

## # 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—

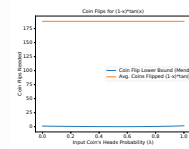
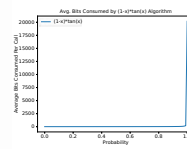
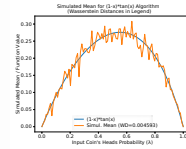
- an  $\epsilon$  of  $1 - (x + c) * 1.001$  was used (or 0.0001 if  $\epsilon$  would be greater than 1), and
- an  $\epsilon$  of  $(x - c) * 0.9995$  for the subtraction variants.

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.

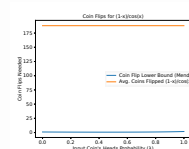
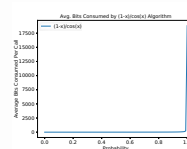
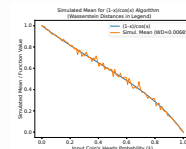
## 0.1 The Charts

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

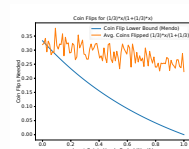
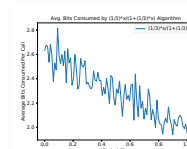
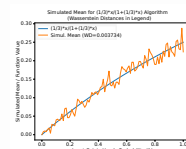
$$(1-x)*\tan(x)$$



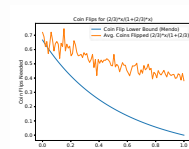
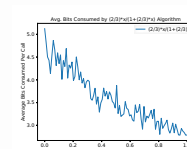
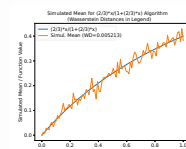
$$(1-x)/\cos(x)$$



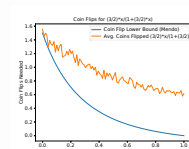
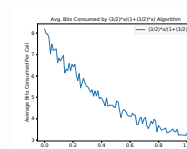
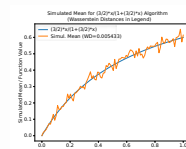
$$(1/3)*x/(1+(1/3)*x)$$



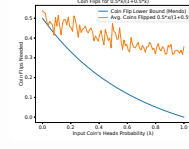
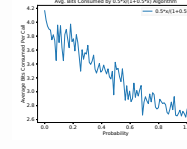
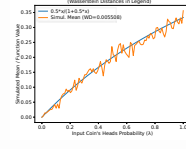
$$(2/3)*x/(1+(2/3)*x)$$



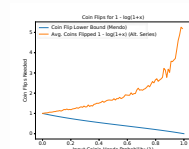
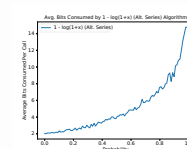
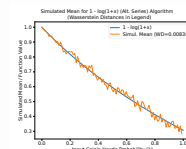
$$(3/2)*x/(1+(3/2)*x)$$



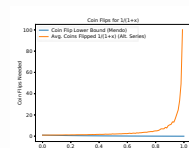
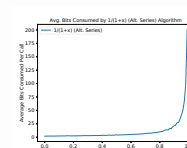
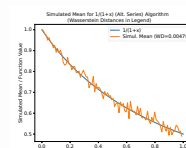
$$0.5*x/(1+0.5*x)$$



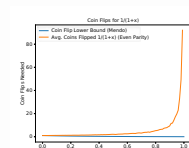
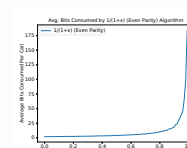
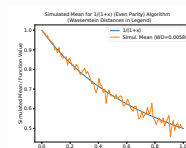
$$1 - \ln(1+x) \text{ (Alt. Series)}$$



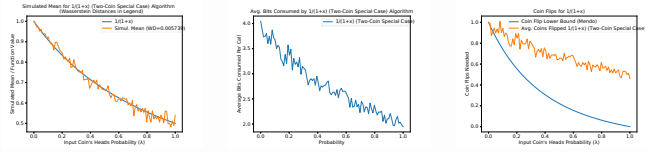
$$1/(1+x) \text{ (Alt. Series)}$$



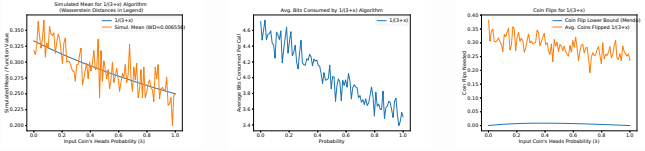
$$1/(1+x) \text{ (Even Parity)}$$



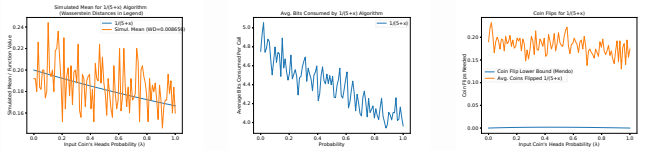
1/(1+x) (Two-Coin Special Case)



