Beating Posits at Their Own Game: Takum Arithmetic

Laslo Hunhold

Parallel and Distributed Systems Group
University of Cologne

20th February 2024

Introduction

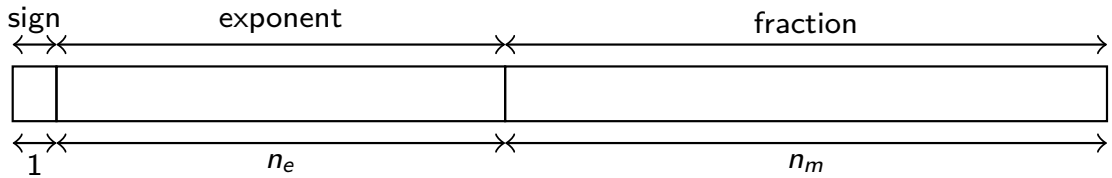
Gustafson Number System Criteria[1]

1. Distribution: The distribution of numbers within the system should accurately reflect those used in calculations, unique bit patterns.
2. Uniqueness: Only one possible bit pattern for each encoded value.
3. Generality: Should be defined for any bit pattern length.
4. Statelessness: The computation within the number system should be unaffected by any state other than the immediate input data.
5. Exactness: Mathematical operations should be as exact as possible.
6. Binary Monotonicity: The mapping of real numbers to bit patterns, interpreted as signed integers, should be monotonic.
7. Binary Negation: Negating the bit pattern as a signed integer should negate the represented real number, invariant for nonnumbers.
8. Flexibility: Conversion between lengths should be straightforward.
9. Absurdum Propagation: If an operation yields an absurdum ('NaR', not a real), it should propagate through all subsequent operations.
10. Implementation Simplicity: Both simple and efficient to implement in hardware (transistor count, energy efficiency, latency, throughput).

[1] John L. Gustafson. 'Every Bit Counts: Posit Arithmetic'. In preparation. 2024

Introduction

IEEE-754 Floating-Point Numbers



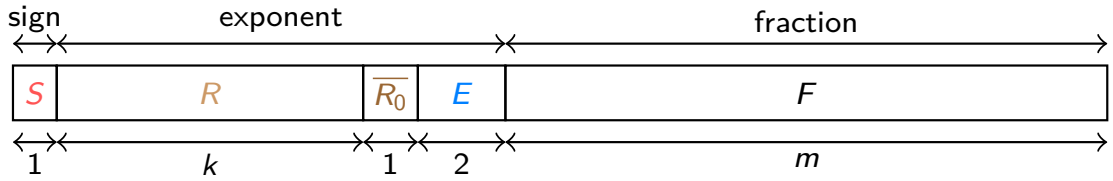
name	n	n_e	n_m	smallest	largest	notes
`float8`	8	4	3	$\approx 2.0 \times 10^{-3}$	$\approx 2.4 \times 10^{2}$	not standardized
`float16`	16	5	10	$\approx 6.0 \times 10^{-8}$	$\approx 6.6 \times 10^{4}$	
`bfloat16`	16	8	7	$\approx 1.2 \times 10^{-38}$	$\approx 3.4 \times 10^{38}$	no subnormals, [2]
`TF32`	19	8	10	$\approx 1.2 \times 10^{-38}$	$\approx 3.4 \times 10^{38}$	no subnormals, [3]
`float32`	32	8	23	$\approx 1.4 \times 10^{-45}$	$\approx 3.4 \times 10^{38}$	
`float64`	64	11	52	$\approx 4.9 \times 10^{-324}$	$\approx 1.8 \times 10^{308}$	
`float128`	128	15	112	$\approx 6.5 \times 10^{-4966}$	$\approx 1.2 \times 10^{4932}$	
`float256`	256	19	236	$\approx 2.2 \times 10^{-78\,984}$	$\approx 1.6 \times 10^{78\,913}$	

[2] Shibo Wang and Pankaj Kanwar. 'BFloat16: The secret to high performance on Cloud TPUs'. Aug. 2019.

[3] Paresh Kharya. 'TensorFloat-32 in the A100 GPU Accelerates AI Training, HPC up to 20x'. May 2020.

