On Stochastic Rounding with Few Random Bits

Andrew Fitzgibbon, Stephen Felix
Graphcore

ARITH 2025

Context

Memory, time, and power requirements of AI models mean moving to narrower and narrower (8, 6, or even 4-bit) floating point formats.

Stochastic rounding improves robustness of narrow-precision computations.

But… adds the cost of supplying random bits (rbits).

Recent work looks at supplying rbits more cheaply, or using fewer rbits.

This paper: quantity, not quality

Recent work looks at SR with few random bits, but optimal number of bits dependent on many factors

- Accumulator precision (precision of values before rounding, e.g. p=11 for Binary16, p=8 for BFloat16, other for specific h/w implementations
- Cost of random bits
- Accuracy requirements

Recent work [1] also looks at bias in SR with few random bits

[1]: El Arar, Fasi, Filip, Mikaitis. "Probabilistic error analysis of limited-precision stochastic rounding", 2024

Defining SR

Real value X is to be rounded to floating point value set

$$\{m \times 2^e \ \mid m \in \mathbb{M}\}$$

Where range of integer significands $\mathbb{M} = \{2^p \leq m < 2^{p+1}, m \in \mathbb{N}\}$

Loosely, ignoring subnormals etc, we obtain real-valued significand using

$$E = \lfloor \log_2 X \rfloor$$

$$S = |X| \times 2^{-E + P - 1}$$

and round to integer $\lfloor S \rfloor$ or $\lfloor S \rfloor + 1$

Defining SR

Loosely, ignoring subnormals etc, we obtain real-valued significand using

$$E = \lfloor \log_2 X \rfloor$$

$$S = |X| \times 2^{-E + P - 1}$$

and round to integer $\lfloor S \rfloor$ or $\lfloor S \rfloor + 1$

Using:

High-school rounding: $\hat{S} = \lfloor S + 0.5 \rfloor$

Stochastic rounding: $\hat{S} = \lfloor S + R \rfloor$ where $R \sim \text{Uniform}(0,1)$

Viewed as bits

Given S as binary real

0011011.10110101011110

We add either 0.5

.10000000000000

High-school rounding: $\hat{S} = \lfloor S + 0.5 \rfloor$

Or random bits

Stochastic Rounding: $\hat{S} = \lfloor S + R \rfloor$ where $R \sim \text{Uniform}(0,1)$

.01011101000101

Or "few" random bits

.01100000000000 "Default" implementation – biased

.01110000000000 "Quick fix" implementation – still biased

Viewed as bits

Given S as binary real

0011011.10110101011110

Add a "few" random bits

.01100000000000 "Default" implementation – biased

.01110000000000 "Quick fix" implementation – still biased

Or first RTNE to the number of SR bits, and add the few bits

0011011.10110101011110

0011011.11000000000000

.01110000000000 "Corrected" implementation – unbiased

Our contribution

1. Identify bias in the low-r, low-p scenario ([1] did low-r only)

2. Show how to correct the bias in implementations
    1. Quick and reasonably accurate
    2. Slightly less quick and more accurate

3. With bias computations for all of the above

SR in practice

Concrete implementation in “gfloat” python package

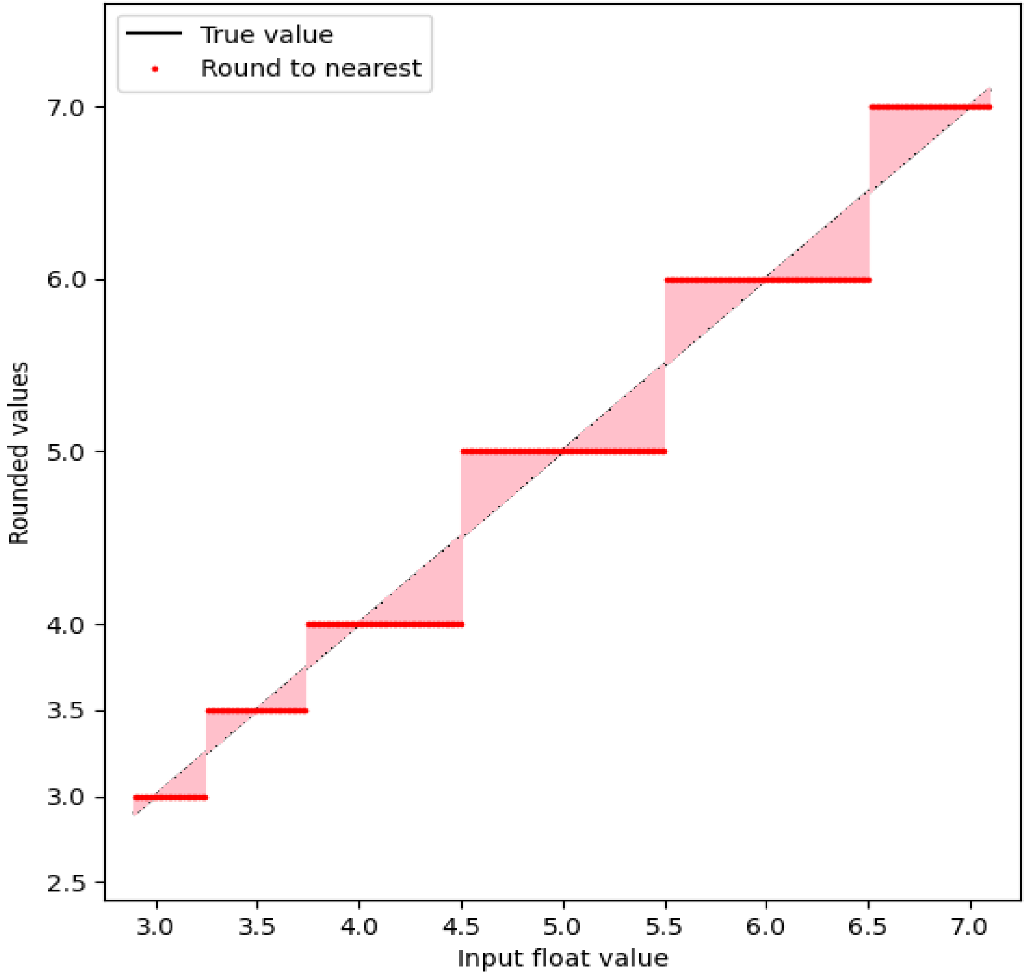
Normal “round” function takes input value (and target format) only

