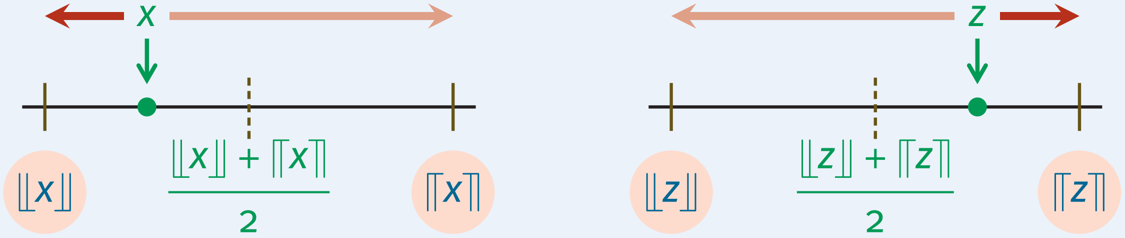
Software Simulation of Stochastic Rounding

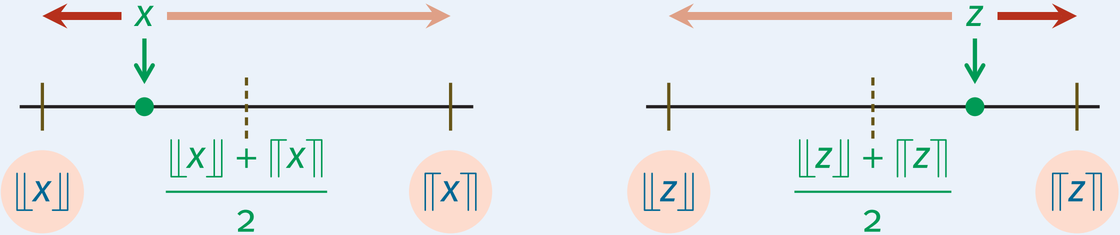
Joint work with
Mantas Mikaitis

Massimiliano Fasi
Durham University

Stochastic rounding



Stochastic rounding



How to simulate it in software?

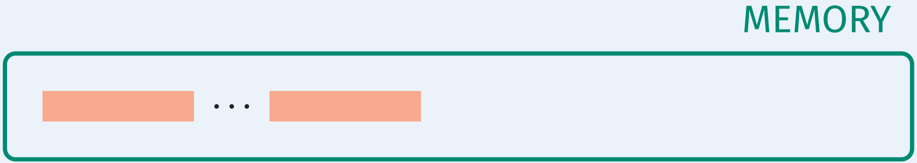
- Compute in higher precision
- Round stochastically

Floating-point formats

	Precision (digits)		Range	Relative accuracy
	Base 2	Base 10		
binary16	11	3.31	$10^{\pm 4.82}$	4.9×10^{-4}
bfloat16	8	2.41	$10^{\pm 38.53}$	3.9×10^{-3}
tfloat32	11	3.31	$10^{\pm 38.53}$	4.9×10^{-4}
binary32	24	7.22	$10^{\pm 38.53}$	6.0×10^{-8}
binary64	53	15.95	$10^{\pm 307.95}$	1.1×10^{-16}
binary128	113	34.02	$10^{\pm 4931.77}$	9.6×10^{-35}

Some available only in **high-end hardware units**.

