UNIVERSITY OF LEEDS

Numerical Features of Low-Precision Mathematical Hardware

Mantas Mikaitis

School of Computing, University of Leeds, Leeds, UK

Mathematics and Computer Science Division Seminar Series
Chicago, IL, U.S., Jun. 28, 2023 (Online)

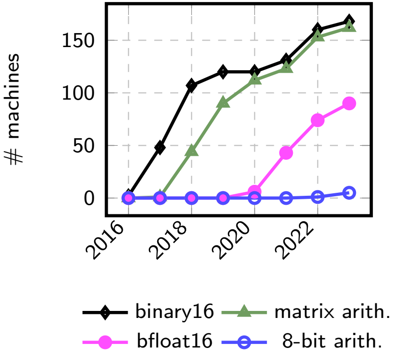
Contents

- Low- and mixed-precision mathematical hardware.
- **Stochastic rounding**: benefits and current state.
- Matrix multiply: **testing the hardware to determine features**.

What we get from today's talk

Learn how the mathematical hardware of high-performance computers is changing.

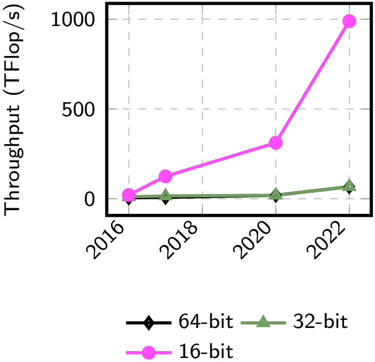
Growth of low-precision arithmetics on the TOP500



Number of machines in the TOP500 with sub-32-bit arithmetics and matrix arithmetics over the years.

Devices included: NVIDIA P100/V100/A100/H100, AMD MI210/MI250X.

Progression of throughput of arithmetics on the TOP500



Growth of the performance of arithmetics in the NVIDIA devices.

NVIDIA P100 (2016), V100 (2017), A100 (2020), H100 (2022)

GPUs in the TOP500

98% of the top 1 Frontier's power comes from GPUs.
If you don't use GPUs, go home!

J. Dongarra presenting in Manchester, 2022

(Not a direct quote)

Short introduction to floating-point arithmetic

Floating-point (FP) number representation

A floating-point system $F \subset \mathbb{R}$ is described with $\beta, t, e_{min}, e_{max}$ with elements

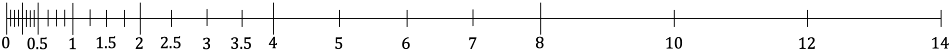
$$x = \pm m \times \beta^{e-t+1}.$$

Virtually all computers have $\beta = 2$ (binary FP).

Here t is precision, $e_{min} \leq e \leq e_{max}$ an exponent, $m \leq \beta^t - 1$ a significand (m, t, e, $m \in \mathbb{Z}$).

Toy FP system

Below: the positive numbers in $F(\beta = 2, t = 3, e_{min} = -2, e_{max} = 3)$.



Standard FP arithmetic: IEEE 754

- The standard established to achieve consistency between implementations.
- First appeared 1985, updated 2008 and 2019.
- Recommended number formats, operations, rounding modes, mathematical functions, accuracy.
- **Most computers comply with this standard.**

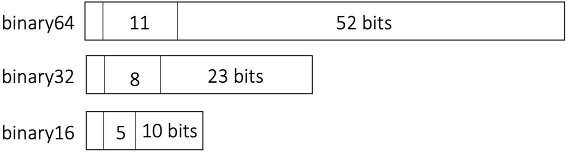
Formats with $\beta = 2$ from the standard. $f_{\min}$—smallest normalized value, $s_{\min}$—smallest denormalized value, $f_{\max}$—largest value.

	binary16	binary32	binary64
t	11	24	53
e_{min}	-14	-126	-1022
e_{max}	15	127	1023
$f_{\min}$	2^{-14}	2^{-126}	2^{-1022}
$s_{\min}$	2^{-24}	2^{-149}	2^{-1074}
$f_{\max}$	$2^{15}(2-2^{-10})$	$2^{127}(2-2^{-23})$	$2^{1023}(2-2^{-52})$

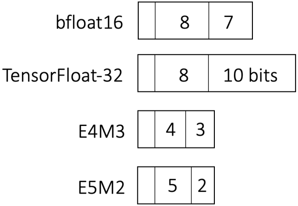
Floating-point format encoding

Numbers are held in memory using bits (convenient when $\beta = 2$).