Introduction

Posit Arithmetic[4]



S : sign bit

$R := (R_{k-1}, \ldots, R_0)$: regime bits

$\overline{R_0}$: regime termination bit

$$r := \begin{cases} -k & R_0 = 0 \\ k-1 & R_0 = 1 \end{cases}$$: regime

$E := (E_1, E_0)$: exponent bits

$\tilde{e} := 2E_1 + E_0$: exponent

$e := (1-2S)(4r+e+S)$: 'actual' exponent

$m := n - k - 2 - 2$: fraction bit count

$F := (F_{m-1}, \ldots, F_0)$: fraction bits

$$f := 2^{-m} \sum_{i=0}^{m-1} F_i 2^i \in [0, 1)$$: fraction

$$p := \begin{cases} \begin{cases} 0 & S = 0 \\ \text{NaR} & S = 1 \end{cases} & R = \overline{R_0} = E = F = \mathbf{0} \\ [(1-3S) + f] \cdot 2^e & \text{otherwise} \end{cases}$$: value

Without loss of generality $S = 0$ and $R_0 = 1 \quad \Rightarrow \quad p \in [1, \infty)$ and $\boxed{e = 4(k-1) + \tilde{e}}$.

[4] John L. Gustafson and Isaac Yonemoto. 'Beating Floating Point at Its Own Game: Posit Arithmetic'. In: Supercomputing Frontiers and Innovations 4.2 (June 2017), pp. 71-86.

First Idea

Improve Posit Exponent Coding Efficiency Only

posit	$k-1$	$\tilde{e}$	e	f	p
0100011	0	0	0	0.75	$1.75 \times 2^{0} = 1.75 \times 10^{0}$
01111001	3	1	13	0.0	$1.00 \times 2^{13} \approx 8.2 \times 10^{3}$
011111111111101101	11	3	47	0.25	$1.25 \times 2^{47} \approx 1.8 \times 10^{14}$
01111111111111111010	15	2	62	0.0	$1.00 \times 2^{62} \approx 4.6 \times 10^{18}$

- Small magnitude exponents are encoded efficiently
- Large magnitude exponents exhibit low entropy ($\rightarrow$ wasted bits)

Idea:

- Encode number of exponent bits as $k+2$
- Don't encode leading 1-bit for $k > 1$, encode three bits for $k = 1$ (allows encoding 0 without shift, more efficient)
- Tradeoff: 1 more bit needed for exponents 0, 1, 2 and 3

modified posit	k	e-bits	e	f	p
01000011	1	000	0	0.75	$1.75 \times 2^{0} = 1.75 \times 10^{0}$
0110101	2	$\underline{1}101$	13	0.0	$1.00 \times 2^{13} \approx 8.2 \times 10^{3}$
0111100111101	4	$\underline{1}01111$	47	0.25	$1.25 \times 2^{47} \approx 1.8 \times 10^{14}$
01111011110	4	$\underline{1}11110$	62	0.0	$1.00 \times 2^{62} \approx 4.6 \times 10^{18}$

