



# Monotonicity of Multi-Term Floating-Point Adders

Mantas Mikaitis

School of Computing, University of Leeds, Leeds, UK

AriC Team Seminar, Laboratoire LIP  
Lyon, France, Apr. 11, 2024

Slides: <https://mmikaitis.github.io/talks>



# Monotonicity of summation

Summation over real values

$$f(x_1, x_2, \dots, x_n) = \sum_{i=1}^n x_i,$$

is monotonic because for any  $x_i < x_i^*$  we have that

$$f(x_1, \dots, x_n) \leq f(x_1^*, \dots, x_n^*).$$

## Monotonicity

With multivariate monotonic functions, if one or more of the arguments is increased, the function also increases or stays constant.

# Today's talk

When computing summation **in floating-point arithmetic we would like to preserve monotonicity** of the sum.

Some of the known properties of floating point:

- ✓ Commutativity:  $\text{fl}(a \times b) = \text{fl}(b \times a)$ .
- × Associativity:  $\text{fl}(a + \text{fl}(b + c)) \neq \text{fl}(\text{fl}(a + b) + c)$ .
- × Distributivity:  $\text{fl}(a \times \text{fl}(b + c)) \neq \text{fl}(\text{fl}(a \times b) + \text{fl}(a \times c))$ .
- ? Monotonicity

By the end of this talk...

learn about the monotonicity of floating-point sum and how it is affected by the mathematical hardware computing it.

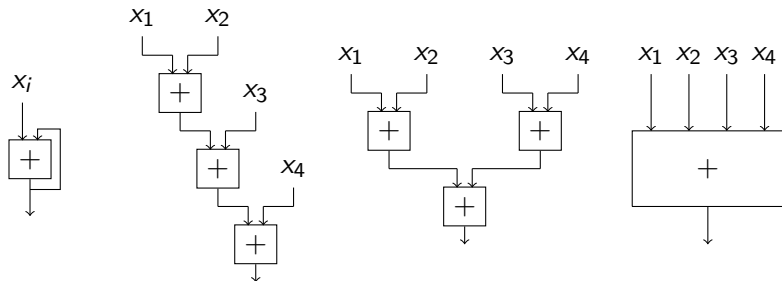
# Addition of multiple numbers in floating point

We are used to adding numbers in pairs, using a two-term FP adders.

IEEE 754 adder **computes as though in infinite precision, then normalizes and rounds.**

## Multi-term adders

Latest hardware includes specialized **multi-term adders** alongside the standard **two-term addition**.



## Standard model [Higham, 2002]

Floating-point addition defined with

$$\text{fl}(x + y) = (x + y)(1 + \delta), \quad |\delta| \leq 2^{-p}$$

where  $\text{fl}()$  refers to normalizing and rounding  $x + y$  to form a floating-point value defined by IEEE 754.

Following classes of multi-term adders are present in current hardware literature:

- Class I (**exact “Kulisch” accumulator**) and Class II (**compute sticky bits correctly [Tenca, 2009]**):  $\text{fl}(x_1 + x_2 + \dots + x_n)$ .
- Class III (**chain of two-term adders**):  
 $\text{fl}(\text{fl}(\dots \text{fl}(x_1 + x_2) + \dots) + x_n)$ .

# Monotonicity in FP: basic results

- Rounding  $\text{fl}(x)$  is monotonic by definition.
- Addition  $\text{fl}(x + y)$  is monotonic.
- Summation  $\text{fl}(\text{fl}(\cdots \text{fl}(x_1 + x_2) + \cdots) + x_n)$  is monotonic for any  $n$  (**Class III**).
- Multiplication  $\text{fl}(x \times y)$  is monotonic.
- Inner product  $\text{fl}(\cdots \text{fl}(\text{fl}(a_1 \times b_1) + \text{fl}(a_2 \times b_2)) + \cdots + \text{fl}(a_n \times b_n))$  is monotonic (**Class III**).
- Fused computations  $\text{fl}(x_1 + x_n + \cdots + x_n)$  are monotonic (**Class I/II**).