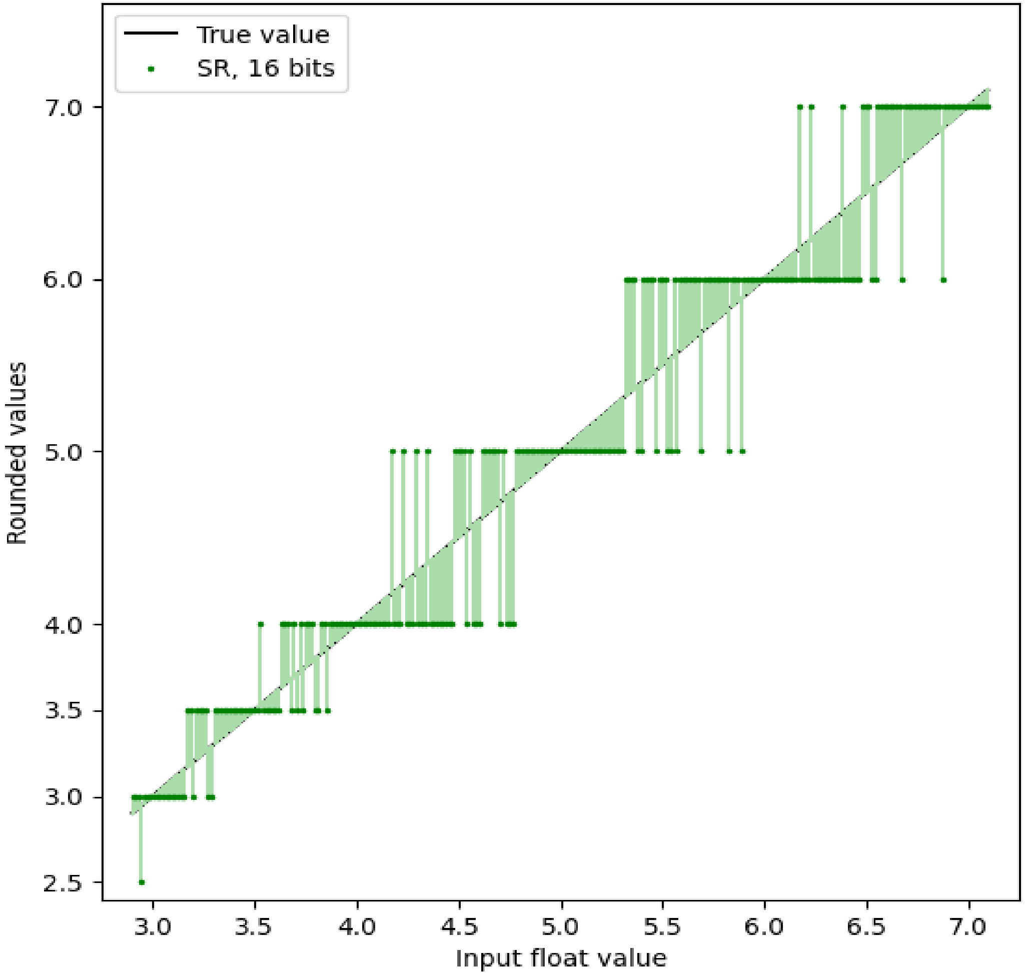
$$\mathrm{round}(S{:}\,\mathbb{R}) \;= \lfloor S + 0.5 \rfloor$$

Stochastic “round” function also takes random bits, an integer $0..2^{\#R}$.

$$\mathrm{round}(S{:}\,\mathbb{R}, R{:}\,\mathbb{N}) \;= \left\lfloor S + R \times 2^{-\#R} \right\rfloor$$