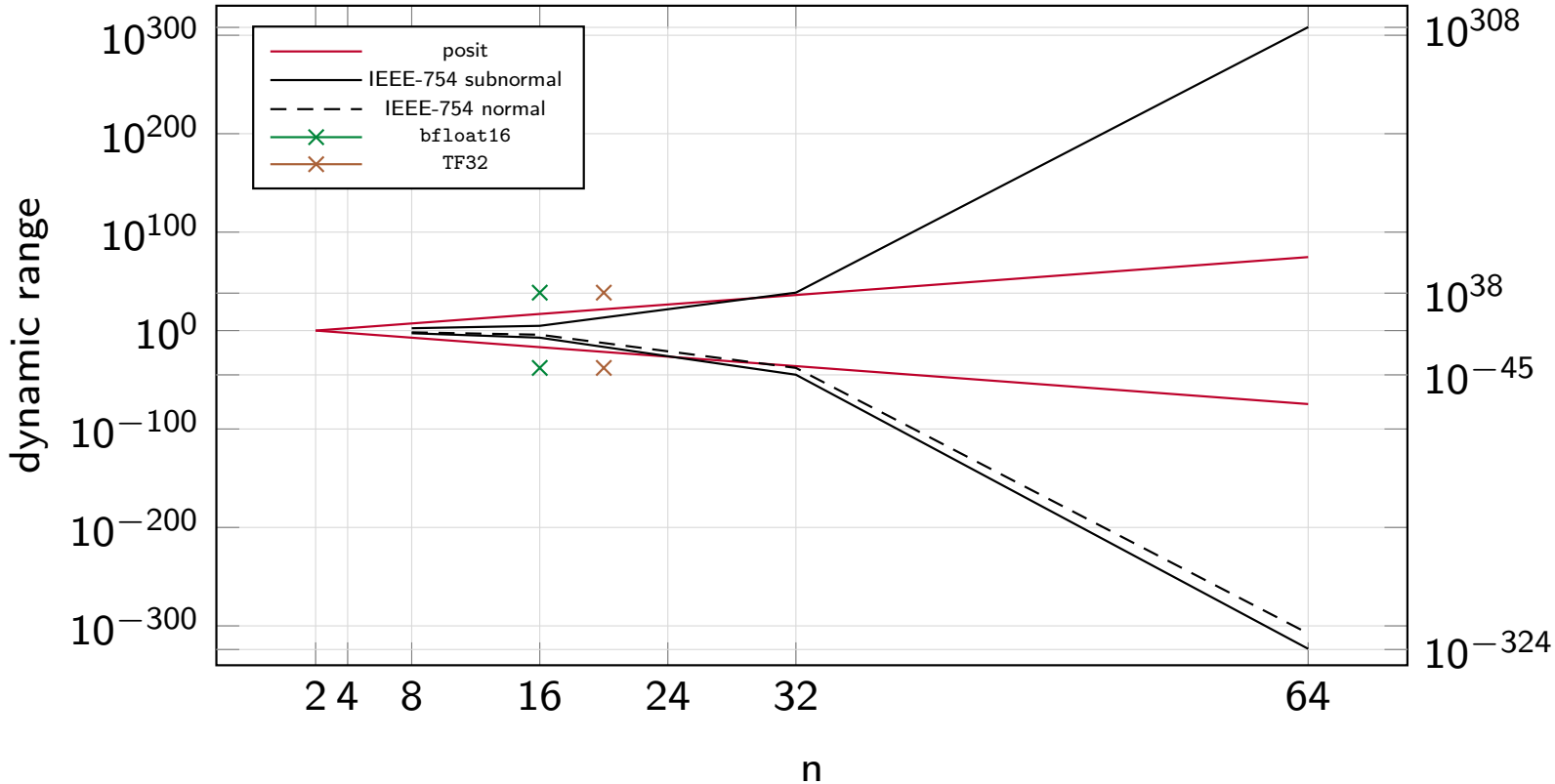
Dynamic Range Criteria

- Problem: Excessive dynamic range $2^{\pm 256}, 2^{\pm 65536}, 2^{\pm 4294967296}$ for $n = 8, 16, 32$ ($\rightarrow$ still wasted bits for numbers that will never be used)
- At some point, additional bits should only contribute to precision, not dynamic range
- Add two new 'Dynamic Range Criteria' for number systems:
  11. The magnitudes of the largest and smallest representable exponent should be equal.
  12. The dynamic range should be reasonably bounded on both ends as the bit pattern length approaches infinity.
- IEEE-754 floating-point numbers violate both conditions, posits violate the latter.
- Goal: Find a new format with reasonably bounded dynamic range

What is a 'reasonable bound' on the dynamic range?

