Low-Precision Arithmetics and Stochastic Rounding

Mantas Mikaitis, University of Manchester

Contact: mantas.mikaitis@manchester.ac.uk

Predictability of Weather and Climate group talk @ Univ. of Oxford, 13 July 2020

Errors in computing and the role of rounding

- Every computer simulation contains errors.
  1. Errors in data collection from a system,
  2. errors in mathematical models of the system,
  3. errors in numerical approximations of the models, and
  4. errors in data and arithmetic (rounding).
- Errors in class 4 are usually believed to be lowest.
- For many years reasonable assumption due to 32/64-bit arith.
- Recently 8/16-bit HW also appears next to 32/64-bit.
- Mixed-precision algorithms with error class 4 more significant.
- Next: 8/16-bit HW without 32/64-bit?
- Class 4 error will probably require reducing.
- Various ways to address at HW and algorithm level.

IEEE 754-2019 floating-point standard

Property	binary32	binary16
p (bits in the significand)	24	11
ε	2^{-23}	2^{-10}
$e_{\max}$	127	15
$e_{\min}$	-126	-14
Min. positive normal	2^{-126}	2^{-14}
Min. positive subnormal	$2^{-126} \times 2^{-23}$	$2^{-14} \times 2^{-10}$

Rounding modes:

- RN – Round to nearest, even on ties
- RZ – Round towards zero
- RU – Round (up) towards $+\infty$
- RD – Round (down) towards $-\infty$

Also, non-standard bfloat16 ($p = 8$).

Novel hardware with reduced precision

- 16-bit floating-point in HW from NVIDIA, ARM, Intel, Google.
- 129 machines in the June TOP500 list contain NVIDIA devices with binary16 arithmetic.
- Mainly Matrix Multiply-Accumulate (MMA), but also general purpose arithmetic.
- List of recent hardware with low-prec. MMA:

Device	GEMM dimensions	input	accumulation
('16) Google TPU v2	$128 \times 128 \times 128$	bfloat16	binary32
('17) Google TPU v3	$128 \times 128 \times 128$	bfloat16	binary32
('17) NVIDIA V100	$4 \times 4 \times 4$	binary16	binary32
('18) NVIDIA T4	$4 \times 4 \times 4$	binary16	binary32
('19) ARMv8.6-A microarch.	$2 \times 4 \times 2$	bfloat16	binary32
('20) NVIDIA A100	$8 \times 4 \times 8$	bfloat16	binary32
	$8 \times 4 \times 8$	binary16	binary32
	$4 \times 2 \times 2$	binary64	binary64
	$4 \times 4 \times 8$	TensorFloat-32	binary64

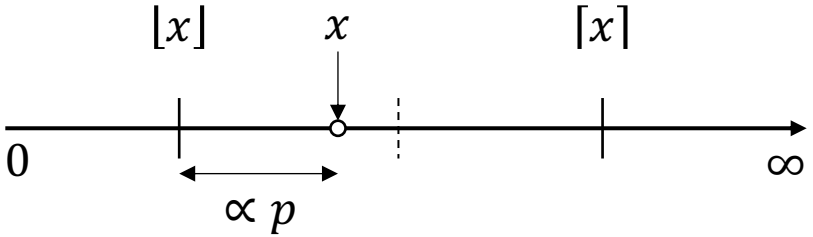
Stochastic rounding (SR)

Definition after Connolly, Higham, and Mary [1].

Given $x \in \mathbb{R}$ with $\lfloor x \rfloor \leq x \leq \lceil x \rceil$ (when $x \notin F$ it is between the two neighbouring floats), stochastic rounding (SR) is defined as

$$\mathrm{SR}(x) = \begin{cases} \lceil x \rceil & \text{with the probability } p, \\ \lfloor x \rfloor & \text{with the probability } 1 - p. \end{cases}$$

Mode 1: $p = 0.5$; Mode 2: $p = \frac{x - \lfloor x \rfloor}{\mathrm{ulp}(x)}$.



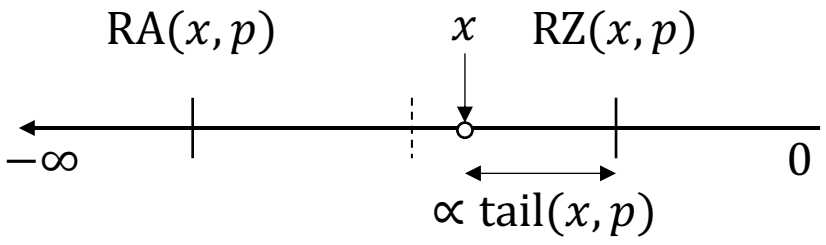
With mode 2, $\mathbb{E}\big(\mathrm{SR}(x)\big) = x$.

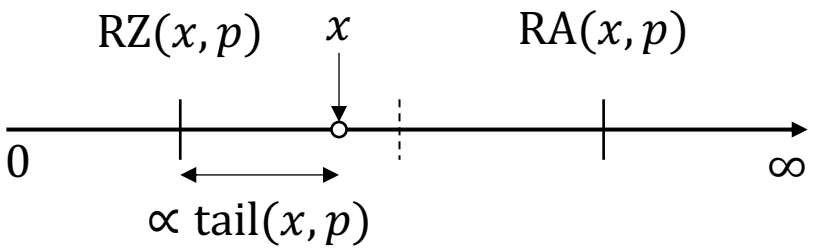
Bit-level definition of SR in FP

Given a real number x, a random number $P \in [0,1)$ from a uniform distribution and the target precision p,

$$\mathrm{SR}(x,p) = \begin{cases} \mathrm{RA}(x,p) & \text{if } P < \mathrm{tail}(x,p), \\ \mathrm{RZ}(x,p) & \text{if } P \geq \mathrm{tail}(x,p), \end{cases}$$

where $\mathbf{tail}(x,p) \in [0,1)$ is a value encoded by the trailing bits that do not fit into precision p (a **residual**), RZ – round towards zero, RA – round away from zero.





Early papers on SR

- Forsythe, 1950 [2], 1959 [3] proposed SR mode 2 to make rounding errors independent random values; allows using probabilistic error bounds.

"*Tests with I. B. M. equipment indicate that random round-off probably eliminates a priori the peculiarities of round-off found by Huskey on the ENIAC*" [2].

- Hull and Swensen, 1966 [4] use SR mode 2 to test whether random errors behave in the same fashion as ordinary errors.
- Vignes, 1993 [4], Jézéquel and Chesneaux, 2008 [5] use SR mode 1 for estimating round-off error propagation.
- SR mode 2 (round to integer) appears in Physics lit. as well, for example in 1986 [6] and 1989 [7].