## Proofs

The proofs of these come down to the monotonicity of rounding and work with the main IEEE 754 rounding modes: **round-to-nearest**, **round-towards-zero**, **round-up**, and **round-down**.

# Monotonicity in FP: basic results

But there is a catch: changing order can break monotonicity because in general  $\text{fl}(a + \text{fl}(b + c)) \neq \text{fl}(\text{fl}(a + b) + c)$ .

Example in binary32:  $a = 1$ ,  
 $\text{fl}(\text{fl}(\text{fl}(\text{fl}(a + 2^{-24}) + 2^{-24}) + 2^{-24}) + 2^{-24}) = 1$ .

Now **decrease**  $a$  to  $a = 1 - 2^{-24}$  and change the order of evaluation:  
 $\text{fl}(\text{fl}(\text{fl}(\text{fl}(2^{-24} + 2^{-24}) + 2^{-24}) + 2^{-24}) + a) = 1 + 2^{-22}$ .

Community knows this behaviour well and expects this.

But if order is unchanged this is not expected to happen.

# Standard floating-point representation

- A binary floating-point number  $x$  has the form  $(-1)^s \times m \times 2^{e-p+1}$
- $s$  is the sign bit,  $p$  is the precision,  $m \in [0, 2^p - 1]$  is the integer significand, and  $e \in [e_{\min}, e_{\max}]$ , with  $e_{\min} = 1 - e_{\max}$ , is the integer exponent.
- The number system is *normalized* so that the most significant bit of  $m$  is always set to 1 if  $|x| \geq 2^{e_{\min}}$ .
- Floating-point numbers with  $2^{p-1} \leq m \leq 2^p - 1$  are normalized.

## Normalization

The result of an operation must be normalized by **shifting the significand** left or right until it falls within the interval  $[2^{p-1}, 2^p - 1]$  and **adjusting the exponent** accordingly.



## Modified standard model

We define an adder that starts with some precision  $p$  and can grow precision when carries occur in floating-point significand addition.

$$\text{flr}(a + b) = \begin{cases} \text{fl}_p(a + b) & \text{if } |a + b| < t, \\ \text{fl}_{p+1}(a + b) & \text{if } |a + b| \geq t, \end{cases} \quad (1)$$

with  $|a| \geq |b|$ ,  $t = 2^{1 + \lceil \log_2 |a| \rceil}$ , power of two nearest to  $|a|$  with  $|a| < |t|$ .

- When this adder is joined to compute expressions such as  $\text{flr}(\text{flr}(x_1 + x_2) + x_3)$ , precision increases propagate.
- Sum with  $n$  additions can grow precision from  $p$  to  $p + \lceil \log_2 n \rceil$ .
- Similar device used by [\[Ashenurst and Metropolis, 1959\]](#) for error analysis.

## Class IV multi-term adders

---

**Algorithm 0:** Given numbers  $\{x_1, \dots, x_n\}$ , with exponents  $\{e_1, \dots, e_n\}$  and significands  $\{1.m_1, \dots, 1.m_n\}$  approximate  $s_n = \sum_{i=1}^n x_i$ .

---

Determine  $e_{max} = \max(e_1, \dots, e_n)$

Align all  $1.m_i$  by shifting each  $e_{max} - e_i$  steps right

Perform addition of aligned significands

Perform normalization and rounding to form  $s_n$

---

We can partially model **Class IV** adders:  $\text{fl}(\text{flr}(\dots \text{flr}(x_1 + x_2) + \dots) + x_n)$ .

Hardware properties:

- start in limited precision accumulator  $p$ ,
- **do not normalize and round until the computation is finished**,
- due to carries, **grow effective precision**.

Note  $\text{flr}()$  does not account for lack of normalization after cancellation, where precision loss occurs. Not addressed in this work.

# Monotonicity: example on current hardware

Originally observed in [Fasi, Higham, Pranesh, Mikaitis, 2021].