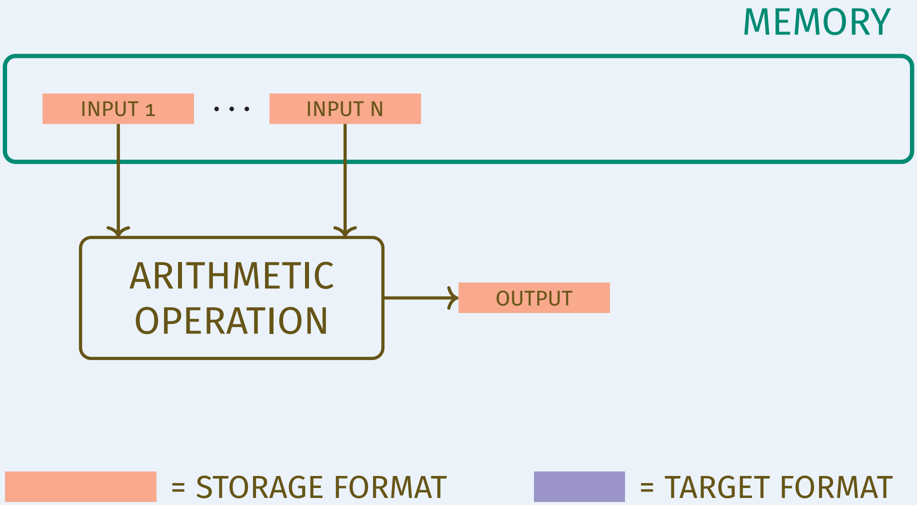
Simulating mathematical functions



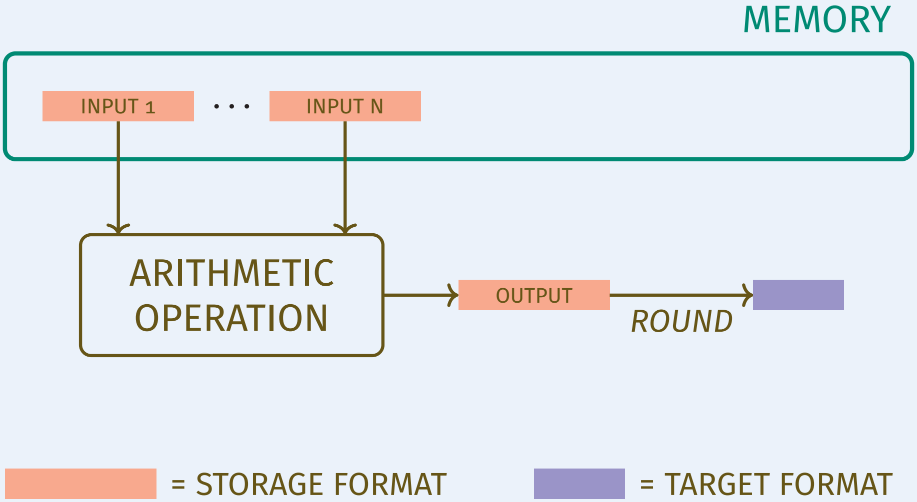
= STORAGE FORMAT

= TARGET FORMAT

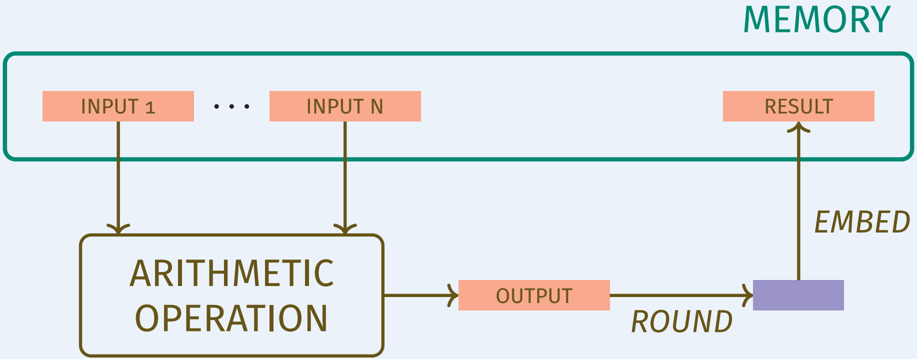
Simulating mathematical functions



Simulating mathematical functions

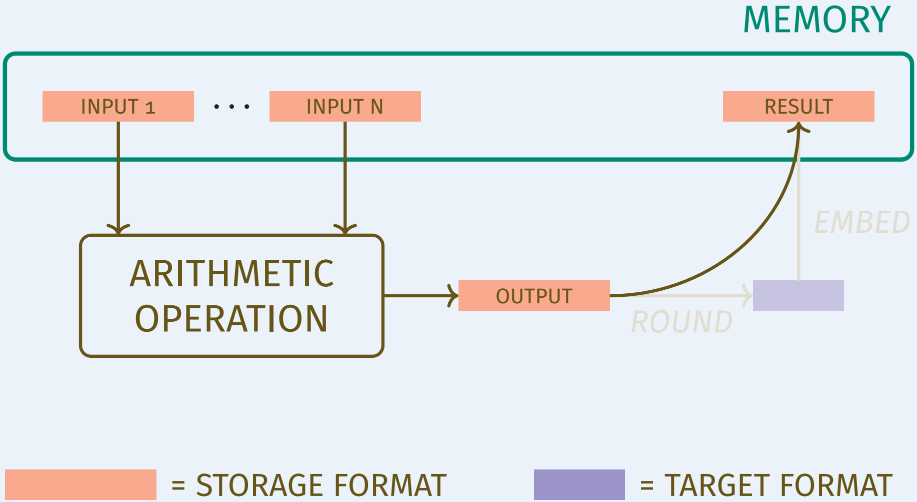


Simulating mathematical functions



= STORAGE FORMAT = TARGET FORMAT

Simulating mathematical functions



Rounding

Formats