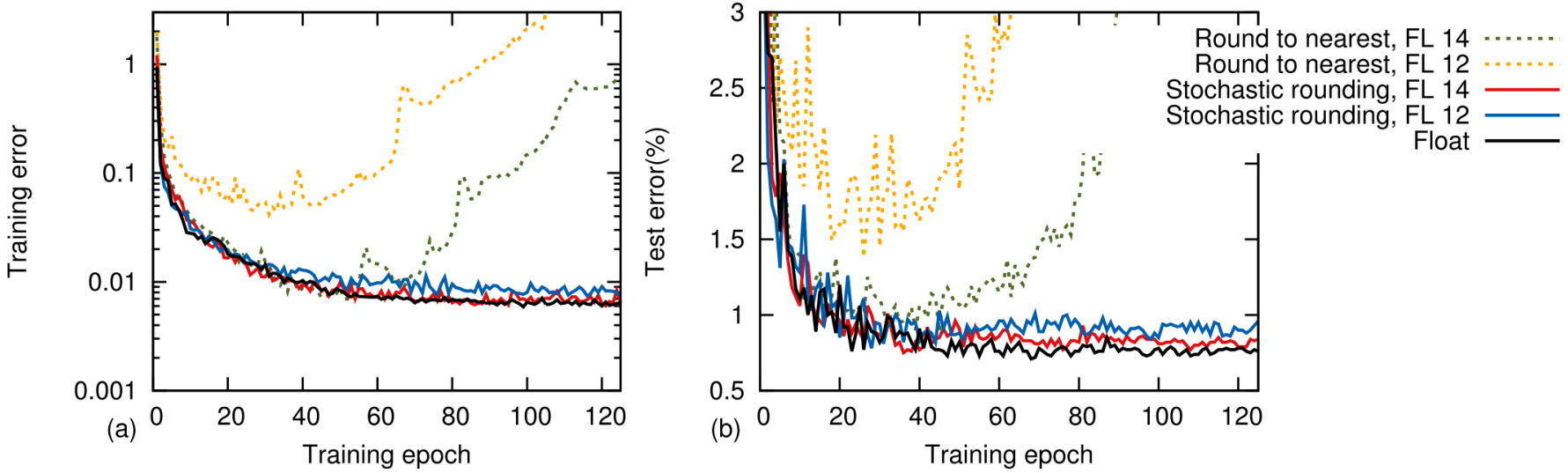
Monte Carlo arithmetic

- Parker, 1997 [8] proposes Monte Carlo(MC) arithmetic with both SR modes. Tool for empirical analysis of round-off errors.
- Févote and Lathuilière, 2016 [9] present industrial software based on MC arith. with both SR modes.
- Denis et al. 2018 [10] present LLVN compiler extension for performing numerical analysis automatically using MC arith.

None of the previously discussed works propose to replace RN mode with SR for reducing errors at runtime.

SR in machine learning

- M. Höhfeld and S. E. Fahlman, 1992 [11] and Gupta et al, 2015 [12] propose SR mode 2 to train neural networks.
- SR allowed to reduce hardware to 6 bits in [11].
- SR allowed [12] to train 16-bit fixed-point neural networks almost as accurately as binary32.
- SR applied on fixed-point inner products in [12].



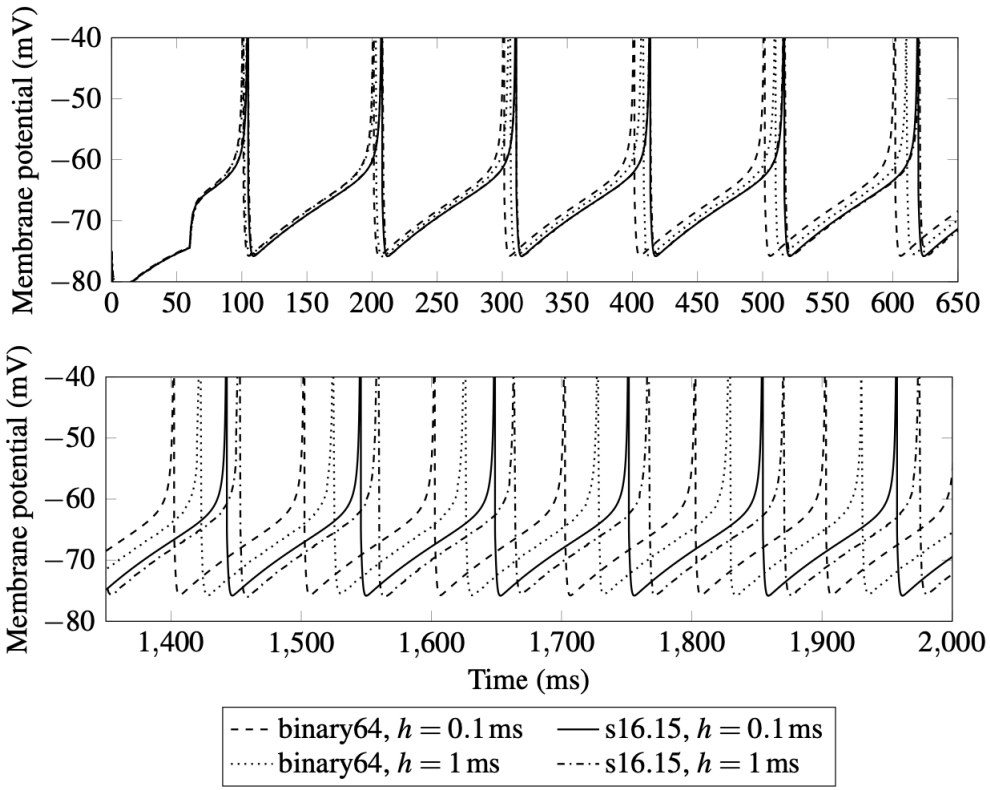
Ordinary Differential Equations (ODE) in fixed-point arithmetic

- In [13] we applied SR when solving neuron ODEs in fixed-point.
- Spike time lag/lead.
- Cause: round-off errors.
- 32-bit fixed-point arith.
- Goal: get closer to float.
- Reproducibility.