Recent GPUs contain hardware for  $D = A \times B + C$  where  $A \in \mathbb{R}^{8 \times 8}$  and  $B \in \mathbb{R}^{8 \times 4}$  are binary16 matrices,  $C, D \in \mathbb{R}^{8 \times 4}$  are binary32 matrices.

We will focus on two result elements in  $D$ :

- $d_{11} = a_{11}b_{11} + a_{12}b_{21} + \dots + a_{18}b_{81} + c_{11}$
- $d_{12} = a_{11}b_{12} + a_{12}b_{22} + \dots + a_{18}b_{82} + c_{12}$

We set  $A, B = 1$  (matrices of ones) and  $c_{11} = 33554430$  and  $c_{12} = 33554432$ .

Computing  $A \times B + C$  with a GPU returns a matrix that has  $d_{11} = 33554436$  and  $d_{12} = 33554432$ .

Since  $c_{11} < c_{12}$  but  $d_{11} > d_{12}$  the 9-term sum is nonmonotonic.

# Commercial devices

Year	Device/Architecture	Input formats	Output formats	Terms	Predicted class
2016	Google TPUv2	bfloat16	binary32	-	Class III
2017	Google TPUv3	bfloat16	binary32	-	Class III
2018	NVIDIA V100	binary16	binary32	5	Class IV
2018	Graphcore IPU1	binary16	binary32	-	-
2020	Google TPUv4i	bfloat16	binary32	4	Class IV
2020	Graphcore IPU2	binary16	binary32	-	-
2020	NVIDIA A100	bfloat16, binary16, binary64, TensorFloat-32	binary32/64	9	Class IV
2021	AMD MI250X	bfloat16, binary16, binary32, binary64	-	5	-
2021	GroqChip	binary16	binary32	160	Class I or II
2022	NVIDIA H100	8-bit, bfloat16, binary16, binary64, TensorFloat-32	binary32, binary64	17	-
2022	Intel Ponte Vecchio	bfloat16, binary16, binary64, TensorFloat-32	-	-	-
2016-2022	Intel AMX	binary16	binary32	17	Class III
2023	Tesla Dojo	CFP8, bfloat16	binary32	8	Class IV
2024	NVIDIA Blackwell	block 4/6/8-bit, [...]	binary32, binary64	-	-

## Results on Class IV adders

- Addition  $\text{flr}(x + y)$  is monotonic with round-to-nearest, round-towards-zero, round-up, and round-down.
- Addition of three operands  $\text{flr}(\text{flr}(x_1 + x_2) + x_3)$  is **non-monotonic** with round-to-nearest, round-toward-zero and round-down, **except if rounded to starting precision**  $\text{fl}(\text{flr}(\text{flr}(x_1 + x_2) + x_3))$ .

Proof:

- Three consecutive positive FP numbers  $a$ ,  $b$  (a power of 2),  $c$ .
- $a < b < c$ ,  $\varepsilon = \frac{c-b}{2}$ .
- $\text{flr}(b + \varepsilon) = b$  and  $\text{flr}(\text{flr}(b + \varepsilon) + \varepsilon) = b$  with RN, RD, RZ.
- $\text{flr}(a + \varepsilon) = b$  (**precision grows**).
- $\text{flr}(\text{flr}(a + \varepsilon) + \varepsilon) > b$  due to precision growth in the first add.
- $\text{fl}(\text{flr}(\text{flr}(a + \varepsilon) + \varepsilon))$ : OK.

## Main result

Summation  $\text{flr}(\cdots \text{flr}(x_1 + x_2) + \cdots) + x_n$ , with  $x_i \in \mathbb{R}$  and  $n \geq 4$  is not monotonic with round-to-nearest, round-towards-zero, and round-toward-negative ( $x_i > 0$ ) or round-toward-positive ( $x_i < 0$ ), with and without the final rounding to the starting precision.