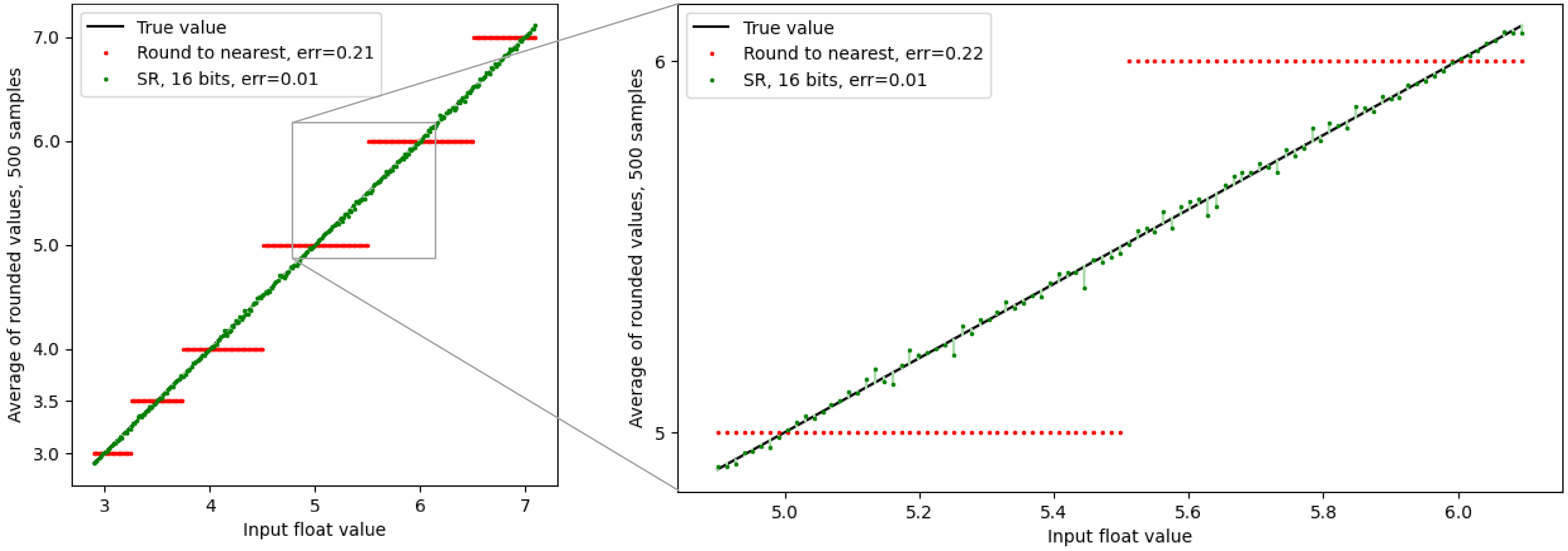
SR in practice



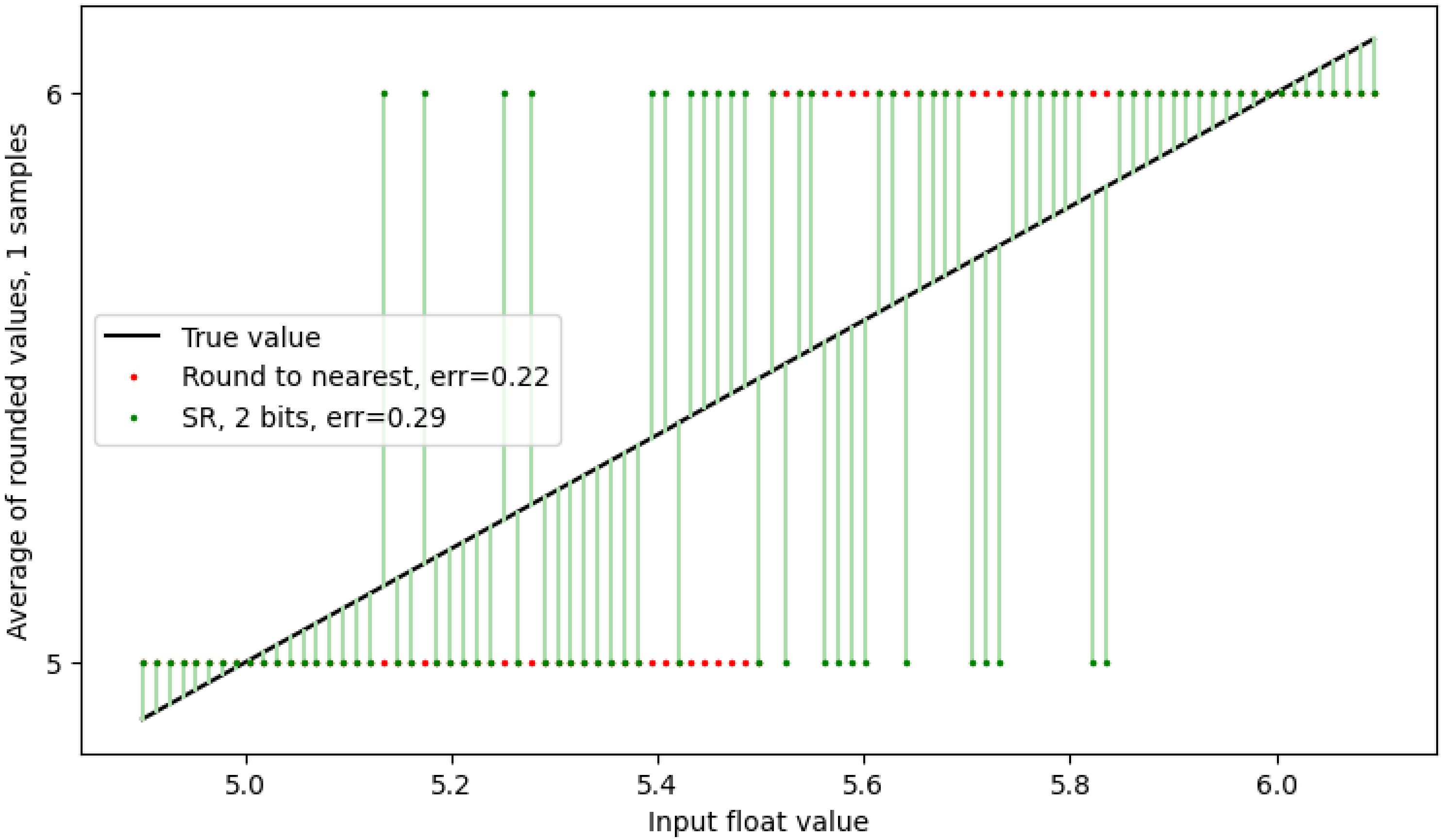


Rounding of $X \in$ "$\mathbb{R}$" to $Y \in \{3.0, 3.5, 4.0, 5.0, 6.0, 7.0\} \subset \mathbb{F}$ for a format $\mathbb{F}$

Average over a few runs...

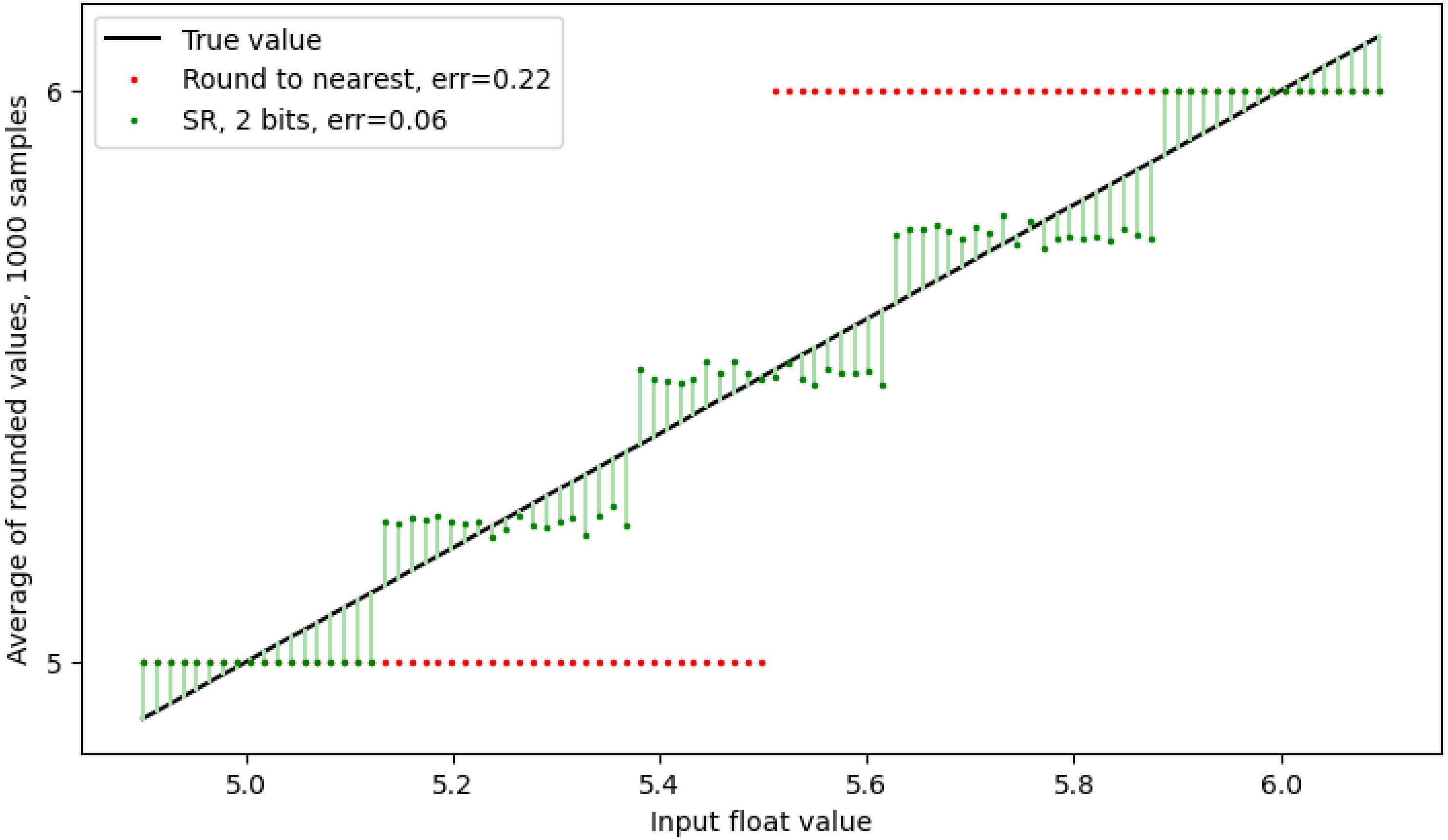


But that was 16 rbits on float64 inputs...