$$\frac{dV}{dt} = 0.04V^2 + 5V + 140 - U + I(t),$$

$$\frac{dU}{dt} = a(bV - U),$$

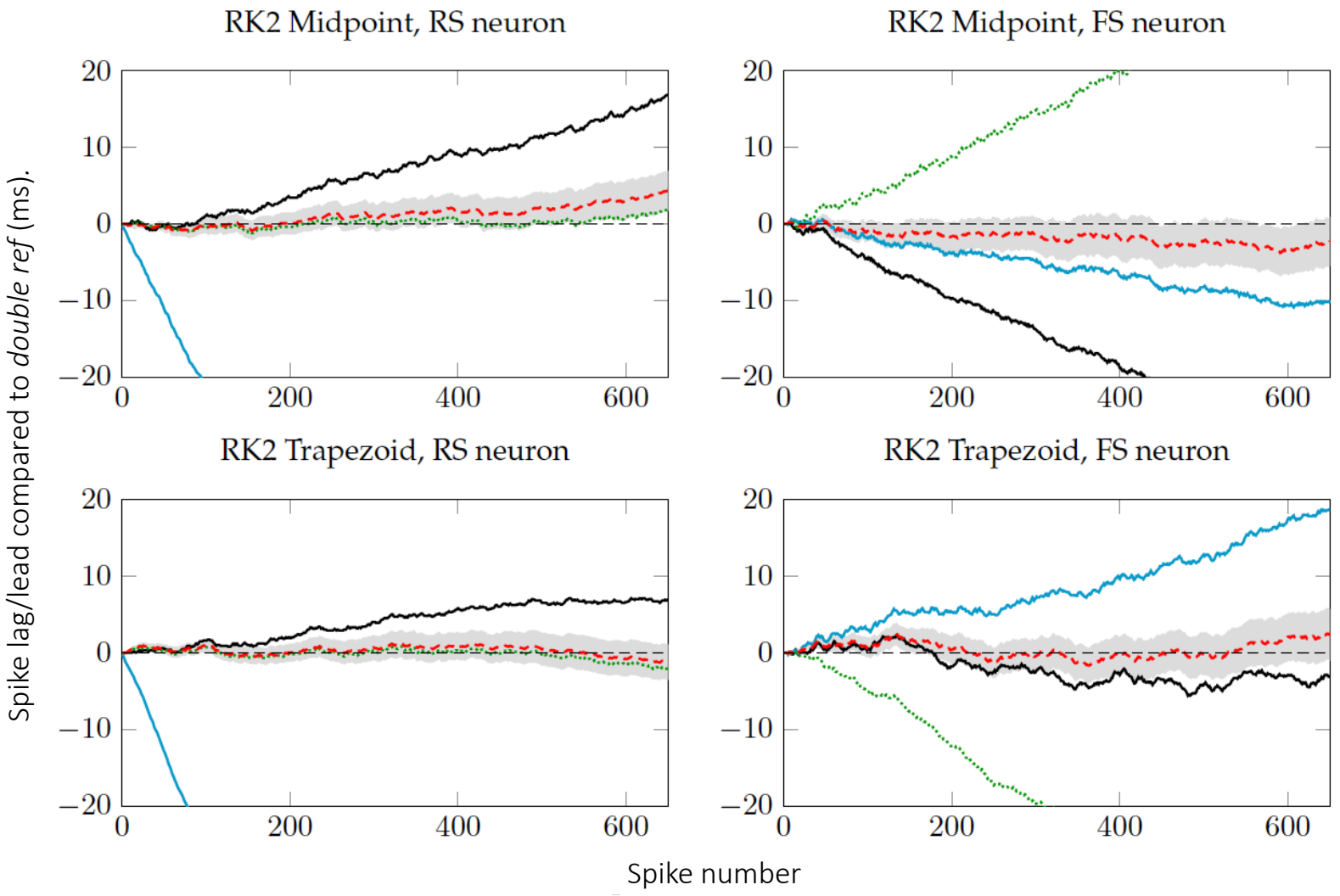
if $V \geq 30$ mV, $V = c$, $U = U + d$.



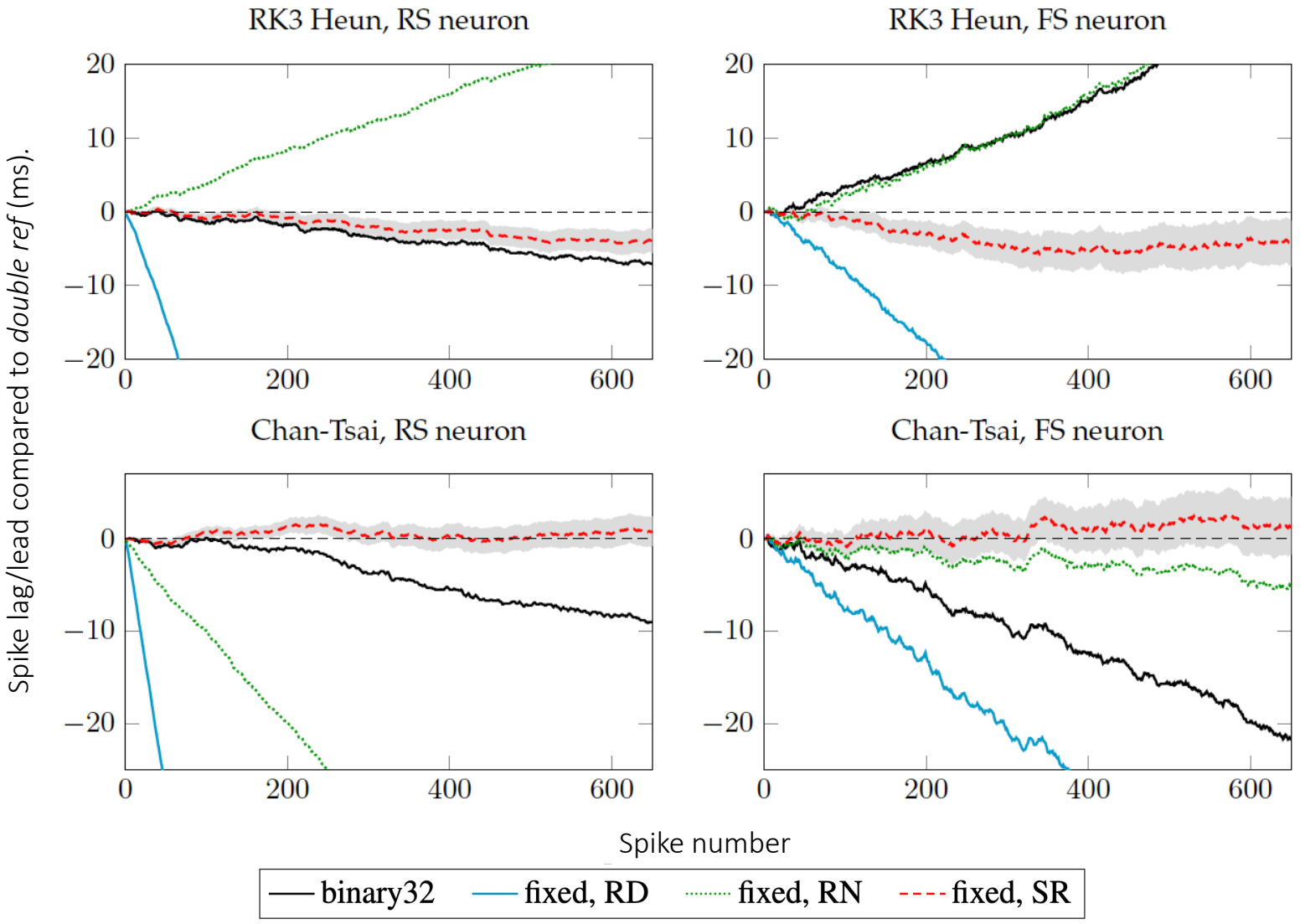
Testing method of Izhikevich ODE solutions