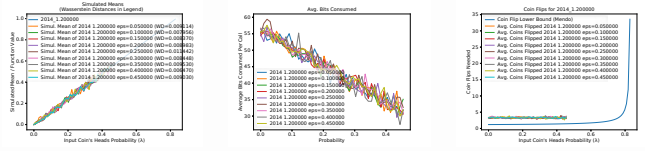
1/(3+x)



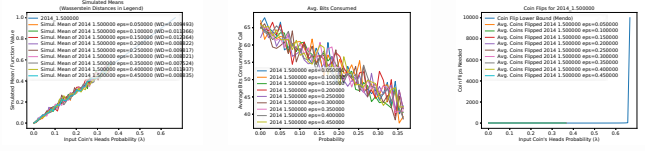
1/(5+x)



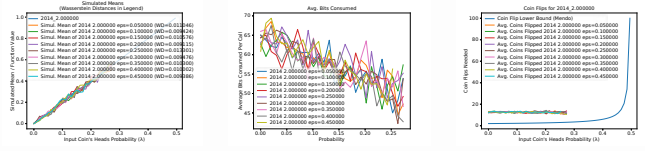
2014 1.200000  
eps=0.050000



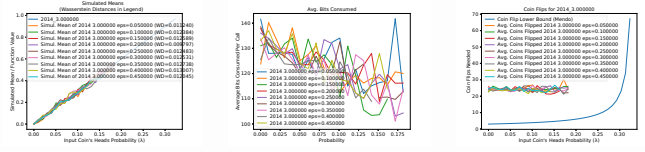
2014 1.500000  
eps=0.050000



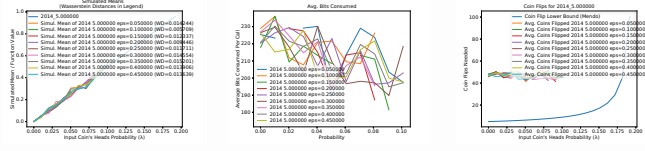
2014 2.000000  
eps=0.050000



2014 3.000000  
eps=0.050000



2014 5.000000  
eps=0.050000



2014 Add. x+0.1

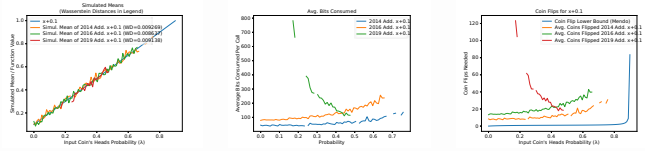


Figure 1 consists of three subplots. The left subplot, titled 'Dispersed Values (Observations vs. Predictions)', shows a scatter plot of True Values (mm) on the y-axis versus Predicted Values (mm) on the x-axis, both ranging from 0.0 to 1.0. It includes data for 2014 (blue), 2015 (orange), and 2016 (green), along with a red line for the proposed method and a blue line for the reference. The middle subplot, titled 'Avg. RMSE Computed', 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 displays the same data series as the left plot. The right subplot, titled 'Error Histogram for 2014-2016', shows the number of observations on the y-axis (0 to 80) versus Normalized Error on the x-axis (0.0 to 0.7). It includes histograms for 2014 (blue), 2015 (orange), and 2016 (green), along with a red line for the proposed method.

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 (0.6 to 0.8).

The top-left plot shows the Stratified Mean AUC (std) on the y-axis (0.6 to 0.8) versus AUC on the x-axis (0.6 to 0.8). The legend indicates the following methods and years:

- 2014, AUC=0.6
- 2014, AUC=0.7
- 2015, AUC=0.6
- 2015, AUC=0.7
- 2016, AUC=0.6
- 2016, AUC=0.7
- 2017, AUC=0.6
- 2017, AUC=0.7
- 2018, AUC=0.6
- 2018, AUC=0.7
- 2019, AUC=0.6
- 2019, AUC=0.7

The top-right plot shows the Average ROC Curves (std) on the y-axis (0.6 to 0.8) versus AUC on the x-axis (0.6 to 0.8). The legend indicates the following methods and years:

- 2014, AUC=0.6
- 2014, AUC=0.7
- 2015, AUC=0.6
- 2015, AUC=0.7
- 2016, AUC=0.6
- 2016, AUC=0.7
- 2017, AUC=0.6
- 2017, AUC=0.7
- 2018, AUC=0.6
- 2018, AUC=0.7
- 2019, AUC=0.6
- 2019, AUC=0.7

The bottom plot shows the CUP Filter for AUC=0.5 on the y-axis (0.6 to 0.8) versus CUP Filter for AUC=0.5 on the x-axis (0.6 to 0.8). The legend indicates the following methods and years:

- CUP Filter Lower Bound (std)
- CUP Filter (std)
- CUP Filter Upper Bound (std)
- 2014, AUC=0.6
- 2014, AUC=0.7
- 2015, AUC=0.6
- 2015, AUC=0.7
- 2016, AUC=0.6
- 2016, AUC=0.7
- 2017, AUC=0.6
- 2017, AUC=0.7
- 2018, AUC=0.6
- 2018, AUC=0.7
- 2019, AUC=0.6
- 2019, AUC=0.7

[illegible][illegible]

Figure 1 consists of three subplots. Subplot (a) is a scatter plot titled 'Scatter Plot (Covariance)' showing 'Predicted Mean (Covariance)' on the y-axis versus 'Input C1: Mean Probability (C1)' on the x-axis. Both axes range from 0.0 to 0.5. Data points for various datasets are plotted, showing a strong positive linear correlation. Subplot (b) is a line plot titled 'Avg. F1 Score for v2-0' showing 'Average F1 Score' on the y-axis (0.0 to 1.0) versus 'Probability' on the x-axis (0.0 to 0.5). It compares 'Proposed Model' (blue line) with '2008 Lin.' (orange line) and '2008 Wts.' (green line). The proposed model maintains a high F1 score (near 1.0) across all probabilities, while the other two models drop significantly as probability increases. Subplot (c) is a line plot titled 'Cost Ratio for v2-0' showing 'Cost Ratio' on the y-axis (0 to 100) versus 'Input C1: Mean Probability (C1)' on the x-axis (0.0 to 0.5). It compares 'Proposed Model' (blue line) with '2008 Lin.' (orange line) and '2008 Wts.' (green line). The proposed model's cost ratio remains low (near 0) until a probability of about 0.4, then rises sharply. The other two models show much higher and more fluctuating cost ratios.

[illegible]

Figure 1 consists of three subplots comparing the proposed method (solid lines) with the baseline method (dashed lines) for different input correlations.

**Top Left Plot: Distorted Net Output (kPa) vs Input Cor. Net's Probability (%)**

This plot shows the distorted net output for various input correlations. The x-axis ranges from 0.00 to 0.15, and the y-axis ranges from 0 to 1.5. The legend indicates the following input correlations and methods:

- $\sigma=0.5$  (Blue line)
- Distorted Net of 2048 Luts,  $\sigma=0.5$  (Dashed Blue line)
- Distorted Net of 2048 Luts,  $\sigma=0.5$  (Solid Blue line)
- Distorted Net of 2048 Luts,  $\sigma=0.5$  (Dashed Green line)
- Distorted Net of 2048 Luts,  $\sigma=0.5$  (Solid Green line)
- Distorted Net of 2048 Luts,  $\sigma=0.5$  (Dashed Purple line)
- Distorted Net of 2048 Luts,  $\sigma=0.5$  (Solid Purple line)
- Distorted Net of 2048 Luts,  $\sigma=0.5$  (Dashed Red line)
- Distorted Net of 2048 Luts,  $\sigma=0.5$  (Solid Red line)
- Distorted Net of 2048 Luts,  $\sigma=0.5$  (Dashed Brown line)
- Distorted Net of 2048 Luts,  $\sigma=0.5$  (Solid Brown line)
- Distorted Net of 2048 Luts,  $\sigma=0.5$  (Dashed Black line)
- Distorted Net of 2048 Luts,  $\sigma=0.5$  (Solid Black line)

**Top Right Plot: Avg. Cor. Net's Prob. vs Probability**

This plot shows the average correlation net's probability for various input correlations. The x-axis ranges from 0.00 to 0.15, and the y-axis ranges from 0 to 100. The legend indicates the following input correlations and methods:

- 2048 Luts,  $\sigma=0.5$  (Blue line)
- 2048 Luts,  $\sigma=0.5$  (Dashed Blue line)
- 2048 Luts,  $\sigma=0.5$  (Solid Blue line)
- 2048 Luts,  $\sigma=0.5$  (Dashed Green line)
- 2048 Luts,  $\sigma=0.5$  (Solid Green line)
- 2048 Luts,  $\sigma=0.5$  (Dashed Purple line)
- 2048 Luts,  $\sigma=0.5$  (Solid Purple line)
- 2048 Luts,  $\sigma=0.5$  (Dashed Red line)
- 2048 Luts,  $\sigma=0.5$  (Solid Red line)
- 2048 Luts,  $\sigma=0.5$  (Dashed Brown line)
- 2048 Luts,  $\sigma=0.5$  (Solid Brown line)
- 2048 Luts,  $\sigma=0.5$  (Dashed Black line)
- 2048 Luts,  $\sigma=0.5$  (Solid Black line)

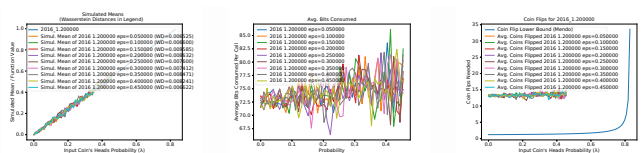
**Bottom Plot: Cor. Net Prob. vs Input Cor. Net's Probability (%)**

This plot shows the correlation net's probability for various input correlations. The x-axis ranges from 0.00 to 0.15, and the y-axis ranges from 0 to 150. The legend indicates the following input correlations and methods:

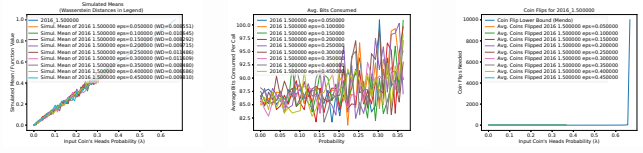
- Cor. Net of 2048 Luts,  $\sigma=0.5$  (Blue line)
- Cor. Net of 2048 Luts,  $\sigma=0.5$  (Dashed Blue line)
- Cor. Net of 2048 Luts,  $\sigma=0.5$  (Solid Blue line)
- Cor. Net of 2048 Luts,  $\sigma=0.5$  (Dashed Green line)
- Cor. Net of 2048 Luts,  $\sigma=0.5$  (Solid Green line)
- Cor. Net of 2048 Luts,  $\sigma=0.5$  (Dashed Purple line)
- Cor. Net of 2048 Luts,  $\sigma=0.5$  (Solid Purple line)
- Cor. Net of 2048 Luts,  $\sigma=0.5$  (Dashed Red line)
- Cor. Net of 2048 Luts,  $\sigma=0.5$  (Solid Red line)
- Cor. Net of 2048 Luts,  $\sigma=0.5$  (Dashed Brown line)
- Cor. Net of 2048 Luts,  $\sigma=0.5$  (Solid Brown line)
- Cor. Net of 2048 Luts,  $\sigma=0.5$  (Dashed Black line)
- Cor. Net of 2048 Luts,  $\sigma=0.5$  (Solid Black line)

[illegible]