Main IEEE 754 formats (**double**, **single**, **half**):

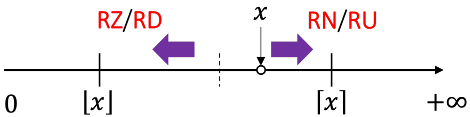


Some non-standard formats (but see **IEEE P3109**):



IEEE 754 standard FP arithmetic: rounding

- Round-to-nearest (**RN**) (ties even)
- Round-toward-zero (**RZ**)
- Round-down (**RD**)
- Round-up (**RU**)



Use of rounding modes

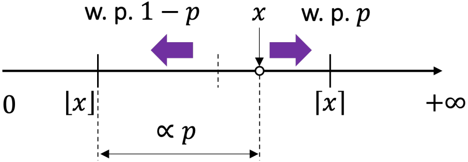
RN is usually enabled by default. Directed modes used for special cases, such as **interval arithmetic**.

Stochastic rounding

What is stochastic rounding

In **stochastic rounding** (**SR**), we are not rounding a number to the same direction, but to either direction with probability.

Given some x and FP neighbours $\lfloor x \rfloor$, $\lceil x \rceil$, we round to $\lceil x \rceil$ with prob. p and $\lfloor x \rfloor$ with $p - 1$.



Mode 1 SR: $p = \frac{x - \lfloor x \rfloor}{\lceil x \rceil - \lfloor x \rfloor}$ **Mode 2 SR**: $p = 0.5$

Mode 2

With **Mode 1 SR** we are rounding x depending on its distances to the nearest two FP numbers, **cancelling out errors of different signs**.

Mode 1 SR example

Consider rounding real numbers to integers. Round 0.25 indefinitely and then consider running total error.

Note that with **SR**, probability of rounding up is 0.25 while rounding down is 0.75.

With **RN** the total error from n roundings is $-0.25n$.

With **SR**, we can assume we **round up on every 4th number**. Error growth:

$$\downarrow -0.25 \qquad \downarrow -0.5 \qquad \downarrow -0.75 \qquad \uparrow 0$$

$$\uparrow 0.75 \qquad \downarrow 0.5 \qquad \downarrow 0.25 \qquad \downarrow 0$$

SR compared with RN

Operator $\mathrm{fl}(x)$

By $\mathrm{fl}(x)$ we denote any rounding operator that maps a number $x \in \mathbb{R}$ to F.

With both rounding modes

- If $x \in F$ $\mathrm{fl}(x) = x$.
- (**Sterbenz's lemma**) If $x, y \in F$ with $y/2 \leq x \leq 2y$ then $\mathrm{fl}(x - y) = x - y$.

Key differences of SR:

- In general $\mathrm{fl}(|x|) \neq |\mathrm{fl}(x)|$ and $\mathrm{fl}(-x) \neq -\mathrm{fl}(x)$.
- $x \leq y$ does not imply $\mathrm{fl}(x) \leq \mathrm{fl}(y)$ (non-monotonicity).
- $\mathrm{fl}(n \times \mathrm{fl}(m/n)) = m$ does not always hold.

[Connolly, Higham, Mary, 2021].

Early history of stochastic rounding

First mention by Forsythe [Forsythe, 1950]. Used in solving ODEs on early computers. Early ideas for implementation (**add random numbers to round-off digits**).

