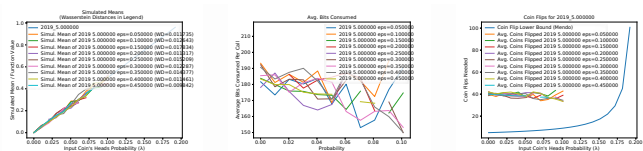
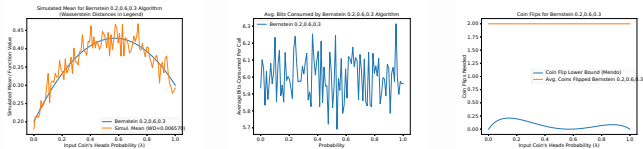
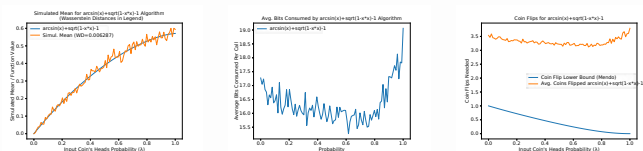
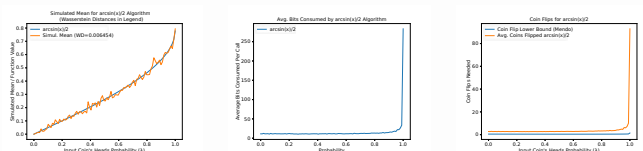
```
2019 2.000000
eps=0.050000
```



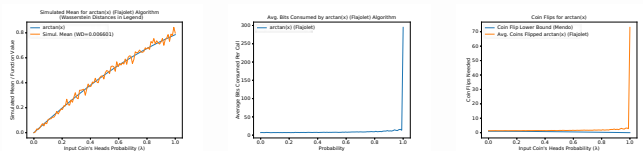
```
2019 3.000000
eps=0.050000
```



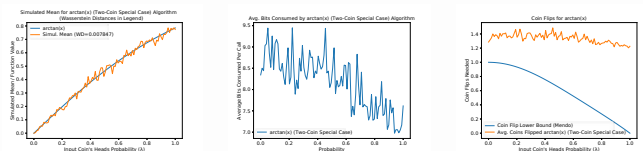
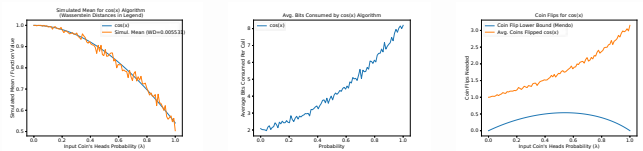
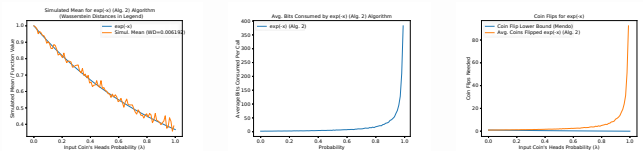
```
2019 5.000000
eps=0.050000
```

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)

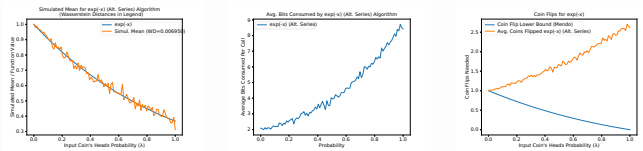


Figure 1 consists of three subplots labeled (a), (b), and (c), each showing performance metrics against the input data's smooth probability (x-axis, ranging from 0.0 to 1.0).

- (a) Scalable Mean for expected  $T_1$ :** The y-axis is 'Scalable Mean for expected  $T_1$ ' (ranging from 0.0 to 1.4). The legend includes 'expected  $T_1$ ' (blue line), 'observed  $T_1$ ' (orange line), and 'observed  $T_1$  - Bias (0.00718)' (green line). The observed  $T_1$  starts at approximately 1.3 and decreases to about 0.4. The observed  $T_1$  minus bias starts at approximately 1.2 and decreases to about 0.3.
- (b) Avg. Bias Computed by expected  $T_1$  Algorithm:** The y-axis is 'Avg. Bias Computed by expected  $T_1$  Algorithm' (ranging from 0 to 1.5). The legend includes 'expected  $T_1$ ' (blue line). The bias starts at approximately 0.1 and increases to about 1.4.
- (c) Avg. RMSE Computed by expected  $T_1$  Algorithm:** The y-axis is 'Avg. RMSE Computed by expected  $T_1$  Algorithm' (ranging from 0.0 to 2.5). The legend includes 'Cox-Fit Linear Band Bound' (blue line), 'Cox-Fit Linear Band Bound' (orange line), and 'Avg. RMSE expected  $T_1$ ' (green line). The Cox-Fit Linear Band Bound starts at approximately 0.8 and increases to about 2.4. The Avg. RMSE expected  $T_1$  starts at approximately 0.8 and increases to about 2.4.