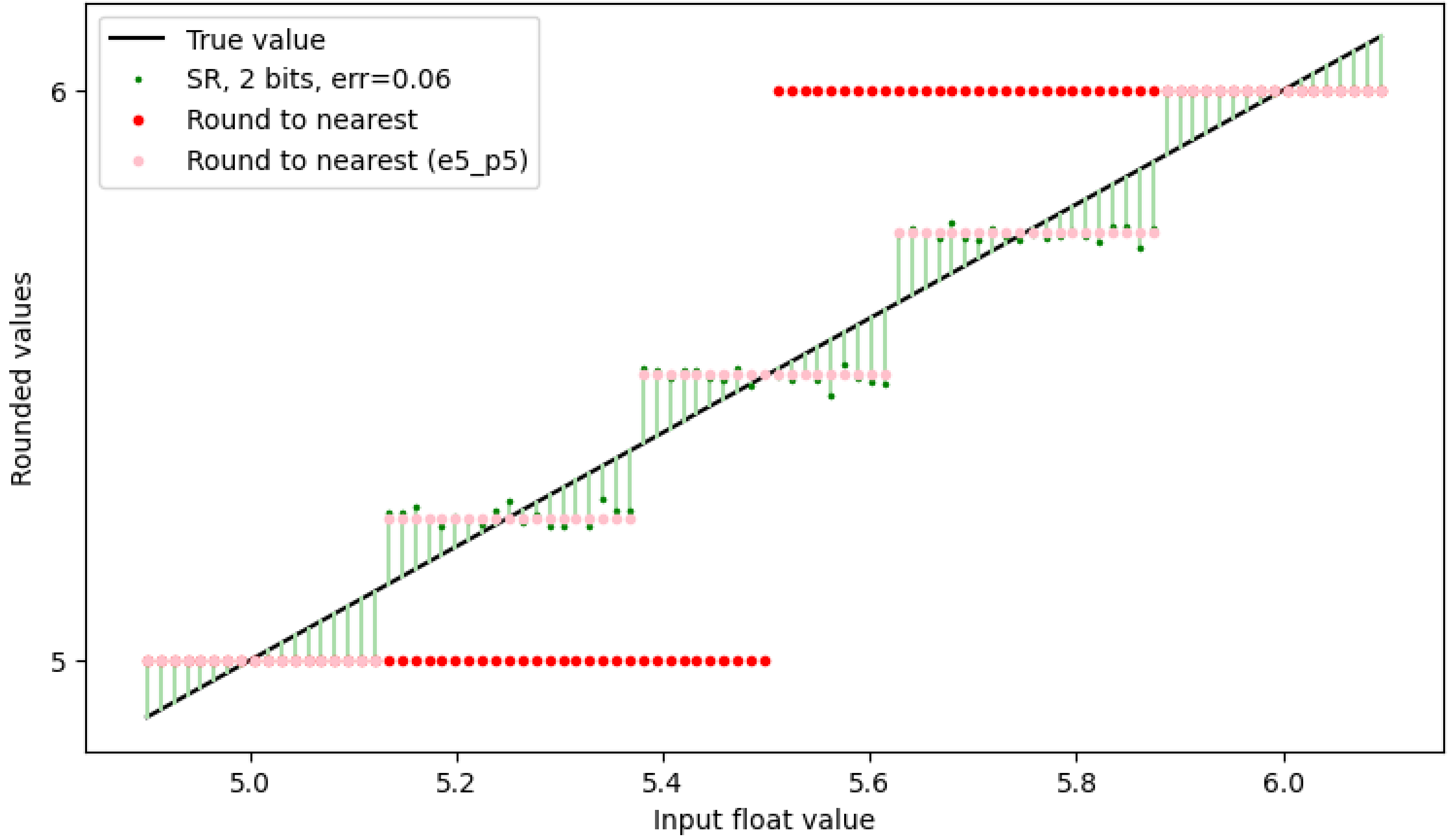
Rounding with 2 bits on float64 inputs

Few-bit SR, high-precision inputs



Averaged: looks like 2 rbits gives 2 precision bits on average

Few-bit SR, high-precision inputs



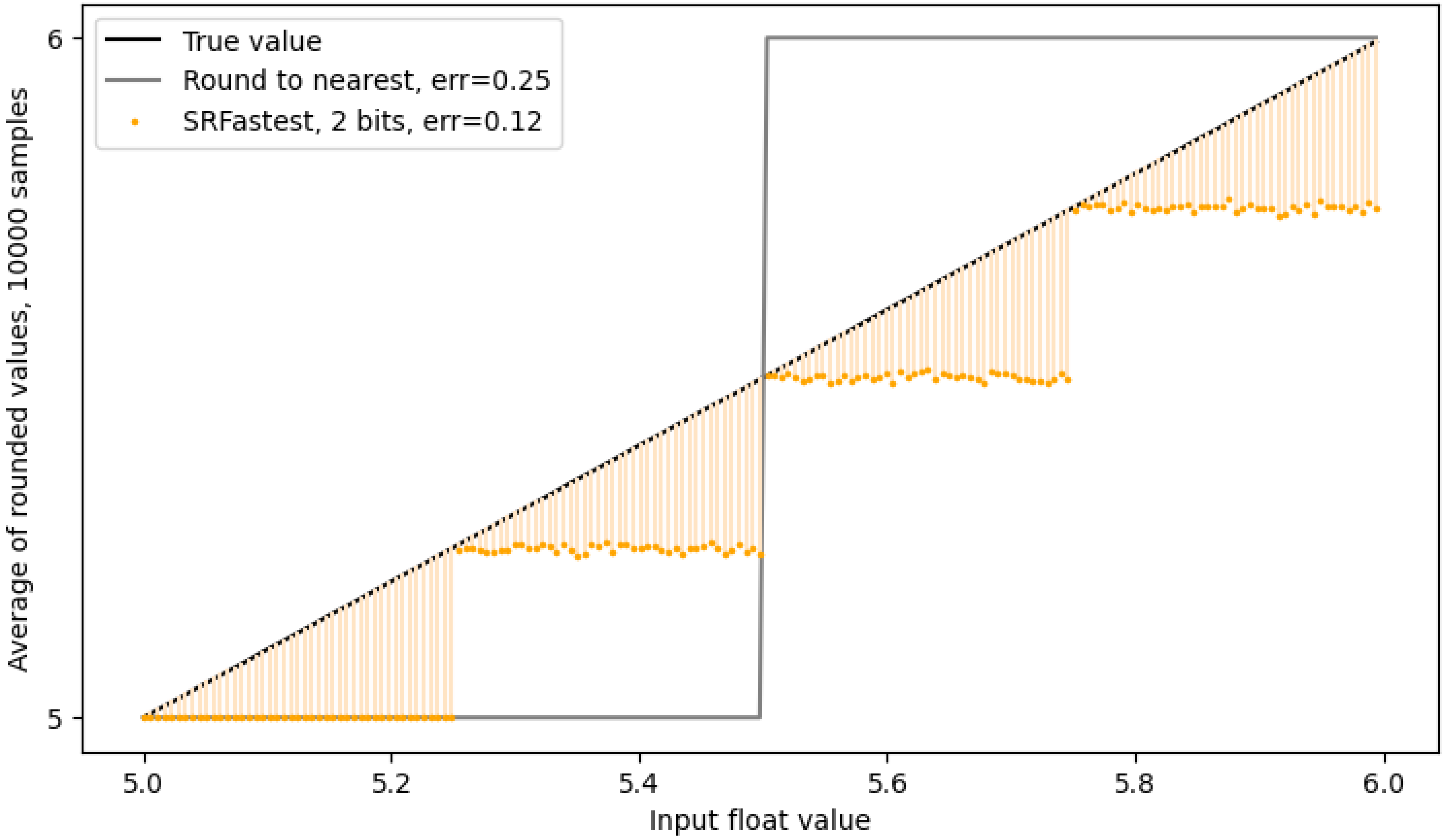
Averaged: looks like 2 rbits gives 2 precision bits on average
Yes, confirmed by RTNE to a format with 2 more precision bits

Few-bit SR, high-precision inputs

Averaged: looks like 2 rbits gives 2 precision bits on average
Yes, confirmed by RTNE to a format with 2 more precision bits

“High school” SR is biased

$$\text{round}(S\colon \mathbb{R}, R\colon \mathbb{N}) = \lfloor S + R \times 2^{-\#R} \rfloor$$



Aside: how do we measure bias?

Expected value of error

$$\text{Bias} = \mathbb{E}[X - \text{round}(X)\ ]$$

But what is the expectation over?

Need to pick a probability distribution $p(X)$ to write, more precisely:

$$\text{Bias} = \mathbb{E}_{X \sim p(X)}[X - \text{round}(X)\ ]$$

What probability distribution? It can't be uniform on $[-MAX, MAX]$, as then always rounding toward zero is "unbiased".

We really want a family of test distributions, one per float-pair.

Aside: A list of bias formulae (see paper)