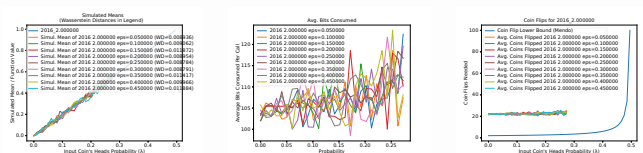
```
2016 1.200000
eps=0.050000
```



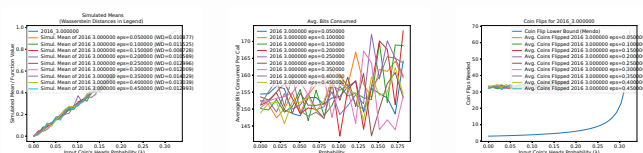
```
2016 1.500000
eps=0.050000
```



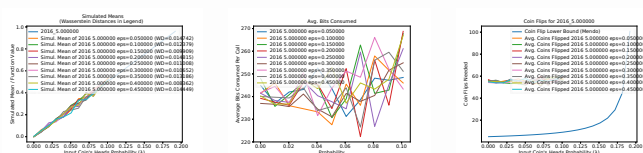
```
2016 2.000000
eps=0.050000
```



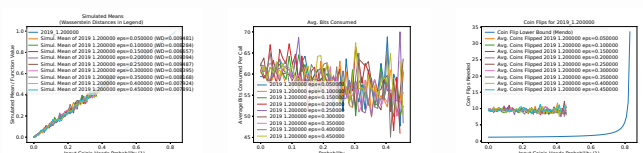
```
2016 3.000000
eps=0.050000
```



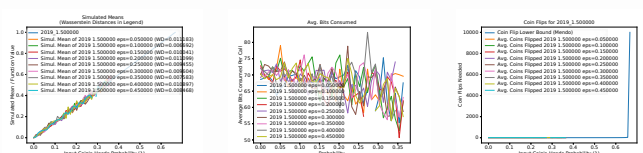
```
2016 5.000000
eps=0.050000
```



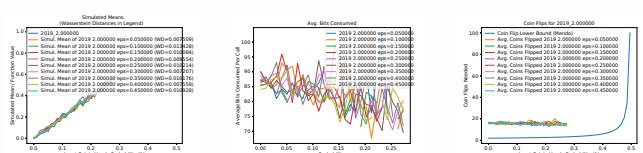
```
2019 1.200000
eps=0.050000
```



```
2019 1.500000
eps=0.050000
```



```
2019 2.000000
eps=0.050000
```



```
2019 3.000000
eps=0.050000
```

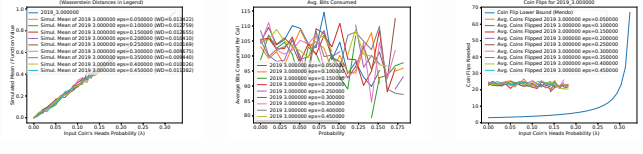


Figure 10 displays simulation results for various algorithms, organized into a 10x3 grid. Each row corresponds to a specific algorithm. The columns show: 1) Simulated Mean for the algorithm (with error bars and legend), 2) Average Bits Consumed per Call vs. Probability, and 3) Coin Flips for the algorithm (with upper and lower bounds).

- Row 1: Bernoulli (2,0,0,0,0) Algorithm**
  - Simulated Mean: Shows a linear increase from 0 to 1.0 as input coin's heads probability increases from 0.0 to 1.0. Legend includes Bernoulli (2,0,0,0,0) and Simul. Mean (SD=0.00000).
  - Average Bits Consumed: Fluctuates between approximately 150 and 200 bits per call across the probability range.
  - Coin Flips: Shows a sharp increase in coin flips as the input probability approaches 1.0, with the upper bound reaching 1000 and the lower bound around 250.
- Row 2: Bernoulli (2,0,0,0,1) Algorithm**
  - Simulated Mean: Shows a linear increase from 0 to 1.0. Legend includes Bernoulli (2,0,0,0,1) and Simul. Mean (SD=0.00000).
  - Average Bits Consumed: Fluctuates between approximately 150 and 200 bits per call.
  - Coin Flips: Shows a sharp increase in coin flips as the input probability approaches 1.0, with the upper bound reaching 1000 and the lower bound around 250.
- Row 3: Bernoulli (2,0,0,0,2) Algorithm**
  - Simulated Mean: Shows a linear increase from 0 to 1.0. Legend includes Bernoulli (2,0,0,0,2) and Simul. Mean (SD=0.00000).
  - Average Bits Consumed: Fluctuates between approximately 150 and 200 bits per call.
  - Coin Flips: Shows a sharp increase in coin flips as the input probability approaches 1.0, with the upper bound reaching 1000 and the lower bound around 250.
- Row 4: Bernoulli (2,0,0,0,3) Algorithm**
  - Simulated Mean: Shows a linear increase from 0 to 1.0. Legend includes Bernoulli (2,0,0,0,3) and Simul. Mean (SD=0.00000).
  - Average Bits Consumed: Fluctuates between approximately 150 and 200 bits per call.
  - Coin Flips: Shows a sharp increase in coin flips as the input probability approaches 1.0, with the upper bound reaching 1000 and the lower bound around 250.
- Row 5: Bernoulli (2,0,0,0,4) Algorithm**
  - Simulated Mean: Shows a linear increase from 0 to 1.0. Legend includes Bernoulli (2,0,0,0,4) and Simul. Mean (SD=0.00000).
  - Average Bits Consumed: Fluctuates between approximately 150 and 200 bits per call.
  - Coin Flips: Shows a sharp increase in coin flips as the input probability approaches 1.0, with the upper bound reaching 1000 and the lower bound around 250.
- Row 6: Bernoulli (2,0,0,0,5) Algorithm**
  - Simulated Mean: Shows a linear increase from 0 to 1.0. Legend includes Bernoulli (2,0,0,0,5) and Simul. Mean (SD=0.00000).
  - Average Bits Consumed: Fluctuates between approximately 150 and 200 bits per call.
  - Coin Flips: Shows a sharp increase in coin flips as the input probability approaches 1.0, with the upper bound reaching 1000 and the lower bound around 250.
- Row 7: Bernoulli (2,0,0,0,6) Algorithm**
  - Simulated Mean: Shows a linear increase from 0 to 1.0. Legend includes Bernoulli (2,0,0,0,6) and Simul. Mean (SD=0.00000).
  - Average Bits Consumed: Fluctuates between approximately 150 and 200 bits per call.
  - Coin Flips: Shows a sharp increase in coin flips as the input probability approaches 1.0, with the upper bound reaching 1000 and the lower bound around 250.
- Row 8: Bernoulli (2,0,0,0,7) Algorithm**
  - Simulated Mean: Shows a linear increase from 0 to 1.0. Legend includes Bernoulli (2,0,0,0,7) and Simul. Mean (SD=0.00000).
  - Average Bits Consumed: Fluctuates between approximately 150 and 200 bits per call.
  - Coin Flips: Shows a sharp increase in coin flips as the input probability approaches 1.0, with the upper bound reaching 1000 and the lower bound around 250.
- Row 9: Bernoulli (2,0,0,0,8) Algorithm**
  - Simulated Mean: Shows a linear increase from 0 to 1.0. Legend includes Bernoulli (2,0,0,0,8) and Simul. Mean (SD=0.00000).
  - Average Bits Consumed: Fluctuates between approximately 150 and 200 bits per call.
  - Coin Flips: Shows a sharp increase in coin flips as the input probability approaches 1.0, with the upper bound reaching 1000 and the lower bound around 250.
- Row 10: Bernoulli (2,0,0,0,9) Algorithm**
  - Simulated Mean: Shows a linear increase from 0 to 1.0. Legend includes Bernoulli (2,0,0,0,9) and Simul. Mean (SD=0.00000).
  - Average Bits Consumed: Fluctuates between approximately 150 and 200 bits per call.
  - Coin Flips: Shows a sharp increase in coin flips as the input probability approaches 1.0, with the upper bound reaching 1000 and the lower bound around 250.

