MANCHESTER 1824
The University of Manchester

MCRNA
Manchester Numerical Analysis

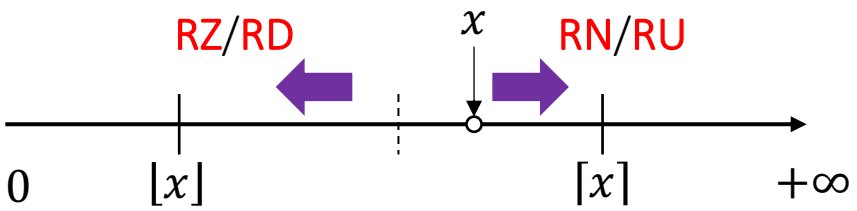
Algorithms for Stochastically Rounded Elementary Arithmetic Operations in IEEE 754 Floating-Point Arithmetic

Massimiliano Fasi, Örebro University, Sweden
Mantas Mikaitis, University of Manchester, UK

Contact: *mantas.mikaitis@manchester.ac.uk*

Virtual talk. June 2021.
28th IEEE Symposium on Computer Arithmetic (ARITH 2021).

Motivation of the project



- Rounding modes in the IEEE 754 standards:
  - RN—Round to Nearest, even on ties,
  - RZ—Round towards Zero,
  - RU—Round (Up) towards $+\infty$, and
  - RD—Round (Down) towards $-\infty$.
- *Stochastic Rounding* (SR) is starting to appear in hardware.
- Benefits shown in NLA, PDE and ODE solv., machine learning.
- Simulate SR before it is ubiquitous to
  - test behaviour,
  - develop applications with SR,
  - inform hardware of what is needed.

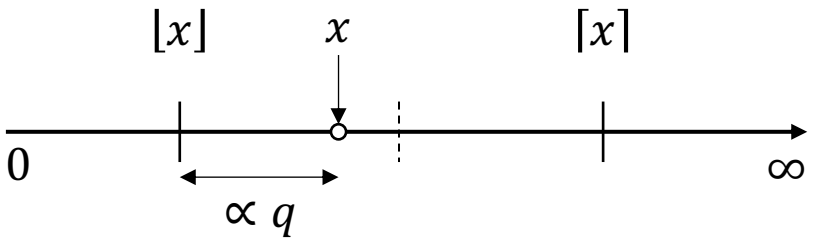
Stochastic rounding (SR)

Definition after Connolly, Higham, and Mary (2021).

Given $x \in \mathbb{R}$ with $\lfloor x \rfloor \leq x \leq \lceil x \rceil$ (with floor and ceiling defined in FP), SR is defined as

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

Mode 1	$q = \dfrac{x - \lfloor x \rfloor}{\lceil x \rceil - \lfloor x \rfloor}$
Mode 2	$q = 0.5$



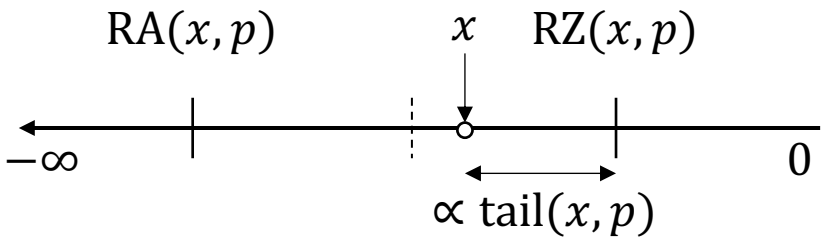
With mode 1, $\mathbb{E}\big(\mathrm{SR}(x)\big) = x$.

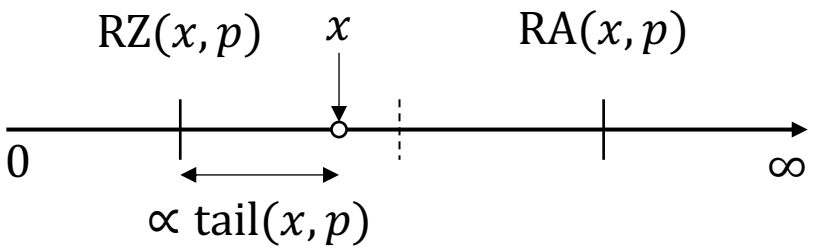
Bit-level definition of SR in floating-point

Mode 1 SR: given $x \in \mathbb{R}$, a random number $Z \in [0,1)$ from a uniform distribution and the target precision p,

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

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





SR in software and hardware

- Can round numbers from hardware prec to lower prec:
  - `chop` (MATLAB) by Higham and Pranesh (2019).
  - `floatp` (MATLAB) by Meurant (2020) includes floats, fixed point, and posits with SR.
  - `CPFloat` (C) by Fasi and Mikaitis (2020)—very efficient bit-wise implementation.