Writing $N = \#R,$ i.e. using N bits of randomness

$$Bias_{SRFF} = -2^{-(N+1)}$$

If incoming values are finite-precision (generally true), with precision D, then

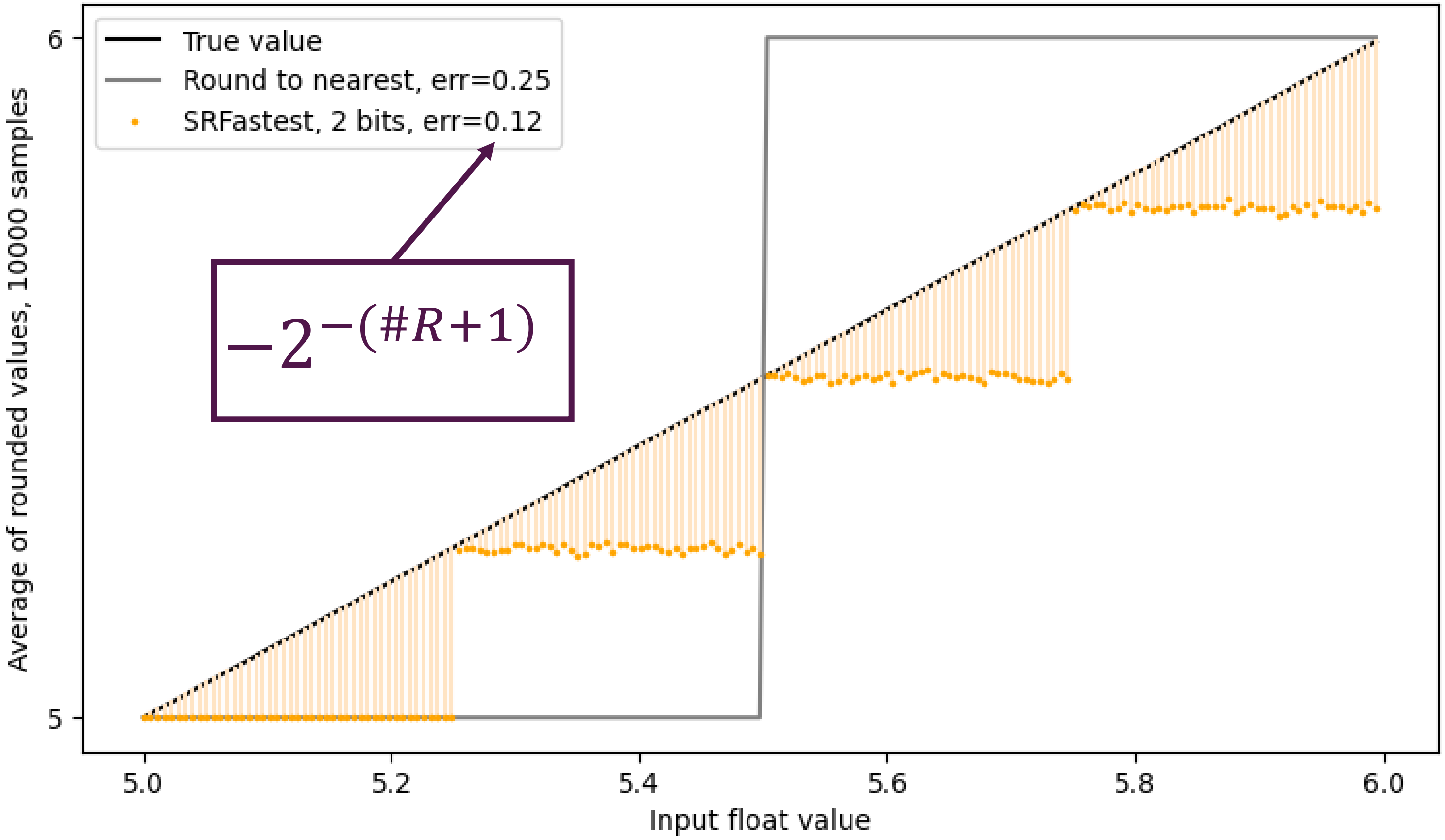
$$Bias_{SRFF,D} \leq 2^{-(D+1)} - 2^{-(N+1)}$$

Bound tight for $N < D$, note zero for $N = D$, i.e. number of rbits equal to difference in precisions.

This is the case in preceding work (the non-few-bit case). I.e. existing hardware is fine, as it uses a lot of rbits; any future hardware trying to save rbits will need to correct this bias.

“High school” SR is biased

$$\text{round}(S\colon \mathbb{R}, R\colon \mathbb{N}) = \lfloor S + R \times 2^{-\#R} \rfloor$$



Debugging: Rounding profiles

With 2 random bits, we can just treat SR as selecting from 4 deterministic schemes:

$$\text{round}(S, 0b00) = \lfloor S + 0.00 \rfloor = \lfloor S \rfloor, \quad \text{“Always floor”}$$

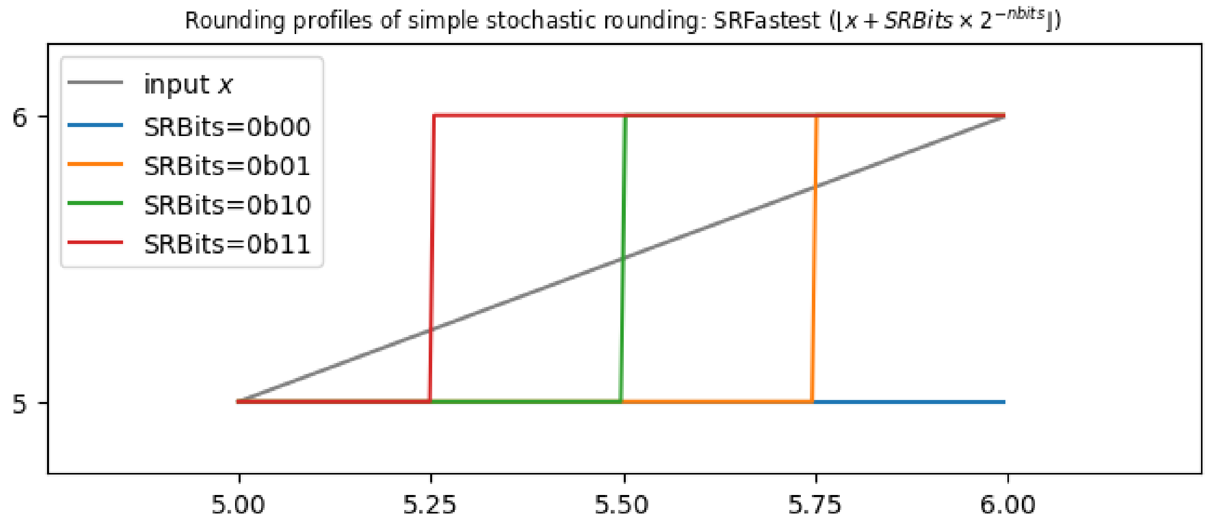
$$\text{round}(S, 0b01) = \lfloor S + 0.25 \rfloor \quad \text{“Ceil if } \delta \geq 3/4\text{”}$$

$$\text{round}(S, 0b10) = \lfloor S + 0.50 \rfloor \quad \text{“Ceil if } \delta \geq 2/4\text{”}$$

$$\text{round}(S, 0b11) = \lfloor S + 0.75 \rfloor \quad \text{“Ceil if } \delta \geq 1/4\text{”}$$

We can plot these…

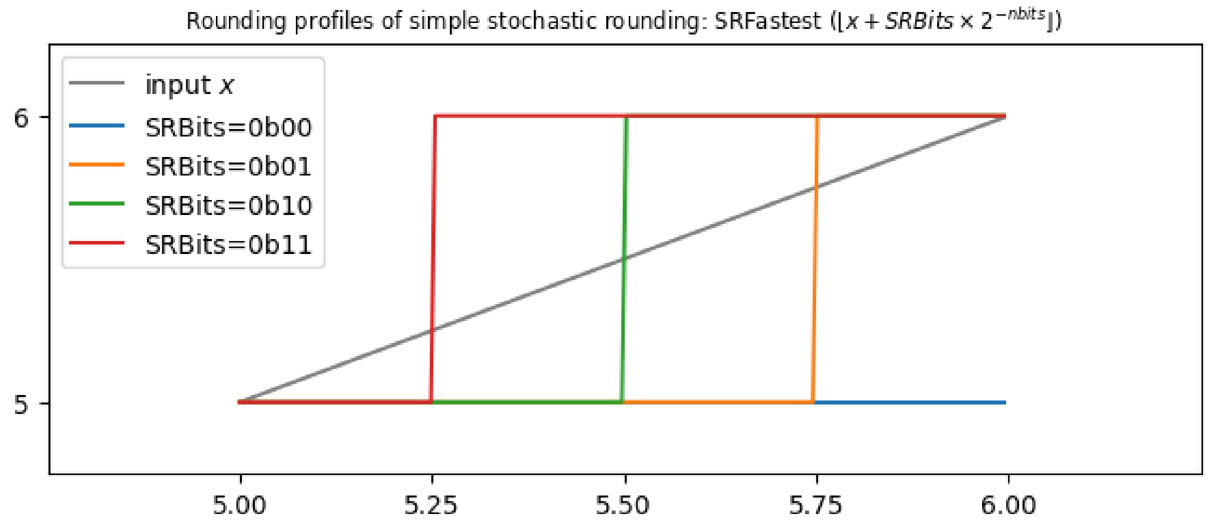
… and see the asymmetry



Quick fix:

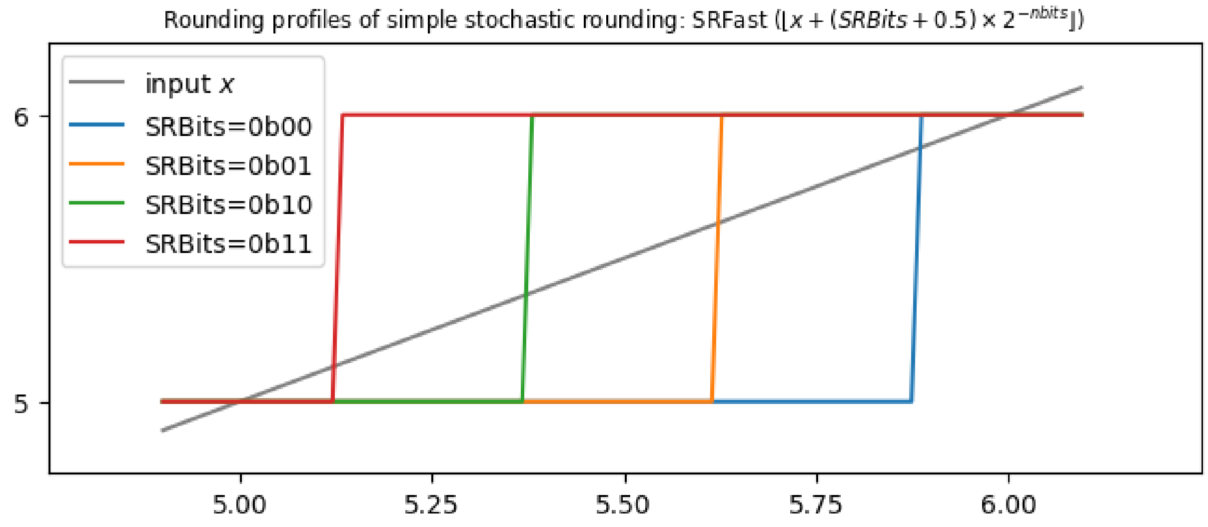
Move from

$$\text{round}(S,R) = \lfloor S + R \times 2^{-\#R} \rfloor$$



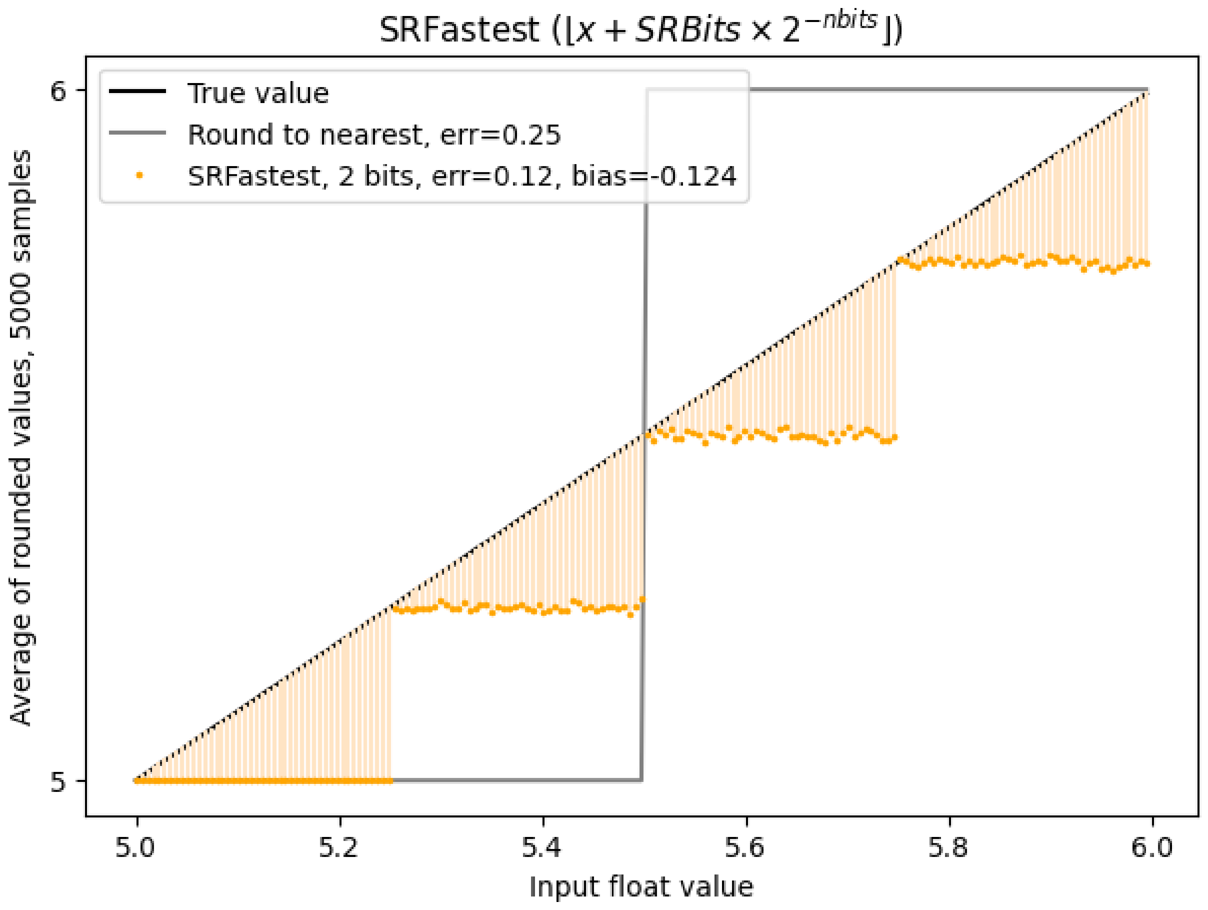
To

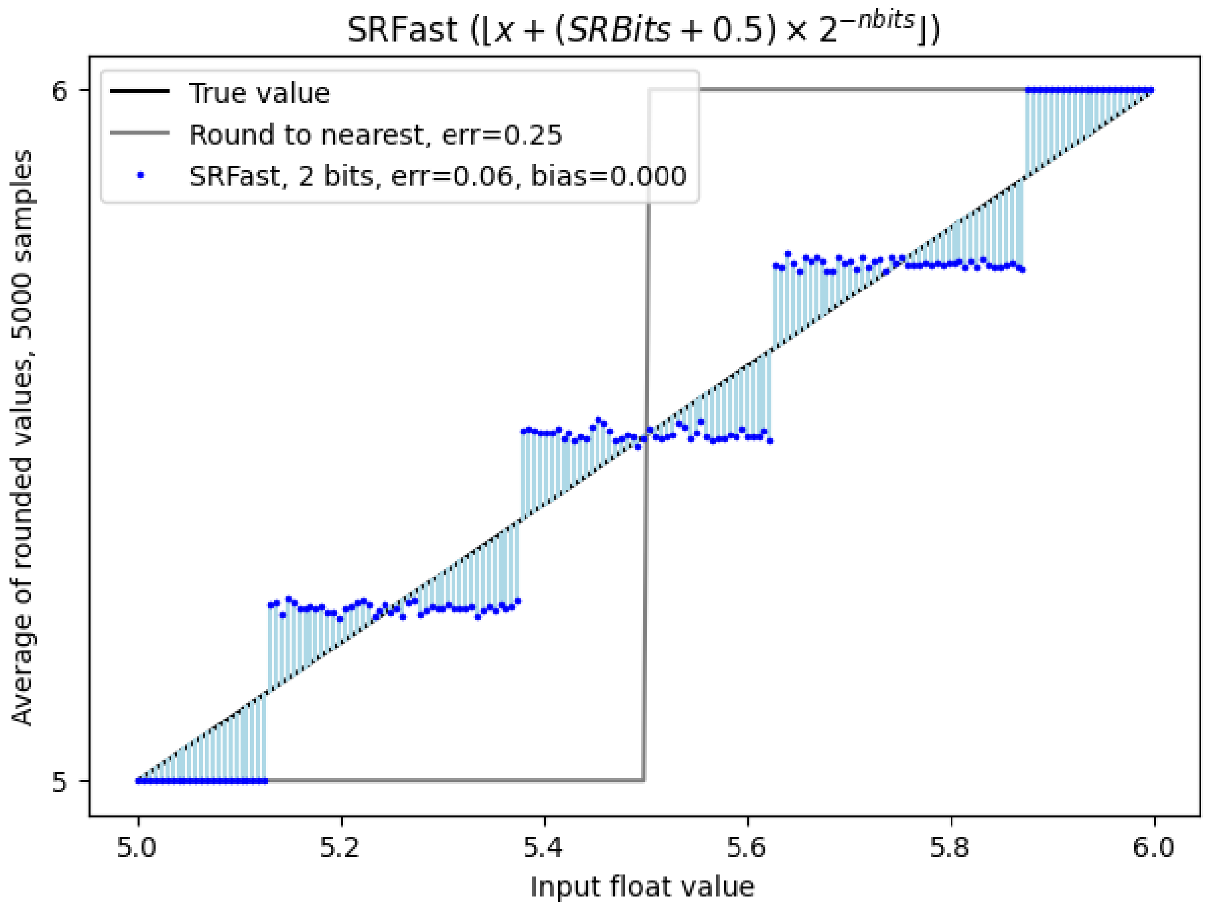
$$\text{round}(S,R) = \left\lfloor S + (R + \tfrac{1}{2}) \times 2^{-\#R} \right\rfloor$$



Maybe call it "jam an extra 1-bit"