Dynamic Range Bounds

Current Number Systems



- `float16` has insufficient dynamic range $\approx 2^{-24} \ldots 2^{16}$
- `float32` (without subnormals), `bfloat16` and `TF32` have 'proven' dynamic range $\approx 2^{-126} \ldots 2^{128}$, almost equal to `posit32`'s
- `float64` has excessive dynamic range $\approx 2^{-1074} \ldots 2^{1023}$
- Empirical dynamic range target: Between $2^{\pm 128}$ and $2^{\pm 1024}$

Dynamic Range Bounds

Physical Constants

Extended from the posit benchmark in Florent de Dinechin et al. 'Posits: The Good, the Bad and the Ugly'. In: CoNGA'19. Singapore, Singapore: Association for Computing Machinery, 2019. DOI: 10.1145/3316279.3316285:

name	symbol	value (without units)	$\log_2(\cdot) \approx$
PLANCK constant	h	$6.62607015 \times 10^{-34}$	-110.22
BOLTZMANN constant	k	1.380649×10^{-23}	-75.94
elementary charge	e	$1.602176634 \times 10^{-19}$	-62.44
speed of light	c	2.99792458×10^{8}	28.16
caesium standard	$\Delta\nu$	$9.192631770 \times 10^{9}$	33.10
AVOGADRO constant	N_A	$6.02214076 \times 10^{23}$	78.99
cosmological constant	Λ	1.1056×10^{-52}	-172.60
mass of the universe	M	1.5×10^{53}	176.65

Physical constants dynamic range target: At least $2^{\pm 177}$

Dynamic Range Bounds

Binary Saturation and Discussion

- Assumption: The upper exponent bound is a saturated binary integer ($2^k - 1$ with $k \in \mathbb{N}_0$, i.e. $0, 1, 3, 7, 15, 31, 63, 127, 255, 511, \ldots$).
- For tapered encoding: Want saturated binary integer whose bit-length is also a saturated binary integer ($2^{2^k} - 1$ with $k \in \mathbb{N}_0$, i.e. $1, 3, 15, 255, 65535, \ldots$)
- Empirical dynamic range target: Between $2^{\pm 128}$ and $2^{\pm 1024}$
- Physical constants dynamic range target: At least $2^{\pm 177}$

$$\Rightarrow \text{Target dynamic range: } 2^{\pm 255} \approx 10^{\pm 77}$$

- Roughly twice the dynamic range as `float32`, `bfloat16` and `TF32`
- Leaves arithmetic-headroom for SI-constants (squaring, etc.)
- Desired saturation properties

Wasted Bits in IEEE-754 Floating-Point Numbers

- Assuming target dynamic range $2^{\pm 255}$ and redundant NaN-representations, how many bits are wasted in IEEE-754 floating-point numbers?

name	n	n_e	n_m	smallest	largest	wasted/%
`float8`	8	4	3	$\approx 2.0 \times 10^{-3}$	$\approx 2.4 \times 10^{2}$	5.08
`float16`	16	5	10	$\approx 6.0 \times 10^{-8}$	$\approx 6.6 \times 10^{4}$	3.12
`bfloat16`	16	8	7	$\approx 1.2 \times 10^{-38}$	$\approx 3.4 \times 10^{38}$	0.78
`TF32`	19	8	10	$\approx 1.2 \times 10^{-38}$	$\approx 3.4 \times 10^{38}$	0.78
`float32`	32	8	23	$\approx 1.4 \times 10^{-45}$	$\approx 3.4 \times 10^{38}$	0.39
`float64`	64	11	52	$\approx 4.9 \times 10^{-324}$	$\approx 1.8 \times 10^{308}$	75.24
`float128`	128	15	112	$\approx 6.5 \times 10^{-4966}$	$\approx 1.2 \times 10^{4932}$	98.45
`float256`	256	19	236	$\approx 2.2 \times 10^{-78\,984}$	$\approx 1.6 \times 10^{78\,913}$	99.90

- Posits waste $\begin{cases} 0\% & n \leq 64 \\ \approx 1.08 \times 10^{-17}\% & n \geq 65. \end{cases}$ of their binary patterns.