> *"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." – Forsythe in 1949*

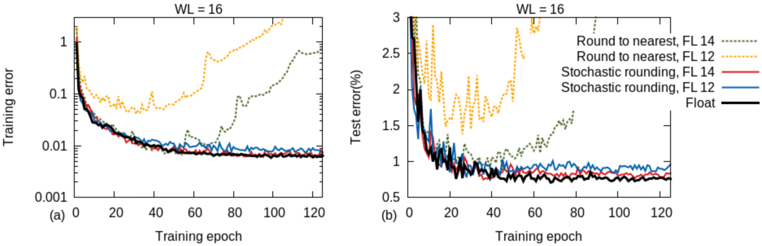
First hardware implementation by Barnes [Barnes et al., 1951]. Decimal 8-digit arithmetic. **Mode 2**. Simpler to implement than RN.

A form of SR was explored by Hull & Swenson [Hull and Swenson, 1966], used to test probabilistic error models.

SR in machine learning

SR resurfaced in machine learning, in 1992 and then 2015.

- [Höhfeld and Fahlman, 1992] used SR in training at very low precisions, such as 13 bits.
    - Update $w + \Delta w$ does not take effect as Δw rounded to zero.
    - Clamping Δw to min. val. causes non-convergence.
    - Round Δw to the minimum representable value with prob. proportional to Δw.
- [Gupta et al., 2015] used SR for training ML models with 16-bit fixed-point arithmetic.



SR in hardware

Commercial hardware that implements SR is specialized for machine learning:

- **Graphcore IPU**
- **Intel Loihi**
- **Tesla Dojo**
- **Amazon Trainium**

IEEE P3109 are considering it (see J. Demmel's talk at BIRS Workshop).

Stagnation in FP summation

Stagnation in floating-point

In summation, stagnation occurs when $\mathrm{fl}(a + b) = a$ for $a \gg b$ and $b \to 0$.

Stagnation is well illustrated with a **divergent series**

$$\sum_{i=1}^{\infty} 1/i = 1 + 1/2 + 1/3 \cdots$$

Here the addends are getting smaller while the total sum is increasing.

In limited precision arithmetic, the addends will eventually round off and the series converge.

Stagnation in FP summation

Below, stagnation/convergence points:

- **RN**: when the sum stops changing.
- **SR**: when the sum does not change for a significant number of iterations.

Arithmetic	Terms	Sum
binary64 **RN**	2^{48}	34.122
binary32 **RN**	2097152	15.404
binary32 **SR**	$\sim 50 \times 10^6$	18.303
binary16 **RN**	513	7.0859
binary16 **SR**	3.5×10^6	16.078

Rounding error analysis

Given $x \in \mathbb{R}$ that lies in the range of F it can be shown that

$$\mathrm{fl}(x \mathrm{\ op\ } y) = (x \mathrm{\ op\ } y)(1+\delta), \quad |\delta| \le u,$$

where $u = 2^{-t}$ and $\mathrm{op} \in \{+, -, \times\}$.

Model of arithmetic

This is one of the standard models used to analyse rounding errors.

Rounding error analysis

Rounding errors δ accumulate. For example, consider computing $s = x_1y_1 + x_2y_2 + x_3y_3$.

We compute $\widehat{s}$ with

$$\begin{aligned}\widehat{s} &= \Big(\big(x_1y_1(1+\delta_1) + x_2y_2(1+\delta_2)\big)(1+\delta_3) + x_3y_3(1+\delta_4)\Big)(1+\delta_5)\\ &= x_1y_1(1+\delta_1)(1+\delta_3)(1+\delta_5) + x_2y_2(1+\delta_2)(1+\delta_3)(1+\delta_5)\\ &\quad + x_3y_3(1+\delta_4)(1+\delta_5).\end{aligned}$$

Therefore we deal with a lot of terms of the form $\prod_{i=1}^{n}(1+\delta_i)$.

Worst case backward error bound (exact result for perturbed inputs)

$\prod_{i=1}^{n}(1+\delta_i) = 1+\theta_n, \quad |\theta_n| \leq \gamma_n$, with $\gamma_n = \frac{nu}{1-nu}$.