Fixed. So are we done yet?



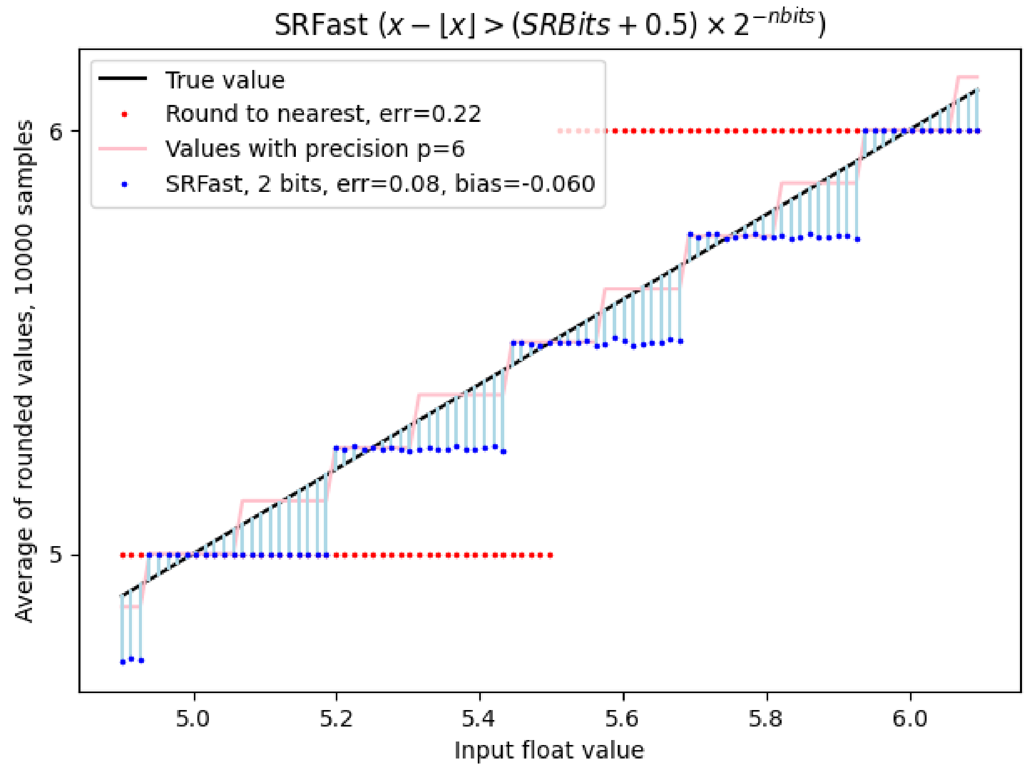


Finite-precision inputs

With limited precision inputs

Bias returns if input precision before SR is close to target precision

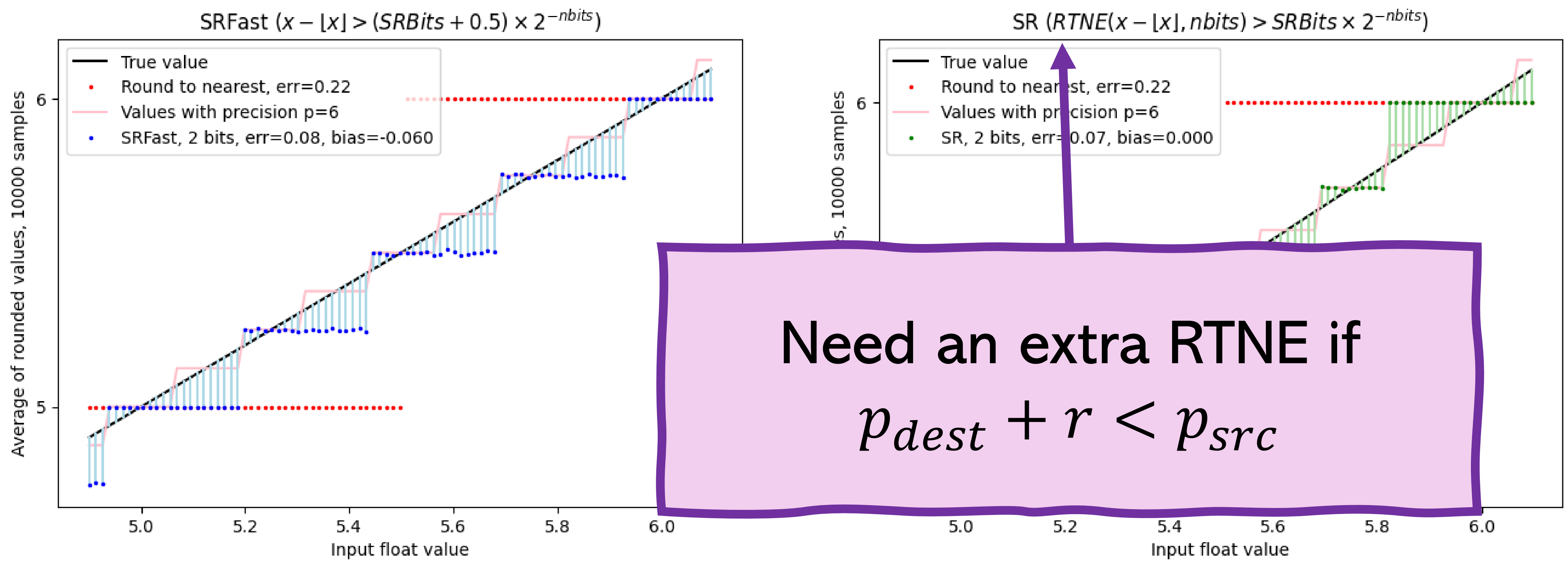
(e.g. bfloat16, p=8 to binary8p4, precision difference is just 4)