Takum Arithmetic

Idea

- Icelandic ‘takmarkað umfang’, meaning ‘limited range’
- Pick up previous modified posit idea, but with fixed-size 3-bit regime (values 0 to 7) instead of prefix code (less overhead)
- Regime value is the number of exponent bits minus 1.
- Exponent bits follow regime bits, implicit leading 1-bit except for regime value 0.

value	0	1	2	3	4	...	255
value bits	0	1	10	11	100	...	11111111
encoding	0000	0001	0010	0011	01000	...	1111111111

- Need a separate direction bit D to store information previously covered by R_0
- Interlude: Comparison for value 255 (posit → modified posit → takum):

11111111111111111111111111111111111111111111111111111111111111011 (67)

↓

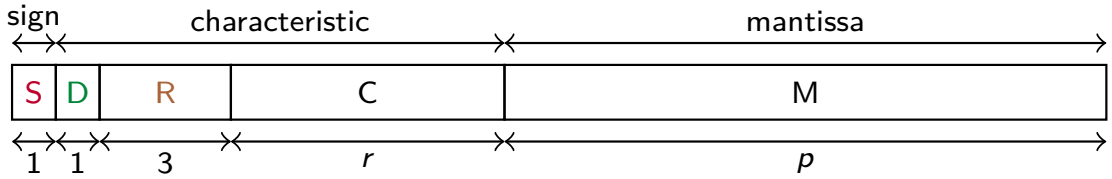
11111101111111 (14)

↓

11111111111 (11)

Takum Arithmetic

Definition



$$
\begin{aligned}
&S && : \text{sign bit}\\
&D && : \text{exponent direction bit}\\
&R := (R_2, R_1, R_0) && : \text{exponent regime bits}\\
&r := \begin{cases} 7 - (4R_2 + 2R_1 + R_0) & D = 0 \\ 4R_2 + 2R_1 + R_0 & D = 1 \end{cases} \in \{0,\ldots,7\} && : \text{exponent regime}\\
&\ell := \max(1, r) \in \{1, \ldots, 7\} && : \text{exponent amplitude bit count}\\
&A := (A_{\ell-1}, \ldots, A_0) && : \text{exponent amplitude bits}\\
&a := \begin{cases} A_0 & r = 0 \\ 2^r + \sum_{i=0}^{r-1} A_i 2^i & r \geq 1 \end{cases} && : \text{exponent amplitude}\\
&b := \begin{cases} \begin{cases} 2 & r = 0 \\ 3 \cdot 2^r & r \geq 1 \end{cases} & D = 0 \\ 0 & D = 1 \end{cases} && : \text{exponent bias}\\
&e := (1 - 2S)(a - b + S) && : \text{exponent}\\
&m := n - \ell - 5 \in \{n - 12, \ldots, n - 6\} && : \text{fraction bit count}\\
&F := (F_{m-1}, \ldots, F_0) && : \text{fraction bits}\\
&f := 2^{-m} \sum_{i=0}^{m-1} F_i 2^i \in [0, 1) && : \text{fraction}\\
&v := \begin{cases} \begin{cases} 0 & S = 0 \\ \mathrm{NaR} & S = 1 \end{cases} & D = R = A = F = \mathbf{0} \\ [(1 - 3S) + f] \cdot 2^e & \text{otherwise.} \end{cases} && : \text{value}
\end{aligned}
$$