Rounding error analysis with SR

Standard error model for SR

With SR we replace u by $2u$ since it can round to the second nearest neighbour in F.

Rounding error analysis

Worst-case error analysis determines the **upper bounds of errors**, while probabilistic error analysis describes **more realistic bounds**.

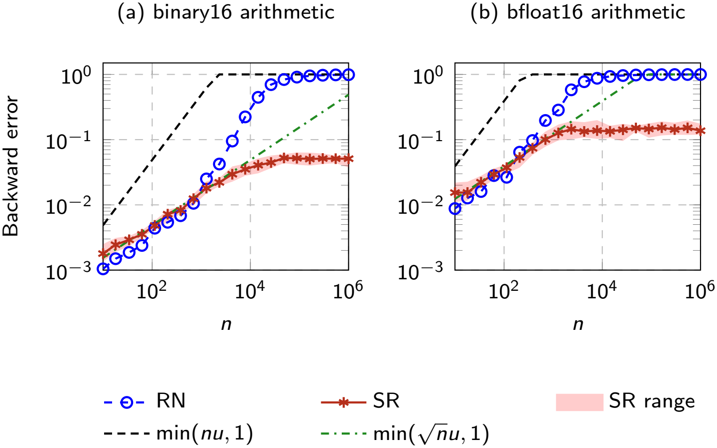
- Worst-case b-err bound with **RN**: $\frac{nu}{1-nu}$.
- Probabilistic bound with **RN**: $\lambda\sqrt{n}u + \mathcal{O}(u^2)$ w. p. $1 - 2e^{-\lambda^2/2}$. Requires an assumption that δ_n are mean independent zero-mean quantities—often satisfied [Connolly, Higham, Mary, 2021].

Wilkinson rule of thumb

$\sqrt{n}u$ error growth is a rule of thumb with **RN**, but always holds with **SR**.

Example error growth with SR in mat-vec prod

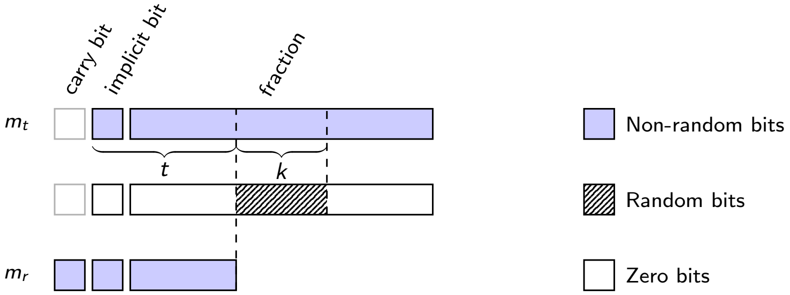
Backward error in $y = Ax$ where $A \in \mathbb{R}^{100 \times n}$ with entries from uniform dist over $[0, 10^{-3}]$ and $x \in \mathbb{R}^n$ over $[0, 1]$.



Implementation of SR

Take m_t to be a high precision unrounded significand from an operation.

Take t to be source precision and k the precision of random numbers.



Applications: ODE solvers in fixed-point arithmetic

First experimental demonstration of the efectiveness of SR outside machine learning [Hopkins, Mikaitis, Lester, Furber, 2020].