Figure 1 consists of three subplots labeled (a), (b), and (c), each showing performance metrics against the normalized GCM iterations (x-axis, ranging from 0.0 to 1.0).

- (a) Scaled error vs. normalized GCM iterations:** The y-axis is 'Scaled error (arbitrary unit)' ranging from 0.0 to 1.0. The legend indicates 'exact' (blue line) and 'Simplified error (proposed algorithm)' (orange line with markers). Both curves show a decreasing trend, with the proposed algorithm's error closely following the exact error.
- (b) Average bit complexity vs. normalized GCM iterations:** The y-axis is 'Average bit complexity (bit)' ranging from 0.0 to 1.0. The legend indicates 'exact' (blue line) and 'approx of (proposed)' (orange line with markers). The exact complexity is a constant blue line at approximately 0.25 bit. The proposed algorithm's complexity starts at the same level but increases sharply as the normalized GCM iterations approach 1.0.
- (c) GCM steps vs. normalized GCM iterations:** The y-axis is 'GCM steps' ranging from 0 to 250. The legend indicates 'Exact (proposed algorithm)' (blue line) and 'Approx (proposed algorithm)' (orange line with markers). The exact steps are a constant blue line at approximately 10 steps. The proposed algorithm's steps start at the same level but increase sharply as the normalized GCM iterations approach 1.0.

Figure 10 consists of three subplots. The left subplot, titled 'Scalability of the proposed algorithm', shows the 'Scalability of the proposed algorithm' on the y-axis (ranging from 0.0 to 1.0) versus 'Input Data Size (Number of Nodes)' on the x-axis (ranging from 0.0 to 1.0). It compares 'Proposed' (blue line) and 'Baseline' (orange line) algorithms. Both show a decreasing trend, with the proposed algorithm maintaining higher scalability at larger input sizes. The middle subplot, titled 'Avg. Bits Consumed by the proposed algorithm', shows 'Avg. Bits Consumed (in bits)' on the y-axis (ranging from 0 to 10) versus 'Input Data Size (Number of Nodes)' on the x-axis (ranging from 0.0 to 1.0). It compares 'Proposed' (blue line) and 'Baseline' (orange line) algorithms. Both show an increasing trend, with the proposed algorithm consuming fewer bits. The right subplot, titled 'Gap Error for input data size', shows 'Gap Error (in bits)' on the y-axis (ranging from 0.0 to 2.0) versus 'Input Data Size (Number of Nodes)' on the x-axis (ranging from 0.0 to 1.0). It compares 'Proposed' (blue line) and 'Baseline' (orange line) algorithms. Both show an increasing trend, with the proposed algorithm having a lower gap error.