Takum Arithmetic

Properties

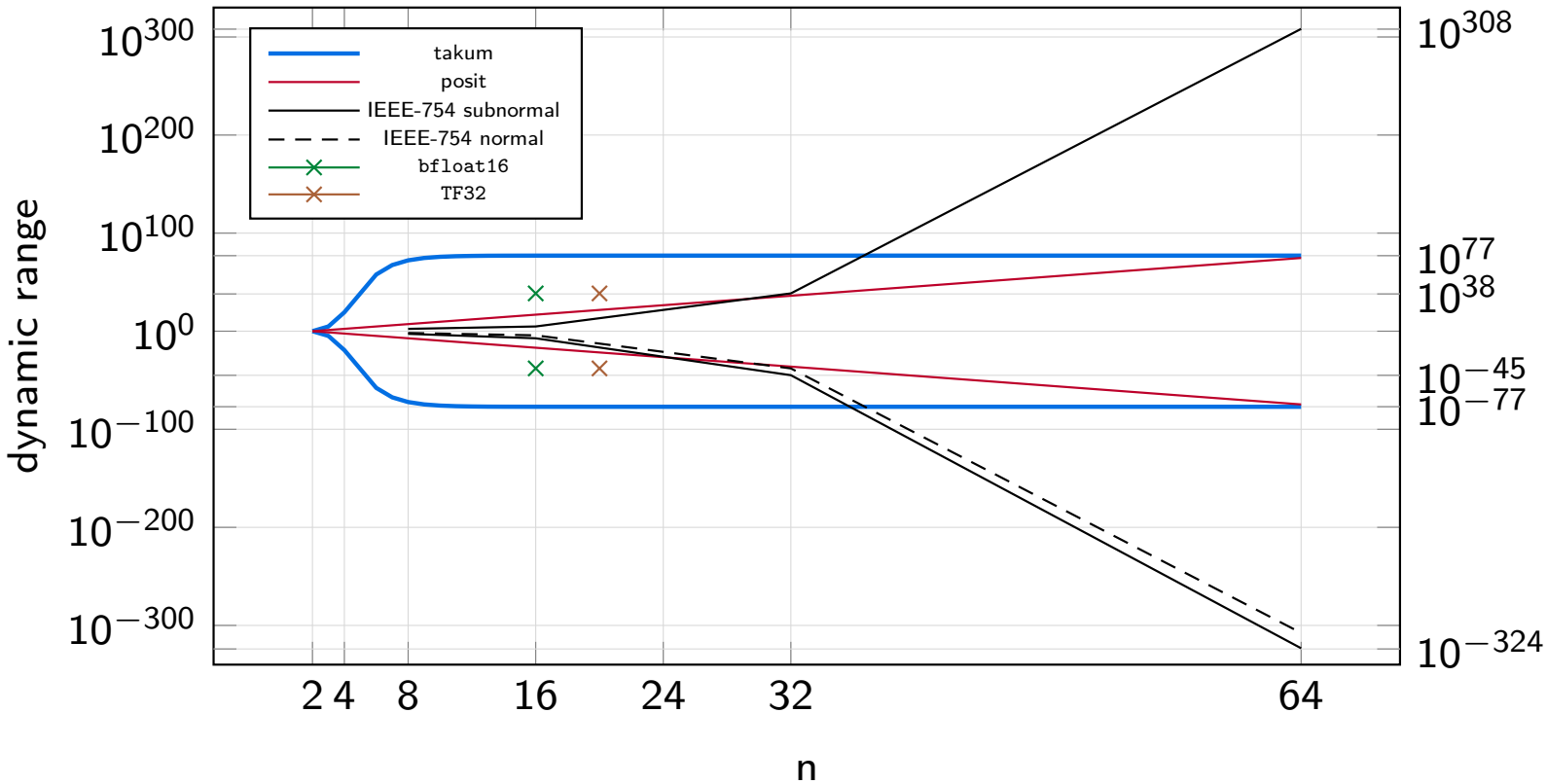
- $n - 12$ fraction bits are guaranteed (posits guarantee none), any extension beyond 12 bits only adds precision (not dynamic range)

  $$0100001101010011010\ldots$$
  $$\vdots$$
  $$0111111111111010011\ldots$$

- All 12 number system criteria satisfied (including binary monotonicity and negation)
- Hardware implementation
  - Exponent parsing the same for all precisions (as limited to first 12 bits)
  - Very small and efficient mask/shift/bias-LUTs for fast exponent and fraction bit extraction
- Nomenclature distinguishes between exponent amplitude a and exponent value e
- New colour scheme with higher contrast and uniform lightness/saturation
  (sign, regime, regime terminator, exponent) $\rightarrow$
  (sign, direction, regime, amplitude)