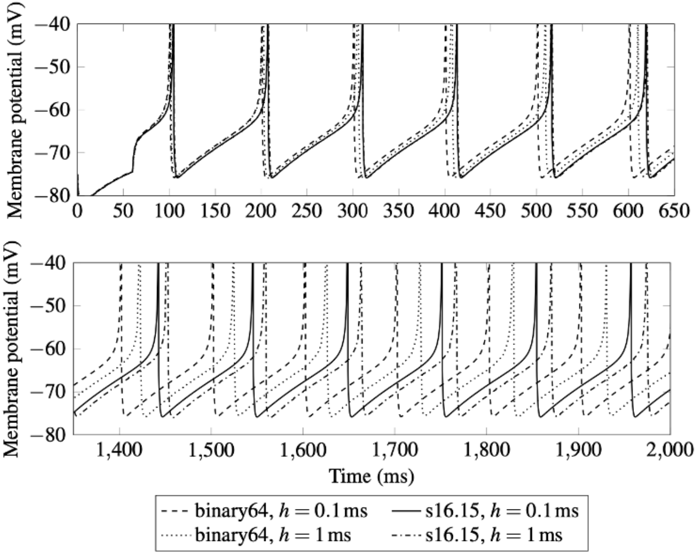
Solve ODEs that model biological neurons.

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

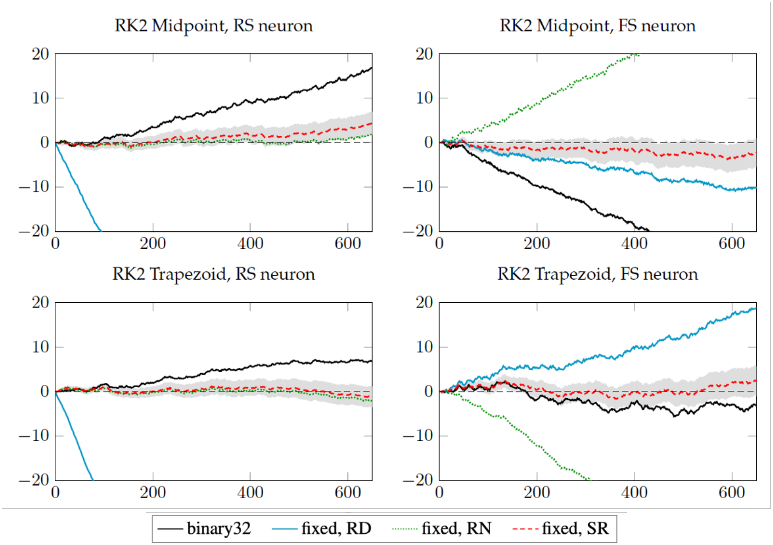
If $V \geq 30\text{mV}$ (**spike**), $V = c$, $U = U + d$.

Electical current spike times are the key in these. Spike lag should be minimized.

Applications: ODE solvers in fixed-point arithmetic



Applications: ODE solvers in fixed-point arithmetic



Applications: ODE solvers in floating-point arithmetic

Another example. Solve

$$u'(t) = v(t),\ v'(t) = -u(t).$$

With $u(0) = 1,\ v(0) = 0$ this is a **unit circle** in *uv* plane.

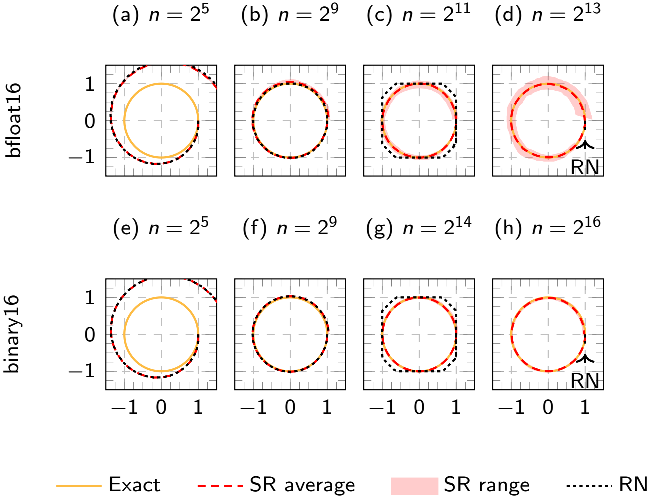
Using the Euler's method (step size $h = 2\pi/n$):

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

Experiment through *h*

Increase *n* until *h* is on the order of round-off error.