- Hardware (details not always provided):
  - Davies et al. (2018) included SR in the Intel Loihi chip (inside the MAC units to operate on 7-bit FP numbers).
  - Graphcore IPU (2020) includes binary16 arithmetic and SR.
  - There are various HW prototypes and patents appearing from the ML community.

Contributions

Problem

- Current simulators round (with SR) to precision p using at least precision $2p$.
- If we wish to simulate SR of higher precision in some hardware, we cannot use current simulators.
- Except with arbitrary precision software (Advanpix, MPFR).

Our contributions

- Algorithms for $+, -, \times, \div, \sqrt{}$ with SR in precision p.
- Generalization of binary64 algorithms by Févote and Lathuilière (2016) (use RN only).

No precision other than p required.

Error-free transformations

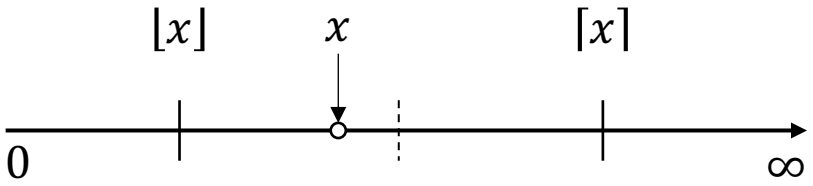
Algorithm I : TwoSum augmented addition.

function TwoSum($a \in \mathcal{F}$, $b \in \mathcal{F}$, $\circ : \mathbb{R} \to \mathcal{F}$)
Compute $\sigma, \tau \in \mathcal{F}$ *s.t.* $\sigma + \tau = a + b$.
$\sigma \leftarrow \circ(a + b)$;
$a' \leftarrow \circ(\sigma - b)$;
$b' \leftarrow \circ(\sigma - a')$;
$\delta_a \leftarrow \circ(a - a')$;
$\delta_b \leftarrow \circ(b - b')$;
$\tau \leftarrow \circ(\delta_a + \delta_b)$;
return (σ, τ);

Algorithm II : TwoProdFMA augmented multiplication.

function TwoProdFMA($a \in \mathcal{F}$, $b \in \mathcal{F}$, $\circ : \mathbb{R} \to \mathcal{F}$)
If a, b *satisfy* (5.1), *compute* $\sigma, \tau \in \mathcal{F}$ *s.t.* $\sigma + \tau = a \cdot b$.
$\sigma \leftarrow \circ(a \times b)$;
$\tau \leftarrow \circ(a \times b - \sigma)$;
return (σ, τ);

Simulating SR of (and using) precision p



- Get the distance to x and use it for SR.
- Use *error-free transforms* to compute

$$\sigma \in \{\lfloor x \rfloor, \lceil x \rceil\}, \text{ and } \tau = x - \sigma, |\tau| \in [0, 2^{e_x}\varepsilon),$$

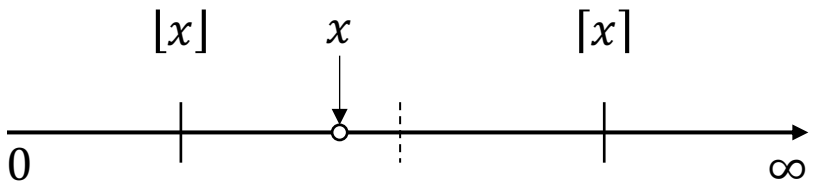
with e_x exponent of x, and $\varepsilon = 2^{1-p}$.

- Scale random number to be in $[0, 2^{e_x}\varepsilon)$ rather than computing the $\mathbf{tail}(x, p)$ (avoid division).
- We use `TwoSum` and `TwoProdFMA` for $+$ and $\times$.
- Transforms for $\div$ and $\sqrt{}$ exist, but small error in τ.

Class 1: SR addition (using RN/RZ/RU/RD)

Algorithm III : Stochastically rounded addition.

1 **function** ADD($a \in \mathcal{F}$, $b \in \mathcal{F}$)
  Compute $\varrho = \mathrm{SR}(a+b) \in \mathcal{F}$.
2 $Z \leftarrow \mathrm{rand}()$;
3 $(\sigma, \tau) \leftarrow \mathrm{TwoSum}(a, b, \mathrm{RN})$;
4 $\eta \leftarrow \mathrm{get\_exponent}(\mathrm{RZ}(a+b))$;
5 $\pi \leftarrow \mathrm{sign}(\tau) \times Z \times 2^{\eta} \times \varepsilon$;
6 **if** $\tau \geq 0$ **then**
7 $\circ = \mathrm{RD}$;
8 **else**
9 $\circ = \mathrm{RU}$;
10 $\varrho \leftarrow \circ(\diamond(\tau + \pi) + \sigma)$;
11 **return** ϱ;

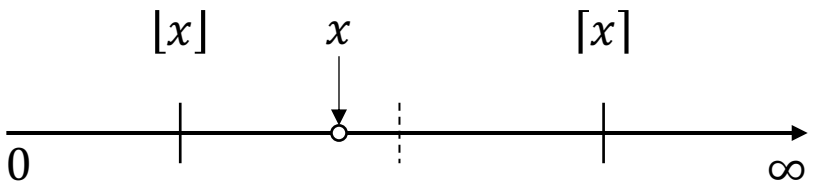


- $Z \in [0, 1)$ is a precision-p random value.
- Repeated addition with RZ deals with cases where $\lceil x \rceil$ is a power of 2.
- We choose RD or RU to round towards σ.
- Comparison in SR def. is performed by the last addition step.