$\mathcal{S}$: format used in memory (storage format, $p_{\mathcal{S}}$ bits of precision)

$\mathcal{T}$: format to be emulated (target format, p bits of precision)

Goal

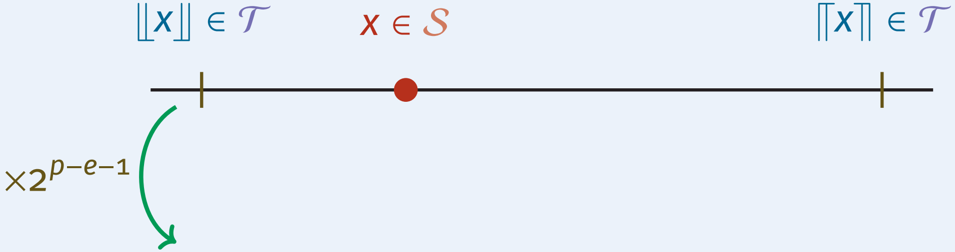
Round $x \in \mathcal{S}$ to $y \in \mathcal{T}$ according to a specified rounding mode.

Assumptions

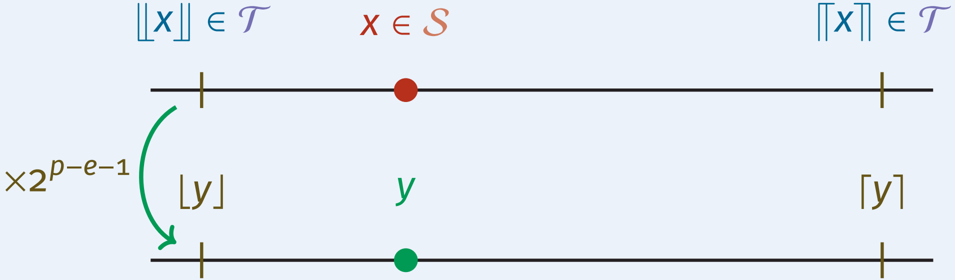
- $\mathcal{S}$ has exponent range no narrower than $\mathcal{T}$
- $\mathcal{S}$ has “enough” bits of precision