- Three consecutive positive FP numbers  $a$ ,  $b$  (a power of 2),  $c$ .
- $a < b < c$ ,  $\varepsilon = \frac{c-b}{2}$ .
- With RN,  $\text{flr}(b + \varepsilon) = b$  for  $\varepsilon \leq (c - b)/2$ , while in precision- $(p + 1)$  arithmetic  $\text{flr}(b + \varepsilon) = b$  for  $\varepsilon \leq (c - b)/4$ .
- Also, in precision- $p$  arithmetic  $\text{flr}(a + (c - b)/2) = b$ .

Consider  $\text{flr}(\text{flr}(\text{flr}(x + \varepsilon) + \varepsilon) + \varepsilon)$  in two cases:

- ①  $x = b$ , then  $\text{flr}(\text{flr}(\text{flr}(b + \varepsilon) + \varepsilon) + \varepsilon) = b$  (all in precision- $p$ ).
- ②  $x = a$ , then the first addition  $\text{flr}(a + \varepsilon) = b$  (and precision increases to  $p + 1$  since  $b$  is a power of two).
  - the second addition  $\text{flr}(b + \varepsilon) = b + \varepsilon$  (in precision  $p + 1$ );
  - the third addition  $\text{flr}(b + \varepsilon + \varepsilon) = c$  (in precision  $p + 1$ ).

When  $x = b$ , sum evaluates to  $b$ , but when  $x = a < b$ , sum evaluates to  $c > b$ . Final rounding to  $p$  does not change the results.  $\square$

# Custom precision simulators

- Various packages available: chop, FLOATP, QPyTorch.
- Usual approach is to perform ops in binary32/64 HW.
- Round down to sub-32-bit precision: careful with double rounding.
- We believe ours is most customizable and fastest: CPFloat [[Fasi & Mikaitis, 2023](#)].
- Can be used in MATLAB, Octave or C.



## Example with CPF10at in MATLAB/Octave

```
>> options.format = 'binary16';
>> [~,options] = cpf10at(0, options)
options =
  struct with fields:
    format: 'binary16'
    params: [11 15]
    subnormal: 1
    round: 5
    flip: 0
    p: 0.5000
    explim: 1
>> cpf10at(pi, options)
ans =
    3.1406
>> options.params(1) = options.params(1) + 1;
>> cpf10at(pi, options)
ans =
    3.1426
```

# Numerical experiments

- We simulated Class IV multi-term adders with MATLAB, using the custom precision simulator CPFloat [Fasi and Mikaitis, 2023].