Applications: ODE solvers in floating-point arithmetic



(a) $n = 2^5$ (b) $n = 2^9$ (c) $n = 2^{11}$ (d) $n = 2^{13}$

(e) $n = 2^5$ (f) $n = 2^9$ (g) $n = 2^{14}$ (h) $n = 2^{16}$

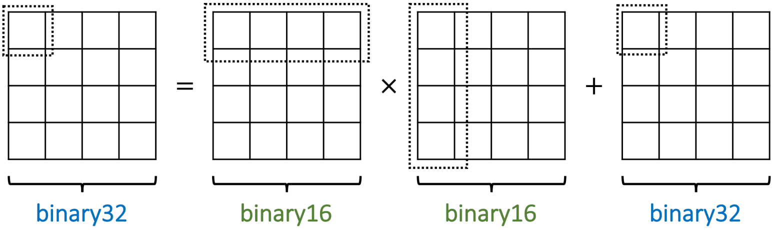
Other applications where stochastic rounding helps

See our survey for further details.

- PDE solvers.
- Numerical verification software.
- Quantum computing.
- Privacy preserving in data sets.

Testing mathematical hardware to determine numerical features

Mixed-precision matrix multipliers on GPUs



$$a_{11} \times b_{11} + a_{12} \times b_{21} + a_{13} \times b_{31} + a_{14} \times b_{41}$$

- We are used to **normalize-round after each op**.
- In hardware it is not necessarily the case.
- Normalize at the end? Savings in circuit area and latency.
- Round or drop bits? Savings.
- Accumulate in higher precision? Can do with 1-bit granularity.

Mixed-precision matrix multipliers on GPUs

Many devices now include **MMA** units:

Year	Device	Input formats	Output formats	Throughput (max)
2016	Google TPUv2	bfloat16	binary32	46 Tflop/s
2017	Google TPUv3	bfloat16	binary32	123 Tflop/s
2018	**NVIDIA V100**	binary16	binary32	125 Tflops/s
2018	Graphcore IPU1	binary16	binary32	125 Tflop/s
2020	Graphcore IPU2	binary16	binary32	250 Tflop/s
2020	**NVIDIA A100**	bfloat16, binary16/64, TensorFloat-32	binary32/64	312 Tflop/s
2021	**AMD MI250X**	bfloat16, binary16/32/64	-	383 Tflop/s
2022	**NVIDIA H100**	**quarter**, bfloat16, binary16/64, TensorFloat-32	binary32/64	2000 Tflop/s
2022	Intel Ponte Veccio	bfloat16, binary16/64, TensorFloat-32	-	-

Why test the mathematical hardware

IEEE 754 does not define strict rules on dot products:

> *"Implementations may associate in any order or evaluate in any wider format."*

- Implementations might differ.
- Not documented in detail by vendors.
- Massive speed means we are using MMAs in other areas.
- Can be said we are back to pre-IEEE-754 with mixed-precision.
- Eventually need to standardise (IEEE working group **P3109**).

For now we test to determine features.

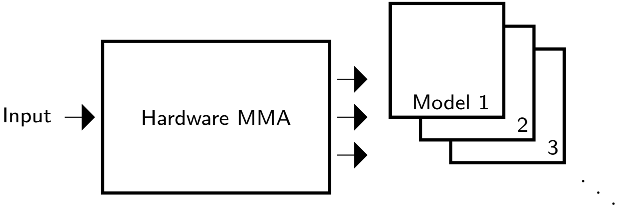
Recap: how do we test mathematical hardware

Presented at this seminar series in 2022.

Technique goes back to software called `Paranoia` from the 1980s.

MMAs more complicated than $+, \times, \div$

Find floating-point **inputs that will yield different outputs on different hardware**.



Our main findings from testing

$$a_{11} \times b_{11} + a_{12} \times b_{21} + a_{13} \times b_{31} + a_{14} \times b_{41}$$

1-bit difference in accumulators of V100 and T4/A100.

Normalization at the end, not at intermediate adds.