Stochastic rounding – in theory

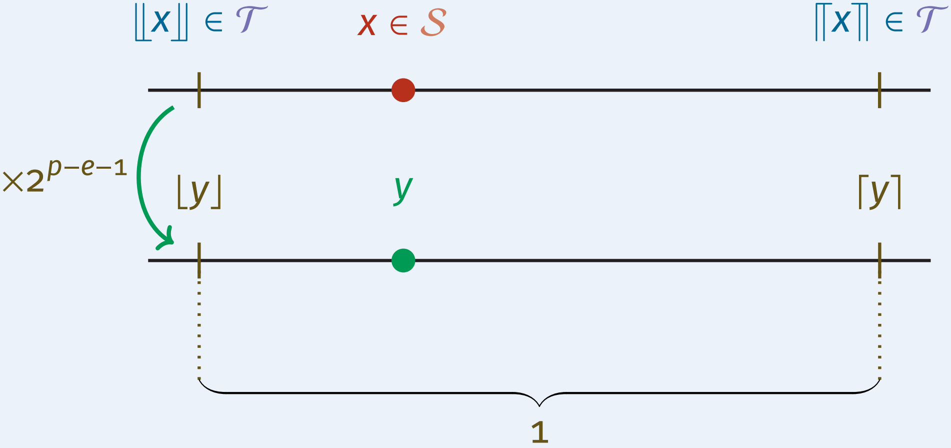
Stochastic rounding – in theory



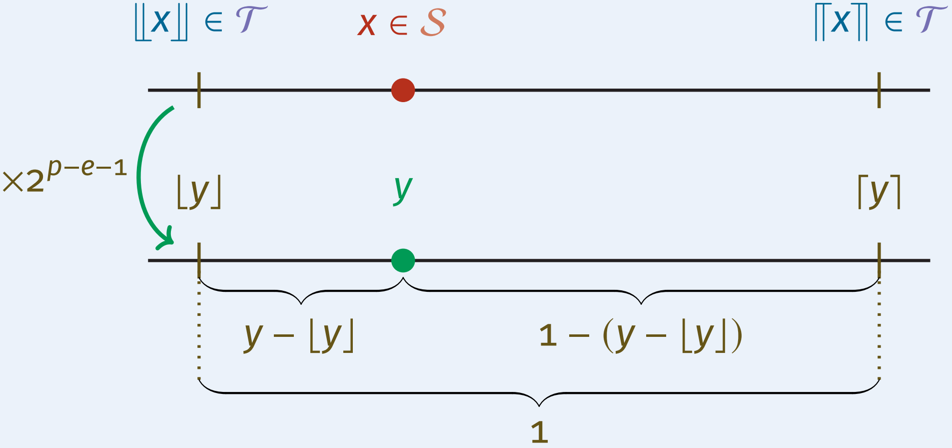
Stochastic rounding – in theory



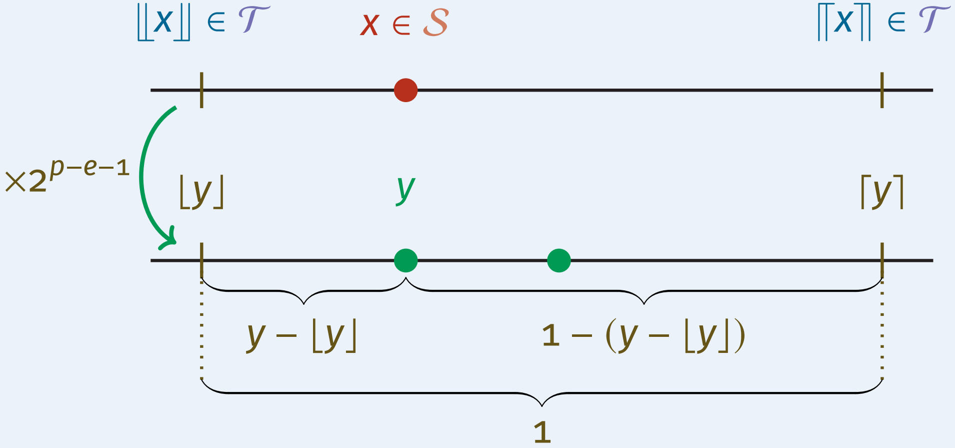
Stochastic rounding – in theory



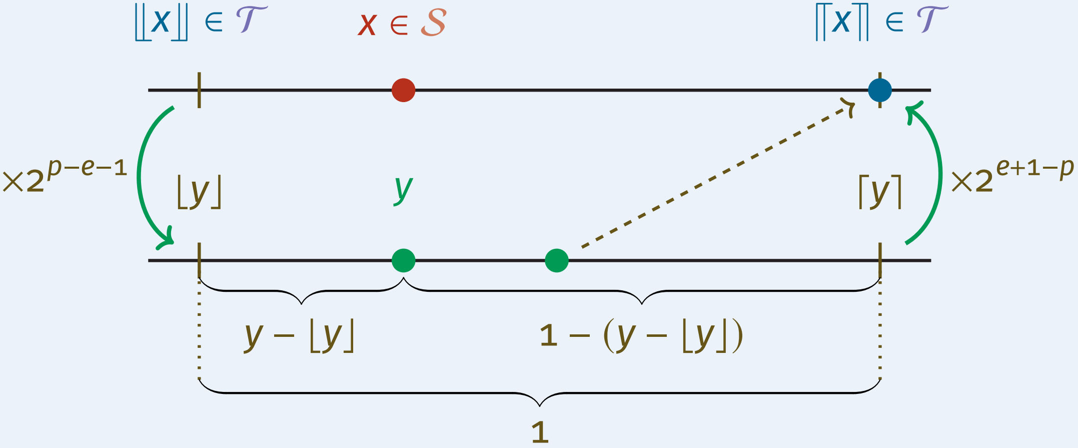
Stochastic rounding – in theory



Stochastic rounding – in theory



Stochastic rounding – in theory



Exact algorithm for stochastic rounding

Input: $x \in \mathcal{S}$

Output: x rounded stochastically to $\mathcal{T}$

```
e = ⌊log₂(x)⌋                 ▷ Compute exponent
y = 2^(p−e−1) · x             ▷ Move radix point
Draw ρ from U_(0,1).
if ρ ≤ y − ⌊y⌋ then           ▷ Round to integer
    y ← ⌈y⌉
else
    y ← ⌊y⌋
return 2^(e+1−p) · y          ▷ Move radix point back
```

Example

Round stochastically to a format with $p = 4$ the number

$$x = 11.000110_2 = 2^1 \cdot 1.1000110_2$$

Example