- Four ODE solvers: RK2 Midpoint, RK2 Trapezoid, RK3 Heun, Chan-Tsai.
- Two different neuron types (regular and fast spiking – RS/FS).
- Five arithmetics: IEEE 754 binary64 (reference), binary32, fixed-point {round-down, round-to-nearest, stochastic}.
- Run for ~1min to produce around 600 spikes.
- For SR, repeat experiment 100 times with different PRNG seeds.
- Report mean and std. dev. spike lag of each spike.

Spike lags: 2nd order solvers ($h = 0.1\ ms$)



Spike lags: 3rd order solvers ($h = 0.1\ ms$)



Psuedo-random number generation

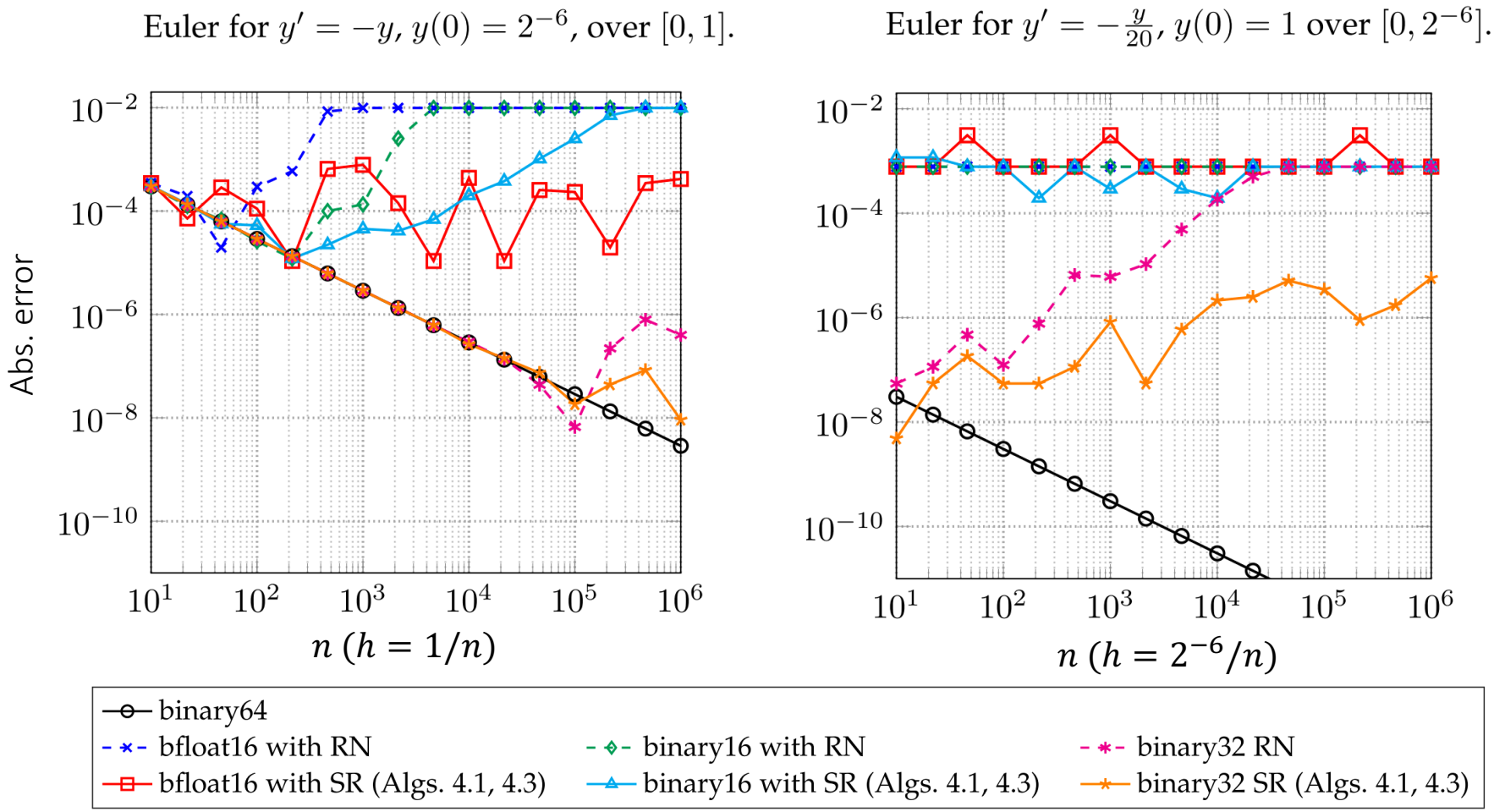
- Set of G. Marsaglia's KISS generators (expensive on ARM but pass randomness tests) [15].
- Linear congruential generator (modular arithmetic by overflow of 32-bit registers):

```
static inline uint32_t cheap_generator () {
    static unsigned long i;
    idum = 1664525L*i + 1013904223L;

    return i;
}
```

- For our tests, no significant difference (mean & std. dev.)!