Figure 1 consists of three subplots labeled (a), (b), and (c), each showing performance metrics versus  $\log_2(x)$  (ranging from 0 to 14).

- (a) Standard Error (Y-axis):** The proposed algorithm (blue line) shows a significantly lower standard error than the baseline (orange line) across the entire range of  $\log_2(x)$ . The blue line starts at approximately 0.05 and increases slightly to 0.1, while the orange line starts at 0.05 and increases to approximately 0.25.
- (b) Avg. Bits Consumed (Y-axis):** The proposed algorithm (blue line) consumes fewer bits than the baseline (orange line) for  $\log_2(x) > 10$ . The blue line remains relatively flat around 10 bits, while the orange line increases sharply to approximately 15 bits for  $\log_2(x) > 10$ .
- (c) Cost (Y-axis):** The proposed algorithm (blue line) has a lower cost than the baseline (orange line) for  $\log_2(x) > 10$ . The blue line remains relatively flat around 0.5, while the orange line increases sharply to approximately 1.5 for  $\log_2(x) > 10$ .

Figure 1 consists of three subplots illustrating the performance of the proposed algorithm. The x-axis for all plots is  $t$ , ranging from 0.0 to 1.0.

- Left Subplot:** O-Value (Mean Squared Error) vs  $t$ . The y-axis ranges from 0.0 to 0.4. Two lines are shown:  $\log_2-1$  (blue line) and  $\log_2-2$  (orange line). Both lines show a decreasing trend, starting around 0.4 at  $t=0$  and ending around 0.1 at  $t=1$ .
- Middle Subplot:** Avg. Wts. Computed by  $\log_2-1(t)$  vs  $t$ . The y-axis ranges from 0.0 to 1.0. Two lines are shown:  $\log_2-1$  (blue line) and  $\log_2-1$  (Then Com. Special Code) (orange line). Both lines show a decreasing trend, starting around 0.9 at  $t=0$  and ending around 0.1 at  $t=1$ .
- Right Subplot:** Cost (Avg. Weight) vs  $t$ . The y-axis ranges from 0.0 to 2.0. Two lines are shown: Cost (Log Linear Model) (blue line) and Cost (Prop. Special Code) (orange line). The blue line starts around 1.4 and decreases to around 0.2. The orange line starts around 1.9 and decreases to around 1.6.

Figure 1 consists of three subplots labeled (a), (b), and (c).

Subplot (a) is titled "Dry-run time for  $\text{pmc}_{\text{v1}}(21)$  algorithm". The x-axis is "Number of nodes (nodes) (x1000)" ranging from 0.0 to 1.0. The y-axis is "Dry-run time (x1000) (s)" ranging from 0.0 to 1.4. It shows two curves: a blue line for "pmc<sub>v1</sub>(21)" and an orange line for "Exact. Mean (std=0.00086)". Both curves show an increasing trend, with the blue line being slightly higher than the orange line.

Subplot (b) is titled "Avg. CPU Consumed by  $\text{pmc}_{\text{v1}}(21)$  Algorithm". The x-axis is "Number of nodes (nodes) (x1000)" ranging from 0.0 to 1.0. The y-axis is "Average CPU consumed (s)" ranging from 0.0 to 4.0. It shows a blue line for "pmc<sub>v1</sub>(21)" which fluctuates and generally increases from approximately 0.5 to 3.5.

Subplot (c) is titled "Cost Files for  $\text{pmc}_{\text{v1}}(21)$ ". The x-axis is "Number of nodes (nodes) (x1000)" ranging from 0.0 to 1.0. The y-axis is "Cost" ranging from 0.000 to 2.000. It shows three curves: a blue line for "Cost File Lower Bound (Std=0.00000)", an orange line for "Avg. Cost (Upper bound)", and a green line for "Cost File Upper Bound (Std=0.00000)". The blue and green lines are very close to each other and follow a similar increasing trend, while the orange line is slightly higher and also increases.