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

(a) Standard Mean for  $\log_2(x+1)$  algorithm. The x-axis is 'Input Cost's Inverse Probability (%)' from 0.0 to 1.0. The y-axis is 'Standard Mean for  $\log_2(x+1)$  (Max)' from 0.0 to 1.0. The legend includes:  $\log_2(x+1)$  (blue line),  $\log_2(x+1)$  (orange line), and  $\log_2(x+1)$  (red line). All three lines are nearly identical, showing a linear increase from (0,0) to (1,1).

(b) Avg. Mem. Consumed by  $\log_2(x+1)$  algorithm. The x-axis is 'Probability' from 0.0 to 1.0. The y-axis is 'Memory (Count of bits)' from 0 to 60. The legend includes:  $\log_2(x+1)$  (blue line) and  $\log_2(x+1)$  (orange line). Both lines are flat at approximately 10 units until a probability of about 0.9, then rise sharply to about 55 units at probability 1.0.

(c) Cost Ratio for  $\log_2(x+1)$ . The x-axis is 'Input Cost's Inverse Probability (%)' from 0.0 to 1.0. The y-axis is 'Cost Ratio' from 0.0 to 1.4. The legend includes: Cost Ratio Lower Bound (Shaded),  $\log_2(x+1)$  (blue line), and  $\log_2(x+1)$  (orange line). The blue line is flat at approximately 0.25. The orange line is flat at approximately 0.25 until a probability of about 0.9, then rises sharply to 1.0 at probability 1.0. A shaded region represents the lower bound, starting around 0.25 and ending at 1.0.

Figure 1 consists of three subplots labeled (a), (b), and (c), each showing performance metrics versus Signal-to-Noise Ratio (SNR) on the x-axis (ranging from 0.0 to 1.0).

- (a) Standard Error (Std) (dB):** The y-axis ranges from -10 to 0. The legend indicates:
  - Blue line:  $\log(x)$  (Then Coen Special Case Algorithm)
  - Orange line:  $\log(x)$  (Then Coen Special Case Algorithm)
  - Red line:  $\log(x)$  (Then Coen Special Case Algorithm)
 All three lines overlap and show a decreasing trend from approximately -5 dB at SNR 0.0 to -10 dB at SNR 1.0.
- (b) Average Bias (Percent) (%):** The y-axis ranges from -10 to 10. The legend indicates:
  - Blue line:  $\log(x)$  (Then Coen Special Case Algorithm)
  - Orange line:  $\log(x)$  (Then Coen Special Case Algorithm)
  - Red line:  $\log(x)$  (Then Coen Special Case Algorithm)
 All three lines overlap and show a decreasing trend from approximately 5% at SNR 0.0 to -10% at SNR 1.0.
- (c) Coverage Rate (Percent) (%):** The y-axis ranges from 0.0 to 1.0. The legend indicates:
  - Blue line: Coen Alg Lower Bound (Shaded)
  - Orange line: Coen Alg Upper Bound (Shaded)
  - Red line: Coen Alg (Then Coen Special Case Algorithm)
 The blue and orange lines form a shaded region representing the Coen Alg Lower Bound and Coen Alg Upper Bound, respectively. The red line represents the Coen Alg (Then Coen Special Case Algorithm), which is a solid line that decreases from approximately 0.8 at SNR 0.0 to 0.6 at SNR 1.0.

Figure 1 consists of three subplots. The top-left plot, titled 'Simulated Results for points 1,2,3 Algorithm', shows the Standard Error (CV) on Test Data (Y-axis, 0.0 to 1.8) versus Signal-to-Noise Ratio (SNR) in dB (X-axis, -10 to 10). It compares two cases: 'points 1,2,3' (blue line) and 'Lower Bound (dB=0.00000000)' (orange line). The top-right plot, titled 'Avg. Bits Consumed by points 1,2,3 Algorithm', shows the Average Bits Consumed per CV of  $\hat{\mathbf{A}}$  (Y-axis, 0 to 4000) versus Probability (X-axis, 0.0 to 1.0). It compares 'points 1,2,3' (blue line) and 'points 1,2,3,4,5,6,7,8,9,10' (orange line). The bottom plot, titled 'Gain Plots for points 1,2,3', shows the Gain (bits per second) (Y-axis, 0 to 400) versus Signal-to-Noise Ratio (SNR) in dB (X-axis, -10 to 10). It compares the 'Gain Plot Lower Bound (dB)' (blue line) and the 'Avg. Gain (Proposed points 1,2,3)' (orange line).

Figure 1 consists of three subplots showing the performance of the proposed algorithm for psws-2(1) across different input and output probabilities (0.0 to 1.0).