- Compute

$$\text{fl}(\cdots \text{fl}(x_1 + x_2) + \cdots) + x_n$$

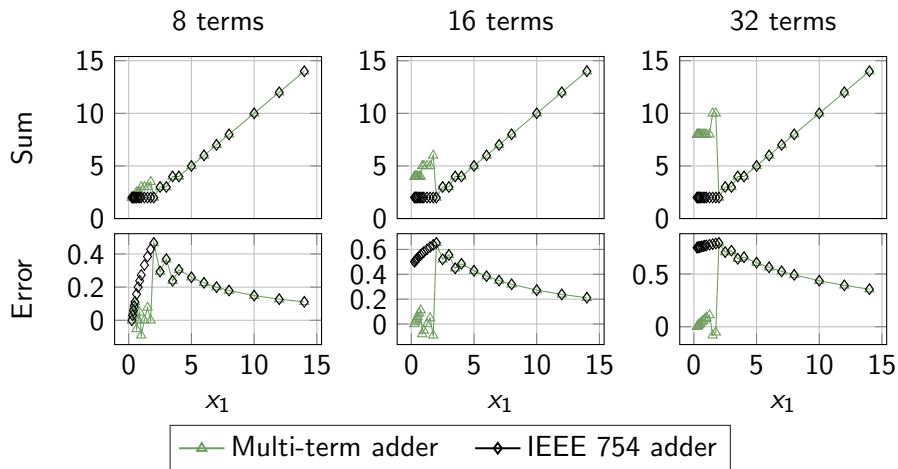
and

$$\text{fl}(\text{flr}(\cdots \text{flr}(x_1 + x_2) + \cdots) + x_n)$$

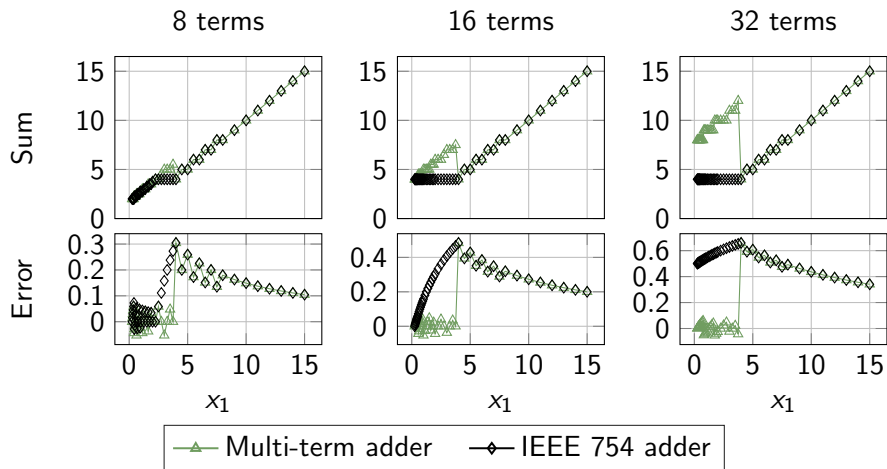
in three small floating-point systems:  $p = 3$ ,  $e_{max} = 3$ ;  $p = 4$ ,  $e_{max} = 3$ ; and  $p = 5$ ,  $e_{max} = 4$ .

- Set all  $x_i = 0.25$  and then vary  $x_1$  by changing it to the adjacent floating-point value towards  $+\infty$  until all representable values are covered.
- Each time we change  $x_1$  we sum the values  $x_i$  with IEEE 754 ops and with the Class IV adder.
- Relative error compared with the same sum performed in binary64 arithmetic.

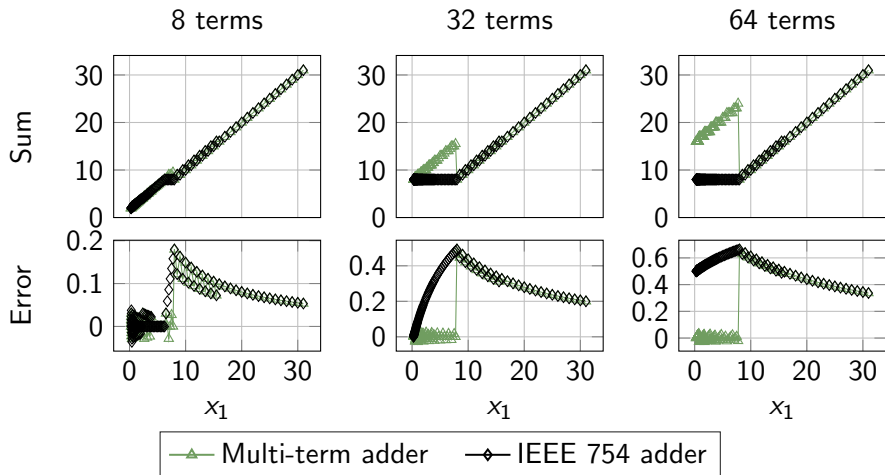
# Numerical experiments with $p = 3$ , $e_{max} = 3$



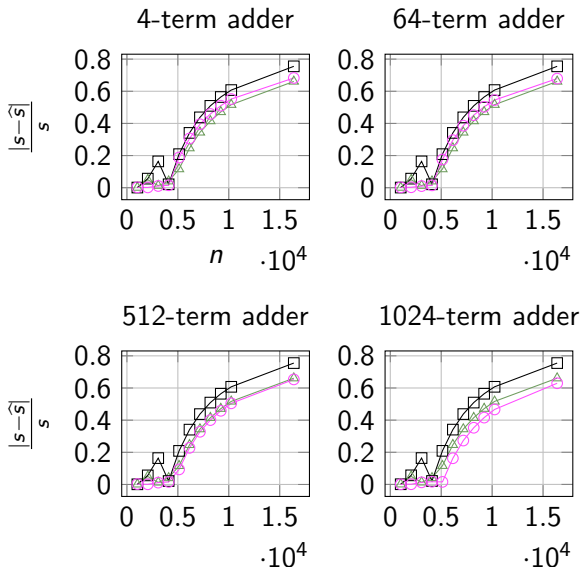
# Numerical experiments with $p = 4$ , $e_{max} = 3$