Round stochastically to a format with $p = 4$ the number

$$x = 11.000110_2 = 2^1 \cdot 1.1000110_2$$

$$y = 2^{4-1-1} \cdot x = 1100.0110_2$$

Example

Round stochastically to a format with $p = 4$ the number

$$x = 11.000110_2 = 2^1 \cdot 1.1000110_2$$

$$y = 2^{4-1-1} \cdot x = 1100.0110_2$$

Possible return values

- $2^{1+1-4} \cdot \lfloor y \rfloor = 2^{-2} \cdot 1100 = 110.0_2$ (probability 0.615)
- $2^{1+1-4} \cdot \lceil y \rceil = 2^{-2} \cdot 1101 = 110.1_2$ (probability 0.375)

Benefits and issues

- ▲ Simple algorithms
- ▲ Straightforward implementation
- ▲ Floating-point storage is hidden
- ▼ Library functions are slow
- ▼ Library functions are prone to numerical errors
- ▼ Floating-point arithmetic is used throughout

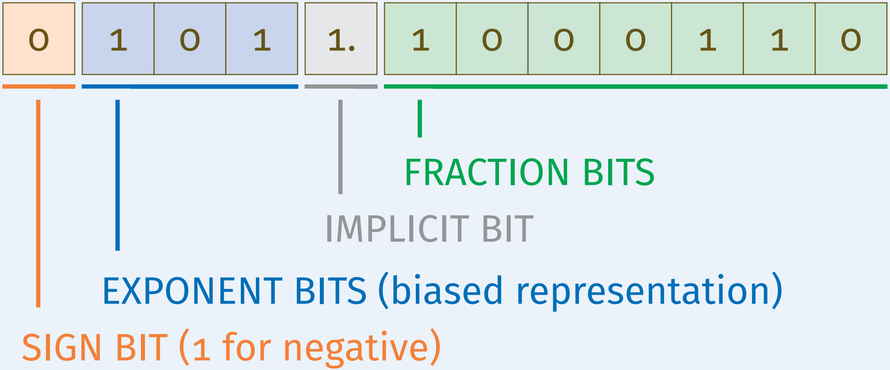
Higham & Pranesh. *Simulating Low Precision Floating-Point Arithmetic*. SIAM J. Sci. Comput., 41(5):585–602, 2019.