Class 2: SR addition (using RN only)

Algorithm IV : A helper function for stochastic rounding.

```
1  function SRROUND(σ ∈ F, τ ∈ F, Z ∈ F)
   Compute round ∈ F.
2      if sign(τ) ≠ sign(σ) then
3          η ← get_exponent(pred(|σ|));
4      else
5          η ← get_exponent(σ);
6      ulp ← sign(τ) × 2^η × ε;
7      π ← ulp × Z;
8      if |RN(τ + π)| ≥ |ulp| then
9          round = ulp;
10     else
11         round = 0;
12     return round;
```



- Approach similar to binary64 implementation in VERROU package by Févote and Lathuilière (2016).
- Function $\mathbf{pred()}$ allows to avoid requirement of RZ.
- On line 8, comparison replaces RD/RU addition.
- Returns 0 (stay), ulp (go forward), or $-ulp$ (go backward).

SR addition (using only RN)

Algorithm V : Stochastically rounded addition without the change of the rounding mode.

1 **function** ADD2($a \in \mathcal{F}$, $b \in \mathcal{F}$)
  Compute $\varrho = \mathrm{SR}(a + b) \in \mathcal{F}$.
2 $Z \leftarrow \mathrm{rand}()$;
3 $(\sigma, \tau) \leftarrow \mathrm{TwoSum}(a, b, \mathrm{RN})$;
4 $\mathrm{round} \leftarrow \mathrm{SRround}(\sigma, \tau, Z)$;
5 $\varrho \leftarrow \mathrm{RN}(\sigma + \mathrm{round})$;
6 **return** ϱ;

In summary, algorithms of class 2 are expected to be faster on Intel, while class 1 faster where switching rounding modes has no cost.

Performance

- Implemented SR $+, -, \times, \div, \sqrt{}$ in binary64.
- Comparison with the approach that uses high-precision library.
- `MPFR 4.0.1` for computing x in higher than binary64 precision.
- 100 pairs of binary64 random numbers.
- Each op with each pair is repeated 10M times.
- Report mean throughput of ops (Mop/s) averaged over 100 pairs.
- Intel Xeon Gold 6130
- `gcc 8.2.0, -mfma -mfpmath=sse -msse2` (avoid 80-bit arithmetic).
- `-O0` for algs. that change rounding modes and `-O3` for RN-only algs.

Performance

Throughput (Mop/s):

	`add`	`add2`	`mul`	`mul2`	`div`	`div2`	`sqrt`	`sqrt2`
Mean (Mop/s)	28.2	68.8	33.9	72.4	32.2	67.2	29.5	57.4
Speedup	7.3×	17.9×	8.7×	18.6×	9.1×	19×	8.3×	16.3×

7.3× to 19× speedup over an approach that depends on the arithmetic of $> 2p$ precision.

Summary

- Hardware with SR is not yet widely available.
- Simulating SR before hardware available is an option.
- We have proposed alternative algorithms for simulating SR.
- Two classes: switch rounding modes or use only RN.
- Algorithms require only IEEE 754 operations, comparisons and some bit-level ops.
- 7.3× to 19× speedup compared with algorithms that require `MPFR` or similar.
- https://ieeexplore.ieee.org/document/9387551 (early access article).
- Implementations in `C` and `MATLAB`, and code for experiments available at

  https://github.com/mmikaitis/stochastic-rounding-evaluation.

References

- M. Connolly, N. J. Higham, and T. Mary. Stochastic rounding and its probabilistic backward error analysis. SIAM J. Sci. Comput., vol 43, 2021.
- N. Higham and S. Pranesh. Simulating low precision floating-point arithmetic. SIAM J. Sci. Comput., vol 41, 2019.
- G. Meurant. https://gerard-meurant.pagesperso-orange.fr/floatp.zip. 2020.
- M. Fasi and M. Mikaitis. CPFloat: a C library for emulating low-precision arithmetic. Preprint, 2020.
- M. Davies et al. Loihi: a neuromorphic manycore processor with on-chip learning. IEEE Micro, vol 38, 2018.
- Graphcore IPU. https://docs.graphcore.ai/projects/ipu-overview/en/latest/about_ipu.html. 2020.
- F. Févote and B. Lathuilière. VERROU: assessing floating-point accuracy without recompiling. Preprint, 2016.