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

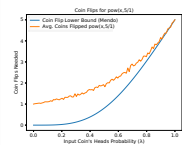
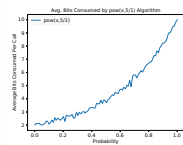
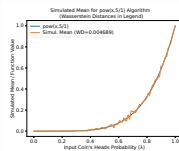
- (a) Scalability:** The y-axis is 'Standard Error in Log10 Scale' (0.0 to 1.0). The x-axis is 'Input Correlation (0.0 to 1.0)'. Two lines are shown: 'pwr2.245' (blue) and 'CovReg-Weiss (0.01-0.000222)' (orange). Both lines show an increasing trend, with the orange line generally higher than the blue line.
- (b) Average SE:** The y-axis is 'Average SE (Logarithmic Scale)' (0.0 to 100). The x-axis is 'Input Correlation (0.0 to 1.0)'. Two lines are shown: 'pwr2.245' (blue) and 'pwr2.245' (orange). Both lines show a sharp decrease in Average SE as input correlation increases, stabilizing near zero for correlations above 0.5.
- (c) CPU Time:** The y-axis is 'CPU Time (seconds)' (0.0 to 100). The x-axis is 'Input Correlation (0.0 to 1.0)'. Two lines are shown: 'Cov-File Lower Bound (blue)' and 'Cov-File Upper Bound (orange)'. Both lines show a sharp decrease in CPU time as input correlation increases, stabilizing near zero for correlations above 0.5.

Figure 1 consists of three subplots labeled (a), (b), and (c), comparing the performance of the proposed algorithm (points 3, 4) against a baseline (points 1, 2).

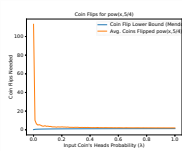
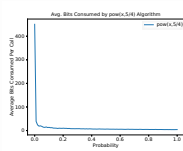
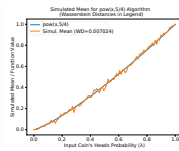
- (a) Standard Error vs. Capital Asset Turnover Ratio:** The y-axis is 'Standard Error (1000000000)' ranging from 0.0 to 1.0. The x-axis is 'Capital Asset Turnover Ratio (10)' ranging from 0.0 to 1.0. The legend indicates 'points 3, 4' (blue line) and 'Baseline (points 1, 2) (red line)'. Both lines show a decreasing trend, with the proposed algorithm's error being slightly lower than the baseline's.
- (b) Avg. Bits Consumed vs. Precision:** The y-axis is 'Avg. Bits Consumed (100000000)' ranging from 0.0 to 1.0. The x-axis is 'Precision' ranging from 0.0 to 1.0. The legend indicates 'points 3, 4' (blue line). The curve starts at approximately 0.8 bits for low precision and drops sharply to near zero as precision increases.
- (c) Cost Functions vs. Capital Asset Turnover Ratio:** The y-axis is 'Cost Functions' ranging from 0.0 to 1.6. The x-axis is 'Capital Asset Turnover Ratio (10)' ranging from 0.0 to 1.0. The legend indicates 'Cost Fcn Upper Bound (red line)', 'Cost Fcn Upper Bound (blue line)', and 'Cost Fcn Upper Bound (red line)'. The curves show a sharp initial drop from a value above 1.6 to near zero, followed by a slight increase and then a plateau.

Figure 1 consists of three subplots. Subplot (a) is a line graph titled 'Empirical Mean of p-value, K/S Algorithm' showing the 'Empirical Mean of p-value, K/S Algorithm' on the y-axis (ranging from 0.0 to 1.0) against the 'Empirical Mean of p-value, K/S Algorithm' on the x-axis (ranging from 0.0 to 1.0). It includes a blue line for 'p-value, K/S Algorithm', an orange line for 'p-value, K/S Algorithm', and a red line for 'p-value, K/S Algorithm'. Subplot (b) is a line graph titled 'Avg. Bias Computed by p-value, K/S Algorithm' showing the 'Avg. Bias Computed by p-value, K/S Algorithm' on the y-axis (ranging from 0.0 to 0.4) against the 'p-value, K/S' on the x-axis (ranging from 0.0 to 1.0). It includes a blue line for 'p-value, K/S' and an orange line for 'p-value, K/S'. Subplot (c) is a line graph titled 'Corr. P-Val. vs. Signal-to-Noise Ratio (SNR)' showing the 'Corr. P-Val.' on the y-axis (ranging from 0.0 to 0.5) against the 'Signal-to-Noise Ratio (SNR)' on the x-axis (ranging from 0.0 to 1.0). It includes a blue line for 'p-value, K/S' and an orange line for 'p-value, K/S'.

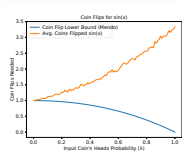
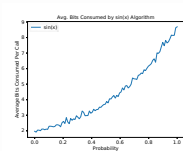
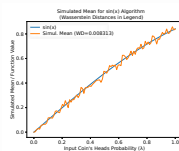
pow(x,5/1)



pow(x,5/4)



sin(x)



sqrt(x)