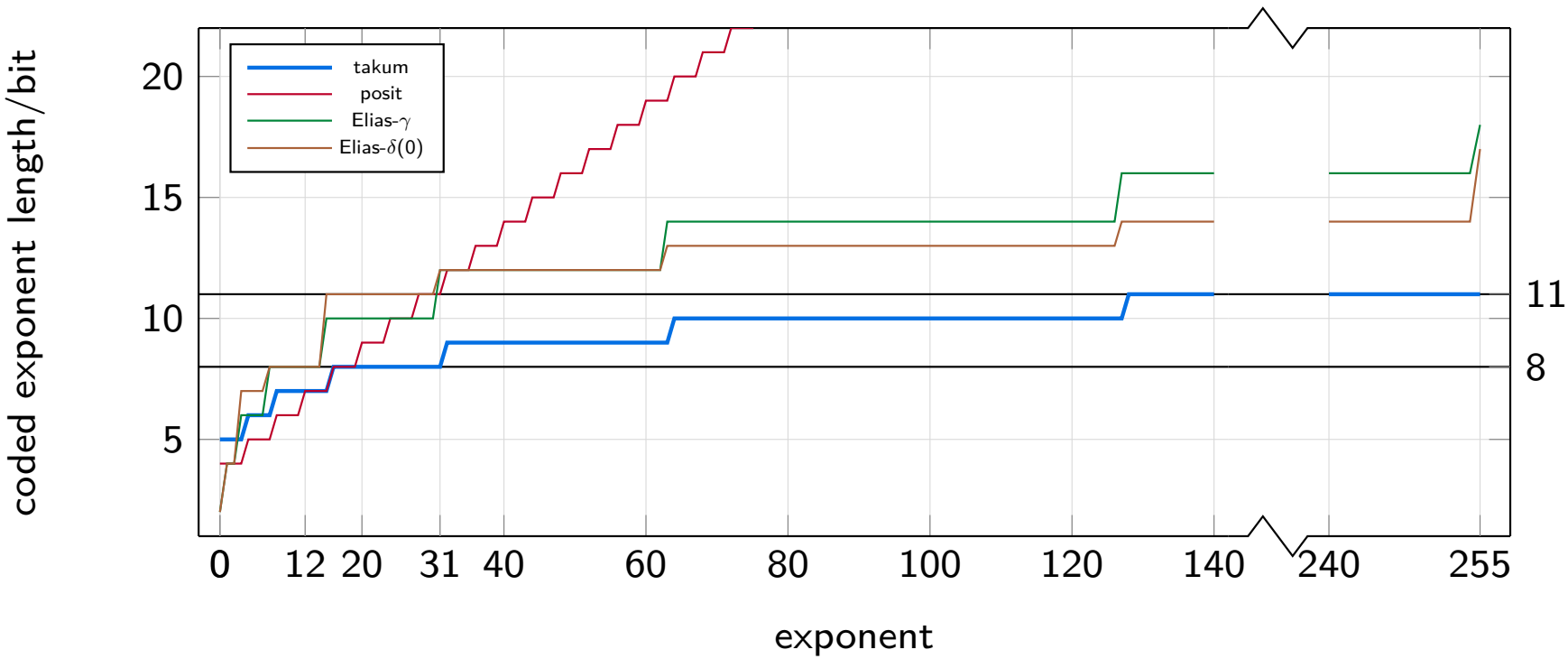
Evaluation

Dynamic Range



Evaluation

Exponent Coding Efficiency



Evaluation

Small Physical Constants Benchmark

name	PLANCK constant	BOLTZMANN constant	elementary charge
symbol	h	k	e
value	$6.62607015 \times 10^{-34}$	1.380649×10^{-23}	$1.602176634 \times 10^{-19}$
float8	0	0	0
posit8	$5.96046448 \times 10^{-8}$	5.960464×10^{-8}	$5.960464478 \times 10^{-8}$
takum8	$1.92592994 \times 10^{-34}$	2.117582×10^{-22}	$5.421010862 \times 10^{-20}$
float16	0	0	0
bfloat16	$6.62038418 \times 10^{-34}$	1.385528×10^{-23}	$1.600892270 \times 10^{-19}$
posit16	$1.38777878 \times 10^{-17}$	1.387779×10^{-17}	$1.387778781 \times 10^{-17}$
takum16	$6.62038418 \times 10^{-34}$	1.364848×10^{-23}	$1.609362600 \times 10^{-19}$
TF32	$6.62790735 \times 10^{-34}$	1.380358×10^{-23}	$1.601951062 \times 10^{-19}$
posit19	$3.38813179 \times 10^{-21}$	3.388132×10^{-21}	$2.168404345 \times 10^{-19}$
takum19	$6.62038418 \times 10^{-34}$	1.380358×10^{-23}	$1.603009853 \times 10^{-19}$
float32	$6.62607018 \times 10^{-34}$	1.380649×10^{-23}	$1.602176598 \times 10^{-19}$
posit32	$7.70371978 \times 10^{-34}$	1.380358×10^{-23}	$1.602215759 \times 10^{-19}$
takum32	$6.62607064 \times 10^{-34}$	1.380649×10^{-23}	$1.602176727 \times 10^{-19}$