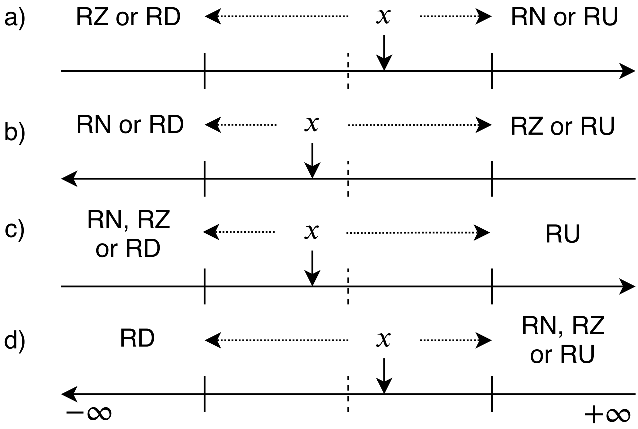
Rounding is not to nearest.

Monotonicity: increase one input, do not change order, dot product decreases. See [Mikaitis, 2023].

No fixed order.

Currently a manual process

Consider rounding:

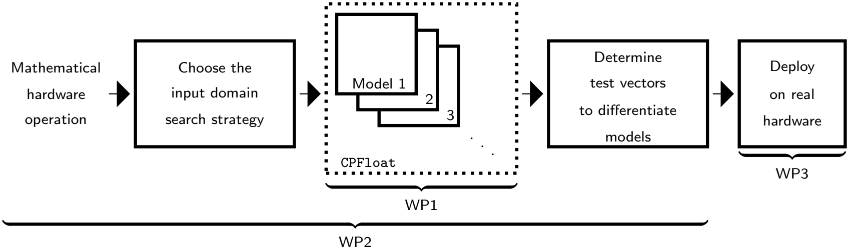


Started working on extending this work

Make the search for testing vectors automatic, or at least partially.

Remove the need for very specialized floating-point knowledge.

Make a library of behaviours and maintain as new hardware comes.


Mathematical hardware operation
Choose the input domain search strategy
Model 1
2
3
CPFloat
Determine test vectors to differentiate models
Deploy on real hardware
WP1
WP3
WP2

Summary

Main takeaway

Mathematical hardware for machine learning is fast, but the level of non-compliance to IEEE 754 is unknown and varies from device to device. Stochastic rounding is non-standard and appearing slowly in hardware.

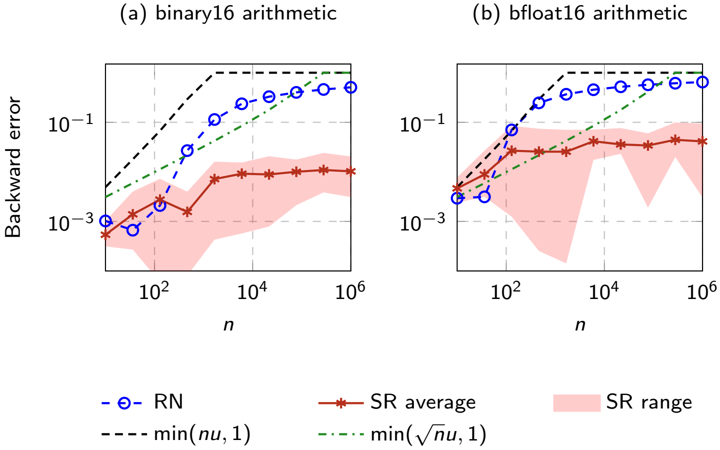
SR paper

M. Croci, M. Fasi, N. J. Higham, T. Mary, and M. Mikaitis. *Stochastic rounding: implementation, error analysis and applications*. **R. Soc. Open Sci.**. Mar. 2022.
`https://bit.ly/3Kzw7mA`.

HW testing paper

M. Fasi, N. J. Higham, M. Mikaitis, and S. Pranesh. *Numerical Behavior of NVIDIA Tensor Cores* . **PeerJ Comp. Sci**. Feb. 2021
`https://bit.ly/442IIGT`.