Floating-point number representation

$$x = (-1)^{0} \cdot 2^{\,2} \cdot 1.1000110_2$$

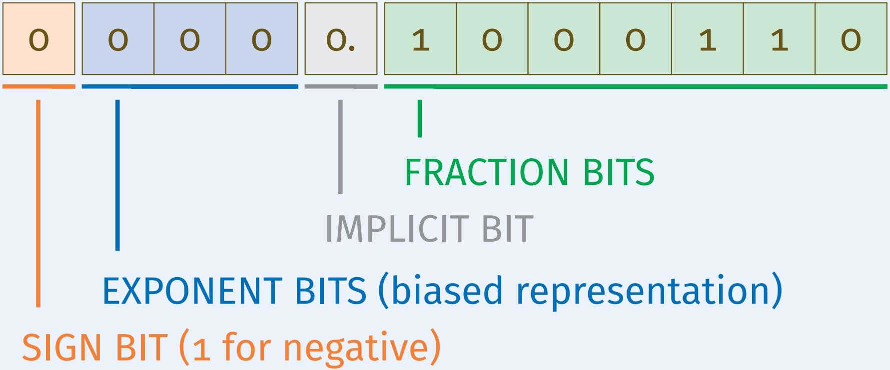
Floating-point number representation

$$x = (-1)^{0} \cdot 2^{2} \cdot 1.1000110_2$$



Floating-point number representation

$$x = (-1)^{0} \cdot 2^{-3} \cdot 0.1000110_2$$



Stochastic rounding – from 8 to 4 digits

0	1	0	1	1.	1	0	0	0	1	1	0

Stochastic rounding – from 8 to 4 digits

0	1	0	1	1.	1	0	0	0	1	1	0	+
0	0	0	0		0	0	0	1	0	1	1	=
0	1	0	1	1.	1	0	1	0	0	0	1	

Stochastic rounding – from 8 to 4 digits

0	1	0	1	1.	1	0	0	0	1	1	0	+
0	0	0	0		0	0	0	1	0	1	1	=
0	1	0	1	1.	1	0	1	0	0	0	1	AND
1	1	1	1		1	1	1	0	0	0	0	=
0	1	0	1	1.	1	0	1	0	0	0	0	

One bitwise and one integer arithmetic operation required.

CPFloat: simulating low precision in C

Design principles

- Vector-oriented
- Parallel (using OpenMP)
- Extensible

Features *(cf. `math.h` library)*

- Floating-point rounding functions
- Mathematical functions (+, −, $\sqrt{\ }$, $\lceil\ \rceil$, sin, cosh, x^y, log, exp, ...)
- "Floating-point" functions (`frexp`, `nextafter`, `isnormal`, ...)

Rounding modes

- Round-to-nearest
  - with ties-to-even (RNE)
  - with ties-to-zero (RNZ)
  - with ties-to-away (RNA)
- Directed rounding (RD)
  - round-to-zero
  - round-to-$+\infty$
  - round-to-$-\infty$
- Round-to-odd (RO)
- Stochastic rounding (RS)

Similar packages

Package name	Primary language	Storage format	RNE	RNZ	RNA	RD	RO	RS
GNU MPFR	C	custom	✓		✠	✓		
rpe	Fortran	b64	✓					
FloatX	C++	b32/b64	✓					
chop	MATLAB	b32/b64	✓			✓		✓
QPyTorch	Python	b32	✓					
FLOATP	MATLAB	b64	✓			✓		✓
CPFloat	C	b32/b64	✓	✓	✓	✓	✓	✓

Similar packages

Package name	Primary language	Storage format	RNE	RNZ	RNA	RD	RO	RS
GNU MPFR	C	custom	✓		✠	✓		
rpe	Fortran	b64	✓					
FloatX	C++	b32/b64	✓					
chop	MATLAB	b32/b64	✓			✓		✓
QPyTorch	Python	b32	✓					
FLOATP	MATLAB	b64	✓			✓		✓
CPFloat	MATLAB	b32/b64	✓	✓	✓	✓	✓	✓