ODEs in floating-point arithmetics



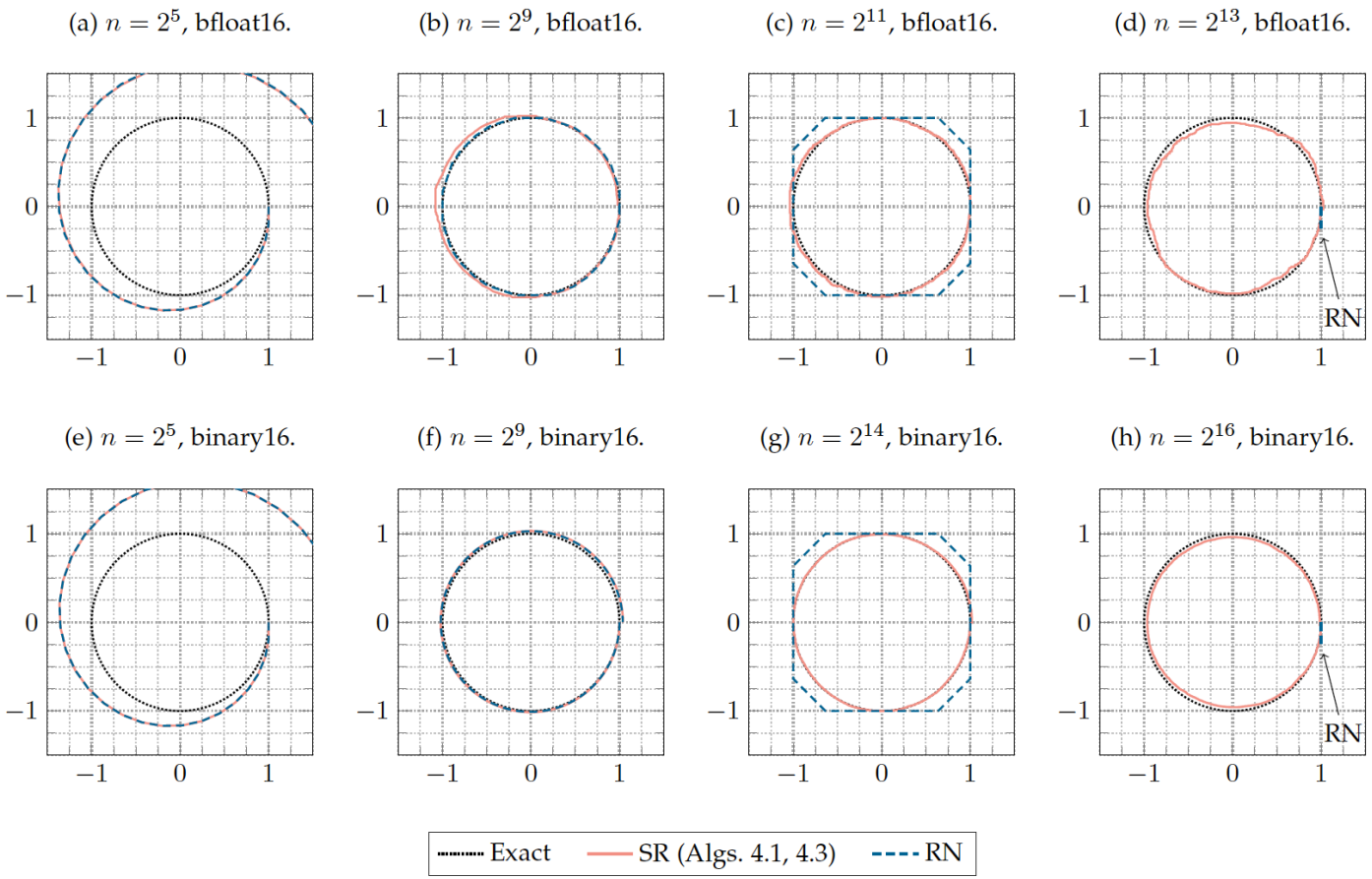
Work with M. Fasi [14] (Preprint).

ODEs in floating-point arithmetics

Use Euler's method to draw unit circle:

$$u_{k+1} = u_k + hv_k,$$
$$v_{k+1} = v_k - hu_k,$$

with $u_0 = 1, v_0 = 0$, $h = 2\pi/n$.



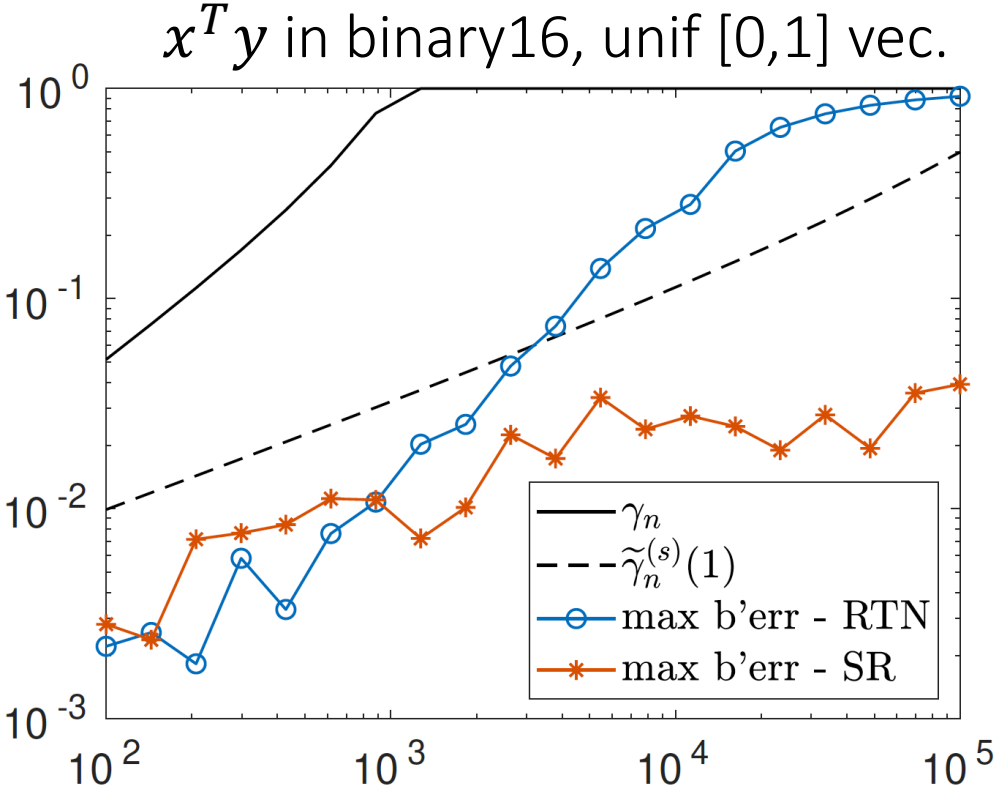
Work with M. Fasi [14] (Preprint).

Backward error results

- M. Connolly et al. 2020 [1] show that in the upper bound of backward error, term nu can be replaced by $\sqrt{n}u$.
- Show that the long standing *rule of thumb* is a *rule* for SR.
- The authors also show that with SR

$$\mathbb{E}\big(\mathrm{fl}(x^T y)\big) = x^T y.$$

- Proves various fl pt prop. that still hold and that do not hold anymore with SR.
- First analysis of SR.