Extra slide: $\sum_{i=1}^{n} 1/i = 1 + 1/2 + 1/3 \cdots$ with SR



Extra slide: ODE solvers in FP arithmetic with SR

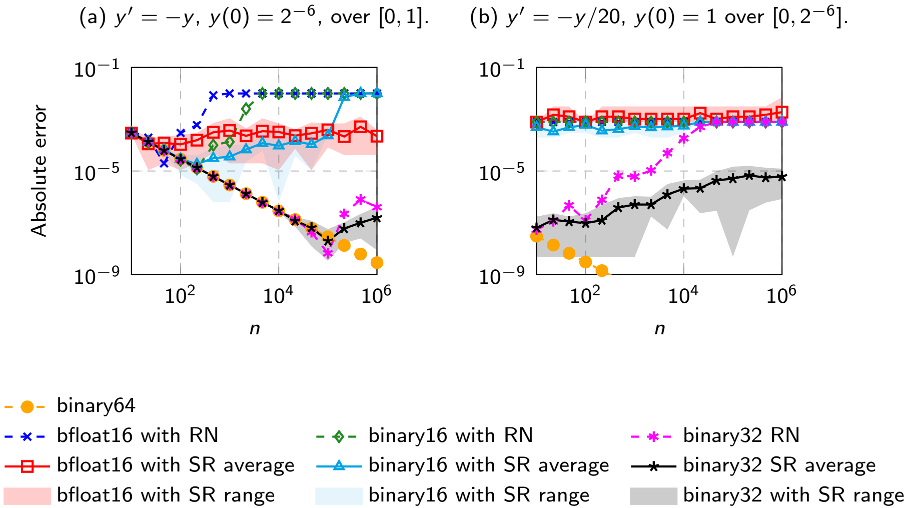
Solve two equations using the Euler's method:

- $y_{n+1} = y_n - hy_n$, with $y_0 = 2^{-6}$, in $[0, 1]$ with timestep $h = 1/n$.
- $y_{n+1} = y_n - h\frac{y_n}{20}$, with $y_0 = 1$, in $[0, 2^{-6}]$ with timestep $h = 2^{-6}/n$.

Experiment by changing n

Increase $n \in [10, 10^6]$ until h on the order of the rounding errors of a particular arithmetic.

Extra slide: ODE solvers in FP arithmetic with SR



(a) $y' = -y$, $y(0) = 2^{-6}$, over $[0, 1]$.

(b) $y' = -y/20$, $y(0) = 1$ over $[0, 2^{-6}]$.

References I

M. P. Connolly, N. J. Higham, T. Mary
Stochastic rounding and its probabilistic backward error analysis.
SIAM J. Sci. Comput. 43. 2021.

G. Forsythe
Round-off errors in numerical integration on automatic machinery.
Bull. Am. Math. Soc. 56. 1950.

R. C. M. Barnes, E. H. Cooke-Yarborough, D. G. A. Thomas
An electronic digital computor using cold cathode counting tubes for storage.
Electron. Eng. 23. 1951.

T. E. Hull, J. R. Swenson
Tests of probabilistic models for propagation of roundoff errors.
Commun. ACM. 9. 1966.

References II

M. Höhfeld and S. E. Fahlman
Learning with limited numerical precision using the cascade correlation algorithm.
IEEE Trans. Neural. Netw. 3. 1992

S. Gupta, A. Agrawal, K. Gopalakrishnan, P. Narayanan
Deep learning with limited numerical precision.
Proc. of the 32nd Int. Conf. on Machine Learning. 2015.

M. Hopkins, M. Mikaitis, D. R. Lester, S. Furber
Stochastic rounding and reduced-precision fixed-point arithmetic for solving neural ordinary differential equations.
Phil. Trans. R. Soc. 378. 2020.

M. Mikaitis
Monotonicity of Multi-Term Floating-Point Adders
arXiv:2304.01407 [cs.MS]. Apr. 2023.