Performance of MATLAB interface

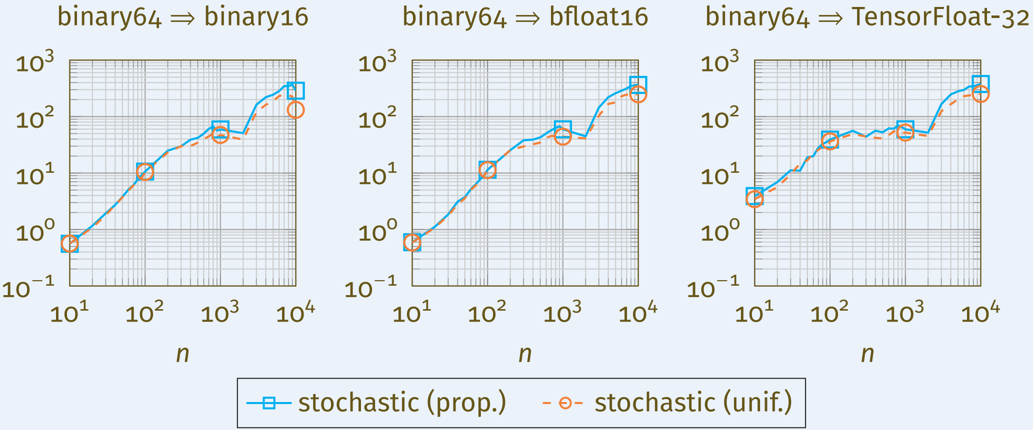


Figure: Ratio of runtime of chop to that of CPFloat on $n \times n$ matrices.

Some details

- Header-only C library
- Only (optional) dependency: PCG Random Number Generator
- Distributed under GNU LGPL v. 2.1 or later
- Freely available on GitHub

O'Neill. *PCG: A Family of Simple Fast Space-Efficient Statistically Good Algorithms for Random Number Generation*. Technical report HMC-CS-2014-0905, Harvey Mudd College, 2014.

Not covered

- Similar techniques for other rounding modes
- CPFloat has mathematical and floating-point functions
- Simulation using error-free transformations

$$x \diamond y = r + e, \qquad \diamond \in \{+, -, \times, \div\}$$

Fasi & Mikaitis. *Algorithms for Stochastically Rounded Elementary arithmetic Operations in IEEE754 Floating-Point Arithmetic.* IEEE Trans. Emerg. Topics Comput., 9(3):1451–1466, 2021.

Future work

- Vectorise the algorithms
- Add C++ and MATLAB class interfaces
- Add facilities for matrix computations

github.com/north-numerical-computing/cpfloat

north-numerical-computing.github.io/cpfloat

Fasi & Mikaitis. CPFloat: A C Library For Emulating Low-Precision Arithmetic. ACM Trans. Math. Softw. 49(2), Article 18, 2023.

APPENDIX

Innocuous double rounding

If $\mathrm{fl}_{\mathcal{S}} : \mathbb{R} \to \mathcal{S}$ is a round-to-nearest function and $\mathrm{fl}_{\mathcal{T}} : \mathcal{S} \to \mathcal{T}$ is a round-to-nearest or a directed rounding function, then

$$\mathrm{fl}_{\mathcal{T}}\big(\mathrm{fl}_{\mathcal{S}}(x \,\square\, y)\big) = \circ(x \,\square\, y), \qquad \square \in \left\{ +, -, \times, \div, \sqrt{\ } \right\}$$

if $p \leq p_{\mathcal{S}}/2 - 1$.

Roux. *Innocuous Double Rounding of Basic Arithmetic Operations*. J. Formaliz. Reason., 7(1):131–142, 2014.

Rump. *Precision-k base-β Arithmetic Inherited by Precision-m Base-β Arithmetic for $k < m$*". ACM Trans. Math. Software, 43(3):1–15, 2017.

Idea of the algorithm

Input: $x \in \mathcal{S}, \mathcal{T}$
Output: x rounded to $\mathcal{T}$

```
if x is too small to be represented in 𝒯 then
    return sign(x) · 0
else if x is too large to be represented in 𝒯 then
    return sign(x) · ∞
else
    if x is subnormal in 𝒯 then
        Reduce p accordingly.
    return x rounded to p significant binary digits.
```