- Left Plot:** Total New Connections (Y-axis, 0.0 to 1.0) vs. Input Node's Average Probability (X-axis, 0.0 to 1.0). The plot shows two curves: a blue line for psws-2(1) and an orange line for a Baseline (psws-0(0.0)0.0). Both curves show an increasing trend, with the blue curve being slightly higher than the orange curve.
- Middle Plot:** Average Bit Connections (Y-axis, -2.0 to 6.0) vs. Probability (X-axis, 0.0 to 1.0). The plot shows a single blue line for psws-2(1), which shows a fluctuating but generally increasing trend.
- Right Plot:** Cost Flaps for psws-2(1) (Y-axis, 0.00 to 2.00) vs. Output Cost's Average Probability (X-axis, 0.0 to 1.0). The plot shows two curves: a blue line for Cost Flaps Lower Bound (psws-2(1)) and an orange line for Cost Flaps Upper Bound (psws-2(1)). Both curves show an increasing trend, with the orange curve being slightly higher than the blue curve.

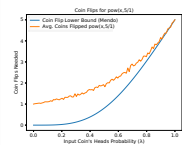
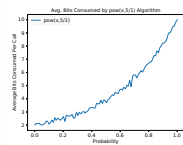
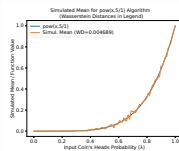
Figure 1 consists of three subplots. Subplot (a) shows the Simulated Mean for the penta-2H algorithm, comparing the proposed algorithm (blue line) and the penta-2H algorithm (orange line). The x-axis is 'Input Coin's Heads Probability (x)' from 0.0 to 1.0, and the y-axis is 'Simulated Mean (1/2^n) (y)' from 0.0 to 1.0. Both lines are nearly identical, following a curve that starts at (0,0) and ends at (1,1). Subplot (b) shows the Average Bits Consumed by the penta-2H algorithm versus Probability. The x-axis is 'Probability' from 0.0 to 1.0, and the y-axis is 'Average Bits Consumed Per CF' from 0 to 2000. The curve starts at approximately 1800 bits for probability 0.0 and drops sharply to near zero for probability 0.1, remaining low thereafter. Subplot (c) shows the Average Bits Consumed by the penta-2H algorithm versus Input Coin's Heads Probability. The x-axis is 'Input Coin's Heads Probability (x)' from 0.0 to 1.0, and the y-axis is 'Avg. Bits Consumed' from 0 to 800. The curve starts at approximately 800 bits for probability 0.0 and drops sharply to near zero for probability 0.1, remaining low thereafter.

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

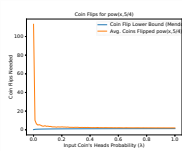
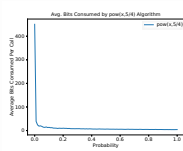
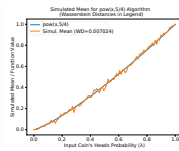
- (a) Sensitivity to input data:** The plot shows 'Sensitivity Rate (Fuzziness rate)' on the y-axis (0.0 to 1.0) versus 'Input Data's Fuzziness Probability (%)' on the x-axis (0.0 to 1.0). Two lines are shown: 'Proposed Mean (0.0000000000)' (blue) and 'Default Mean (0.0000000000)' (orange). Both lines are nearly identical, starting at (0,0) and ending at (1,1).
- (b) Average bits consumed by power, 3-bit algorithm:** The plot shows 'Average Bits Consumed (bits)' on the y-axis (0 to 80) versus 'Probability' on the x-axis (0.0 to 1.0). The 'power, 3-bit' line (blue) starts at approximately 75 bits for probability 0.0 and drops sharply to near 0 bits for probability 0.2, remaining low thereafter.
- (c) Cost of the proposed algorithm:** The plot shows 'Cost (bits/element)' on the y-axis (0 to 1.4) versus 'Input Data's Fuzziness Probability (%)' on the x-axis (0.0 to 1.0). Two lines are shown: 'Cost Flipped power, 3-bit' (orange) and 'Cost Flipped power, 3-bit' (blue). Both lines start at approximately 1.35 bits/element for probability 0.0 and drop sharply to near 0 bits/element for probability 0.2, remaining low thereafter.

[illegible]

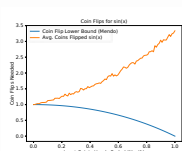
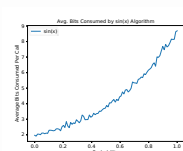
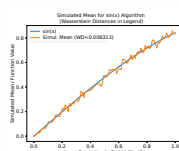
$\text{pow}(x, 5/1)$



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



$\sin(x)$



$\text{sqrt}(x)$