Stagnation: core issue that SR addresses

- SR mainly addresses a problem of *stagnation.*
- In $\mathrm{fl}(a+b)$, if a or b is much smaller than the other, it will be rounded off to 0.
- Long summation and thus dot products have this problem if a basic recursive summation algorithm is used.
- SR alleviates this by occasionally rounding up small terms.
- Basic example assuming integer rounding:

$$\mathrm{RN}(0.25)+\mathrm{RN}(0.25)+\mathrm{RN}(0.25)+\mathrm{RN}(0.25)=0,$$

whereas

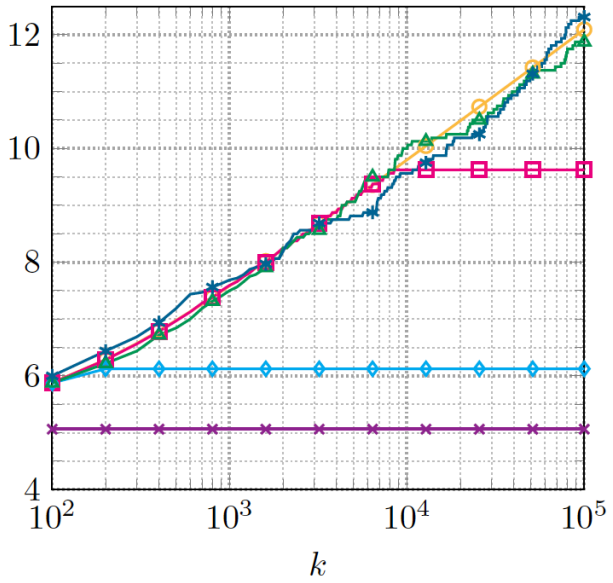
$$\mathrm{SR}(0.25)+\mathrm{SR}(0.25)+\mathrm{SR}(0.25)+\mathrm{SR}(0.25)=0,1,2,3,\text{or } 4\ (\text{most likely } 0 \text{ or } 1).$$

Summation examples

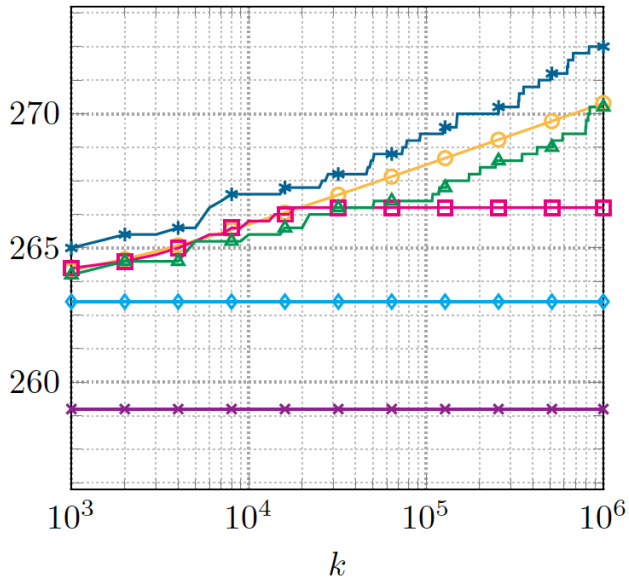
Consider truncated harmonic series

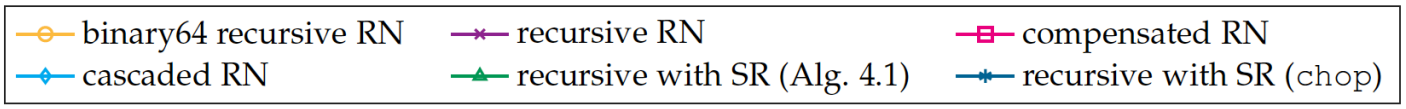
$$H_k(s_0) = s_0 + \sum_{i=1}^{k} \frac{1}{i} = s_0 + 1 + \frac{1}{2} + \frac{1}{3} \ldots + \frac{1}{k}.$$

(a) bfloat16, $s_0 = 0$.



(b) binary16, $s_0 = 256$.





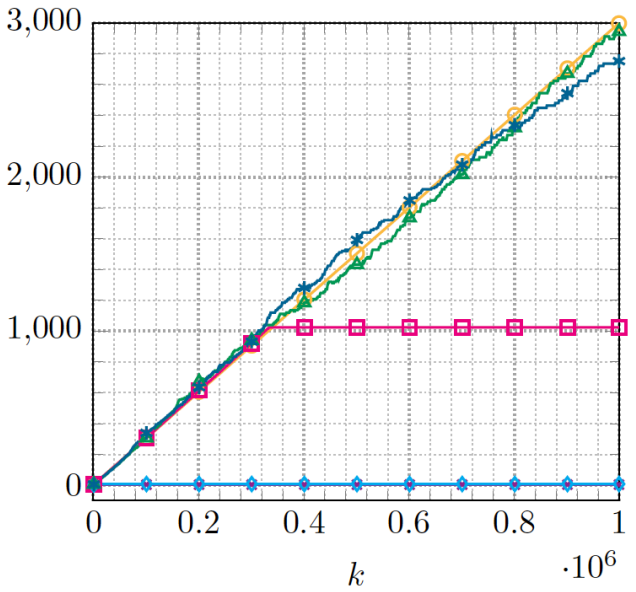
Work with M. Fasi [14] (Preprint).

Summation examples

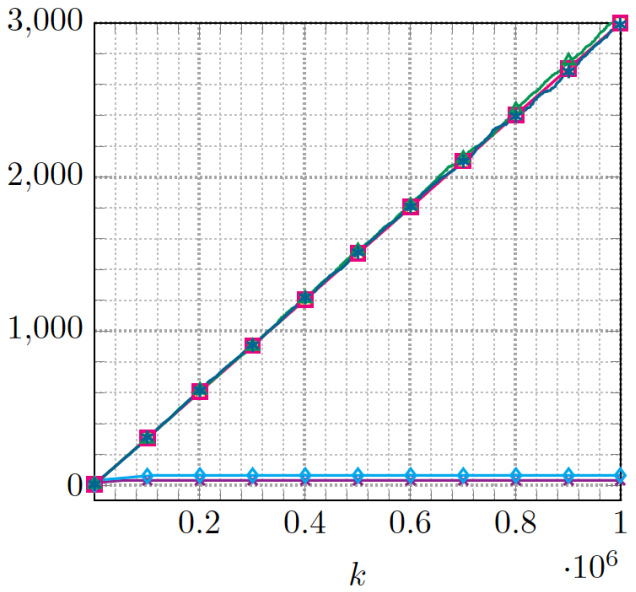
Consider summing random values

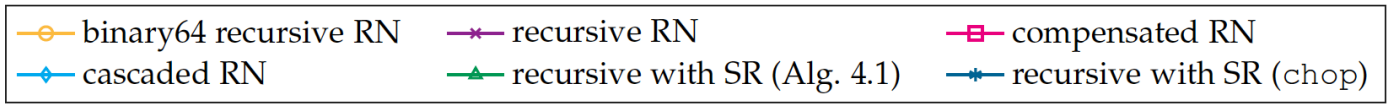
$$S_k(s_0) = s_0 + \sum_{i=1}^{k} x_i \,.$$

(a) bfloat16, $s_0 = 0, x_i \in (-0.002, 0.008)$.