With limited precision inputs

Bias returns if input precision before SR is close to target precision

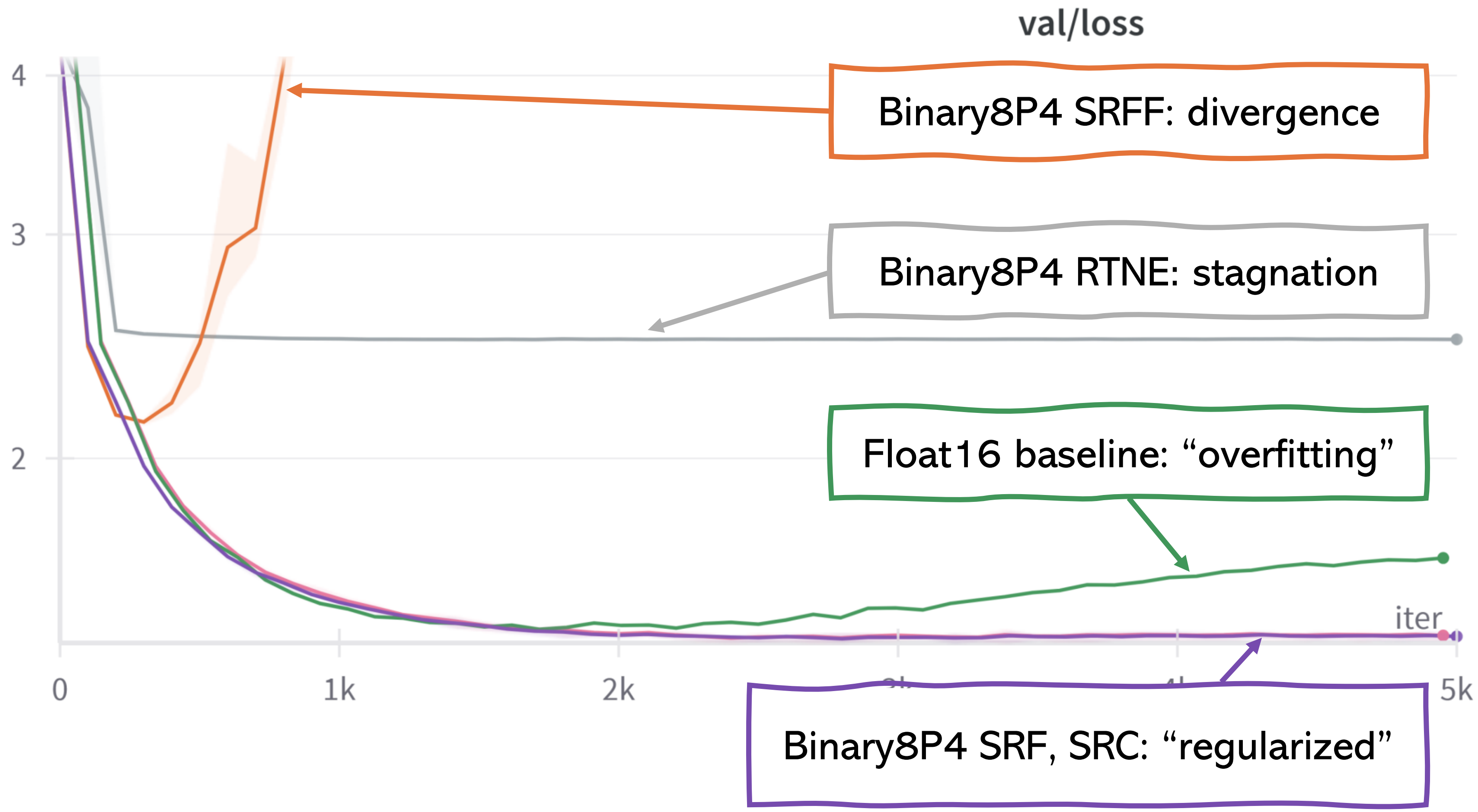
(e.g. bfloat16, p=8 to binary8p4, precision difference is just 4)



Need an extra RTNE if

$$p_{dest} + r < p_{src}$$

Importance in practice to be explored...



Small transformer (10m params)

Training an 8 bit model with 16-bit gradients

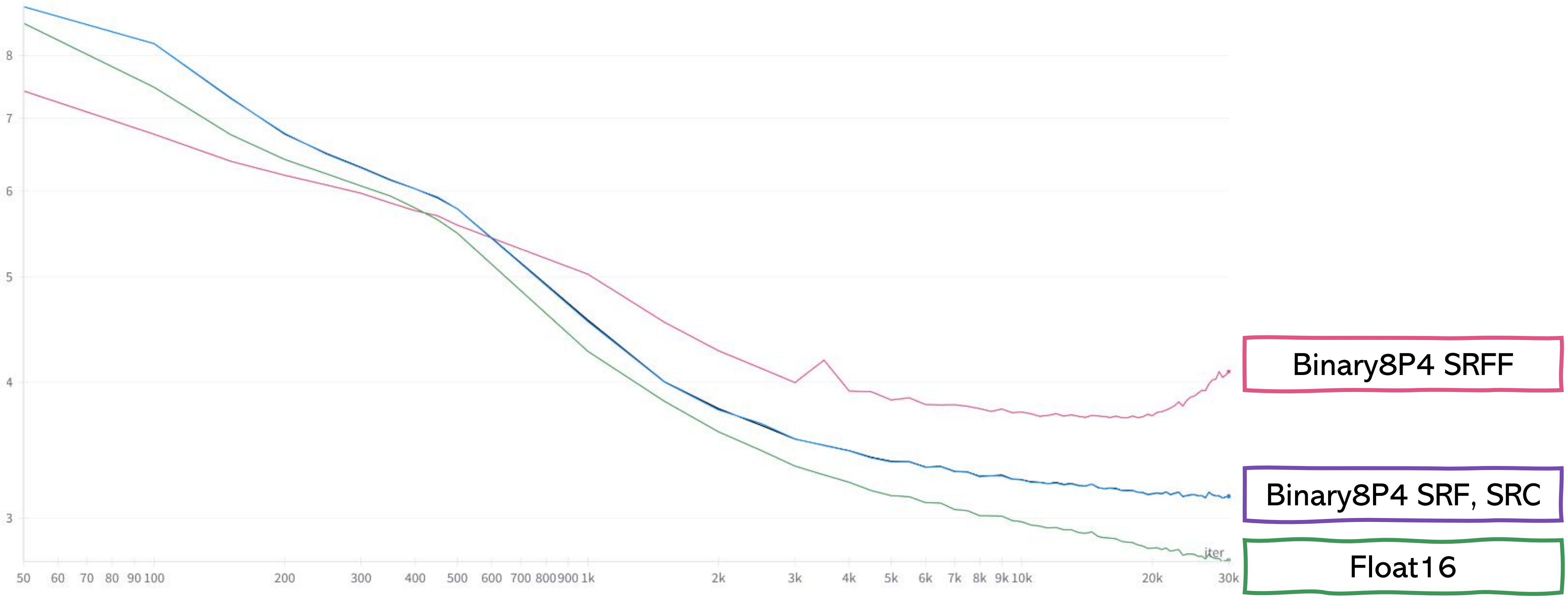
Master weights in Binary8P4

$W_8 : F8 = initialize$
while ...
  $W_{16} = \mathrm{To16}(W_8)$
  $g : F16 = \nabla loss(W_{16})$
  $u : F16 =$ Adam update in $F16$
  $W_8 = \mathrm{RoundTo8}(W_{16} + u)$

[*] *Do not over index on "overfitting" vs "regularized" – this only applies to small models.*

Medium scale

Comparisons are more tricky – some suggestion that learning rate should be higher for SR



Conclusions

Few-bit SR can be effective, and as more FLOPs are issued per cycle, more SR bits are needed per cycle.

A simple implementation of bias correction can perform as well as the “optimal” implementation.

Experiments continue… take a look at

https://github.com/graphcore-research/arith25-stochastic-rounding