# Numerical experiments with $p = 5$ , $e_{max} = 4$



# Impact of addend ordering on accuracy (binary16)



—▲— Inc. order    —■— Dec. order    —○— Use of multi-term adder

## Example issues: square root

We may get into trouble if we use multi-term adders in computing

$$\sqrt{\sum_i^n a_i - \sum_j^k b_j} \text{ when we know } \sum_i^n a_i \geq \sum_j^k b_j.$$

For example in binary32, with  $a = [1, 1, 1, 1, 1, 1, 1, 16777216]$ ,  
 $b = [1, 1, 1, 1, 1, 1, 1, 16777214]$

- With a 8-term Class IV adder we get  $\sqrt{-4} = \text{NaN}$ .
- With IEEE 754 addition we can sort and avoid the NaN.

## Example issues: interval arithmetic

Compute interval of sum through the change of rounding modes (RD and RU) in Class IV multi-term adder.

Take  $a = [16777216, 1, 1, 1, 1, 1, 1, 1]$ ,  $b = [16777214, 1, 1, 1, 1, 1, 1, 1]$

Interval of the sum of  $a$  is  $[16777216, 16777230]$ .

Interval of the sum of  $b$  is  $[16777220, 16777222]$ .

Decreasing one addend, the lower end of interval shifts up; interval narrows due to precision growth.




# Classification of adders and their properties

- **Class I/II** (Kulisch, Tenca): associative, monotonic.
- **Class III** (chain of IEEE 754 two-term adders): not associative, monotonic.
- **Class IV** (limited precision, no intermediate normalization): associative, not monotonic

- Monotonicity important in bisection [Demmel, Dhillon, Ren, 1995]; solving quadratic equations [Higham, 2002]; mathematical functions.
- IEEE 754-2019 recommends *reduction operations*, but does not specify details; may consider revisiting for IEEE 754-2029.

## Paper

M. Mikaitis. *Monotonicity of Multi-Term Floating-Point Adders*. IEEE Trans. Comput. Feb. 2024. Early view. 

# Leeds Mathematical Software and Hardware Lab

New informal group in the School of Computing, Univ. Leeds.



Massimiliano Fasi  
Lecturer

Research&Teaching



Mantas Mikaitis  
Lecturer

Research&Teaching



- Focusing on computer arithmetic, numerical linear algebra, high-performance computing.
- Working with IEEE P3109 and IEEE 754-2029.
- Serving on PC committees of ARITH.
- Planning MSc module on computer arithmetic.
- PhD studentships available.

*Professor Nicholas J. Higham (1961–2024).*

I am grateful to Nicolas Brunie, Max Fasi, and three anonymous referees of IEEE TC for comments and suggestions.



N. J. Higham

Accuracy and Stability of Numerical Algorithms. 2nd edition  
SIAM. 2002



A. F. Tenca

Multi-operand floating-point addition  
19th IEEE Symp. Comput. Arithmetic. 2009



R. L. Ashenurst and N. Metropoli

Unnormalized floating point arithmetic  
J. ACM, 6. 1959.



M. Fasi, N. J. Higham, S. Pranesh, and M. Mikaitis

Numerical behavior of NVIDIA tensor cores  
PeerJ Comput. Sci., 7. 2021



M. Fasi and M. Mikaitis

CPFloat: A C library for simulating low-precision arithmetic  
*ACM Trans. Math. Software*, 49. 2023



J. W. Demmel, I. Dhillon, and H. Ren

On the correctness of some bisection-like parallel eigenvalue  
algorithms in floating point arithmetic  
*Electron. Trans. Numer. Anal.*, 3. 1995