(b) binary16, $s_0 = 1000, x_i \in (-0.002, 0.008)$.

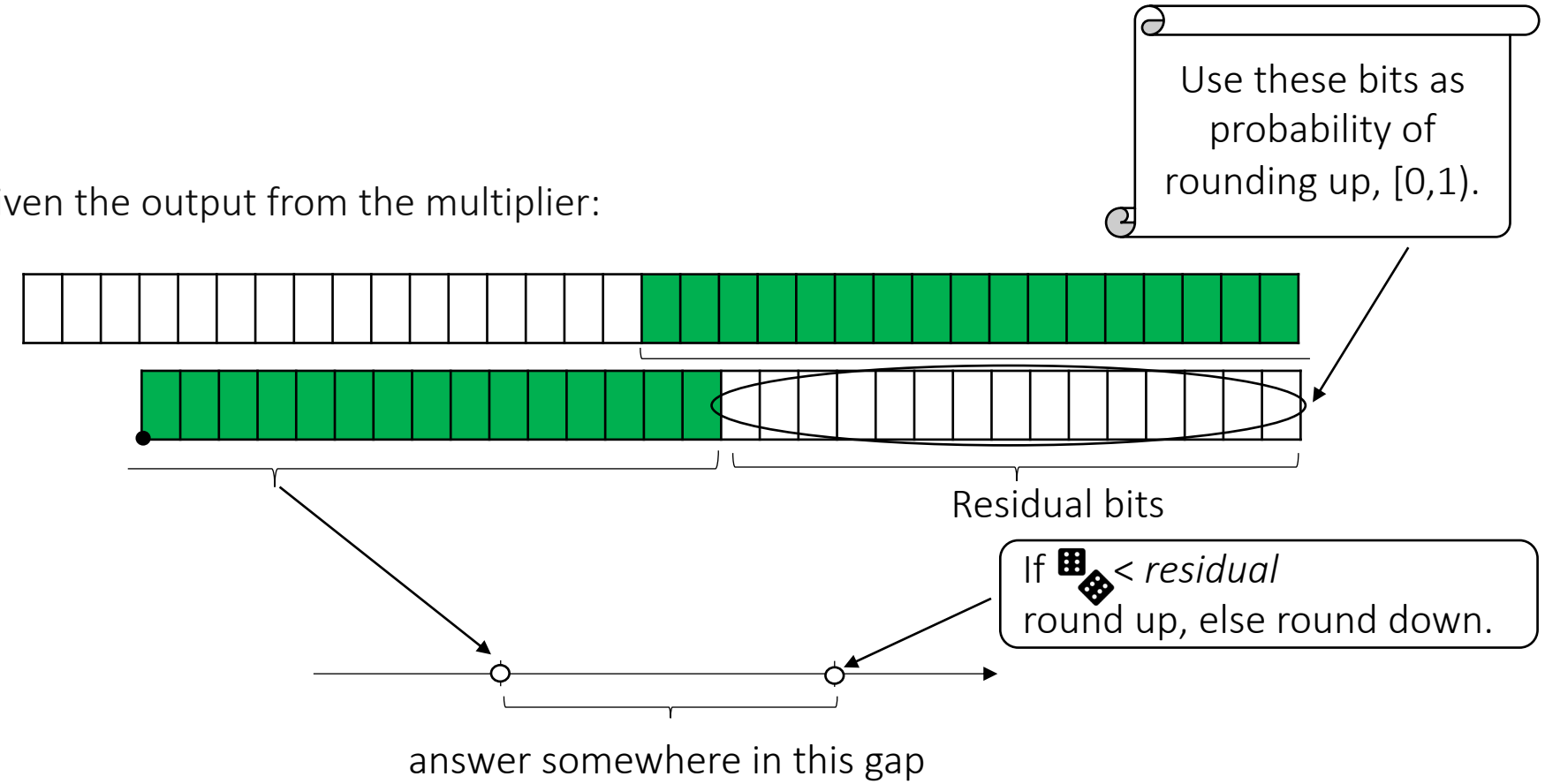




Work with M. Fasi [14] (Preprint).

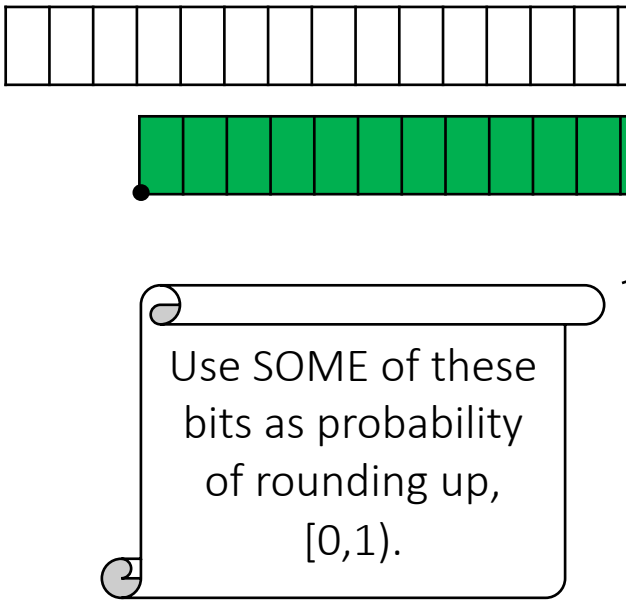
Implementation of SR

Given the output from the multiplier:

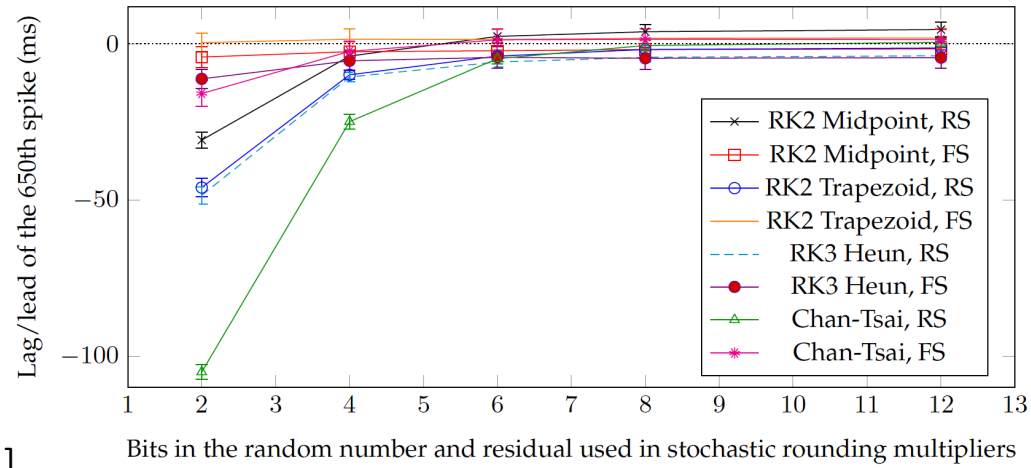


Exploration of varied bit-width in SR

Given the output from multiplier:



Use SOME of these bits as probability of rounding up, [0,1).



Work with M. Hopkins, 2019 [13].

SR in hardware

- Davies et al. 2018 [16] included SR in the Intel Loihi chip (inside the MAC units).
- Graphcore IPU [17] includes binary16 arithmetic and SR.
- I designed a small unit for doing SR on the upcoming SpiNNaker2 chip [18] (not part of ARM – fixed-point numbers in full precision come from ARM and then into this unit; also binary32 to bfloat16 SR).
- SpiNNaker2 uses one of KISS PRNGs for SR [15].

Software tools to simulate SR

For now we can easily study the behaviour of SR in software.
Can round numbers from *working precision* to lower prec:

- *Chop* MATLAB library by Higham and Pranesh [19].
- G. Meurant's MATLAB *floatp* library [20] includes floats, fixed point, and posits with SR.
- Upcoming custom precision float library in C – very efficient bit-wise impl. (work with M. Fasi; preprint and code available soon).

Algorithms for implementing SR in *working precision* without needing extended prec. libraries, by Fasi and Mikaitis [14].

Further research questions

- Applications where SR is useful (and where not).
- Complexity of PRNGs (and how many bits to use in SR).
- IEEE 754-2019 *recommends* augmented arithmetic operations that return answers as well as exact errors.
- Will allow fast compensated summation algorithms.
- Assuming augmented operations will appear in HW, how does non-augmented+SR compare to those (accuracy, hardware cost)?
- HW complexity of binary16+SR versus standard binary32/binary64 and binary16+wider accumulator.
- Overhead of SR must not cancel out the HW savings of switching to binary16.

Conclusion

- Rounding errors become more problematic as precisions in HW are reducing.
- SR is one of the interesting techniques that helps.
- SR helps to delay stagnation in long sums.
- Not yet widely available in HW, but some exists.
- Potential way to drop complicated binary32/64 units in some computers.
- Requires HW research to show how binary16+SR compares to standard binary32/64 arithmetic units.
- Other ways to improve accuracy of low-precision HW (wider internal registers, augmented ops).

References

1. M. Connolly, N. J. Higham, and T. Mary. Stochastic Rounding and its Probabilistic Backward Error Analysis. Preprint, 2020.
2. G. Forsythe. Round-off errors in numerical integration on automatic machinery (prem. report). Bull. Amer. Math. Soc. Vol 56, 1950.
3. G. Forsythe. Reprint of a Note on Rounding-Off Errors. SIAM Rev., 1959.
4. T. E. Hull, and J. R. Swenson. Tests of probabilistic models for propagation of roundoff errors. Commun. ACM, 1966.

References

4. J. Vignes. A stochastic arithmetic for reliable scientific computation. Math. And Comp. in Sim. Vol 35, 1993.
5. F. Jézéquel and J-M. Chesneaux. CADNA: a library for estimating round-off error propagation. Comp. Phys. Comm. Vol 178, 2008.
6. M. P. Nightingale, and H. W. J. Blöte. Gap of the linear spin-1 Heisenberg antiferromagnet: A Monte Carlo calculation. Phys. Rev. B. Vol 33, 1986.
7. C. R. Allton, C. M. Yung, and C. J. Hamer. Stochastic truncation method for Hamiltonian lattice field theory. Phys. Rev. D. Vol 39, 1989.
8. D. S. Parker. Monte Carlo Arithmetic: exploiting randomness in floating-point arithmetic. Tech. Rep. 1997.

References

9. F. Févote and B. Lathuilière. VERROU: Assessing Floating-Point Accuracy Without Recompiling. Preprint, 2016.
10. C. Denis, P. de O. Castro, and E. Petit. Verificarlo: checking floating point accuracy through Monte Carlo Arithmetic. Preprint, 2018.
11. M. Höhfeld and S. E. Fahlman. Probabilistic rounding in neural network learning with limited precision. Neurocomputing 4, 1992.
12. S. Gupta, A. Agawal, K. Gopalakrishnan, and P. Narayanan. Deep learning with limited numerical precision. ICML'15, 2015.
13. M. Hopkins, M. Mikaitis, D. Lester, S. Furber. Stochastic rounding and reduced-precision fixed-point arithmetic for solving neural ordinary differential equations. Phil. Tran. R. Soc. A, 2020.

References

14. M. Fasi and M. Mikaitis. Algorithms for stochastically rounded elementary arithmetic operations in IEEE 754 floating-point arithmetic. Preprint, 2020.
15. D. Jones. Good Practice in (Pseudo) Random Number Generation for Bioinformatics Applications. Tech. rep. 2010.
16. M. Davies et al. Loihi: A Neuromorphic Manycore Processor with On-Chip Learning. IEEE Micro vol 38, 2018.
17. Graphcore IPU. https://docs.graphcore.ai/projects/ipu-overview/en/latest/about_ipu.html. 2020.
18. M. Mikaitis. Stochastic Rounding: Algorithms and Hardware Accelerator. Preprint, 2020.

References

19. N. Higham and S. Pranesh. Simulating Low Precision Floating-Point Arithmetic. SIAM J. Sci. Comp. vol 41, 2019.
20. G. Meurant. https://gerard-meurant.pagesperso-orange.fr/. 2020.