Evaluation

Large Physical Constants Benchmark

name	speed of light	caesium standard	AVOGADRO constant
symbol	c	$\Delta\nu$	N_A
value	2.99792458×10^{8}	$9.192631770 \times 10^{9}$	$6.02214076 \times 10^{23}$
float8	∞	∞	∞
posit8	1.67772160×10^{7}	$1.677721600 \times 10^{7}$	1.67772160×10^{7}
takum8	2.68435456×10^{8}	$4.294967296 \times 10^{9}$	$1.20892582 \times 10^{24}$
float16	∞	∞	∞
bfloat16	2.99892736×10^{8}	$9.193914368 \times 10^{9}$	$6.02101727 \times 10^{23}$
posit16	3.01989888×10^{8}	$9.663676416 \times 10^{9}$	$7.20575940 \times 10^{16}$
takum16	2.99892736×10^{8}	$9.126805504 \times 10^{9}$	$6.04462910 \times 10^{23}$
TF32	2.99892736×10^{8}	$9.193914368 \times 10^{9}$	$6.02101727 \times 10^{23}$
posit19	2.99892736×10^{8}	$9.126805504 \times 10^{9}$	$2.95147905 \times 10^{20}$
takum19	2.99892736×10^{8}	$9.193914368 \times 10^{9}$	$6.02101727 \times 10^{23}$
float32	2.99792448×10^{8}	$9.192631296 \times 10^{9}$	$6.02214064 \times 10^{23}$
posit32	2.99792384×10^{8}	$9.192636416 \times 10^{9}$	$6.02101727 \times 10^{23}$
takum32	2.99792448×10^{8}	$9.192632320 \times 10^{9}$	$6.02214136 \times 10^{23}$

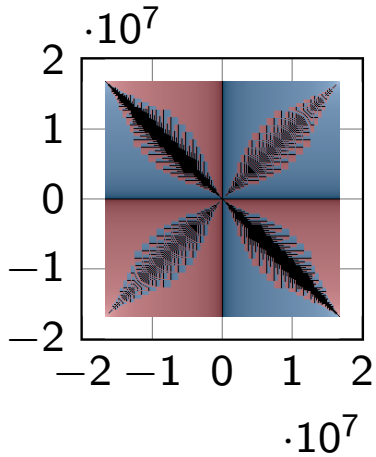
Evaluation

Extreme Physical Constants Benchmark

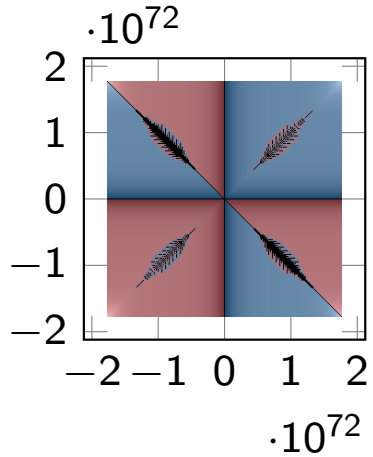
name	cosmological constant	mass of the universe
symbol	Λ	M
value	1.1056×10^{-52}	1.5×10^{53}
`float8`	0	∞
`posit8`	5.9605×10^{-8}	1.7×10^{7}
`takum8`	1.0440×10^{-53}	9.6×10^{52}
`float16`	0	∞
`bfloat16`	0	∞
`posit16`	1.3878×10^{-17}	7.2×10^{16}
`takum16`	1.0963×10^{-52}	1.5×10^{53}
`TF32`	0	∞
`posit19`	3.3881×10^{-21}	3.0×10^{20}
`takum19`	1.1028×10^{-52}	1.5×10^{53}
`float32`	0	∞
`posit32`	7.5232×10^{-37}	1.3×10^{36}
`takum32`	1.1056×10^{-52}	1.5×10^{53}

Evaluation

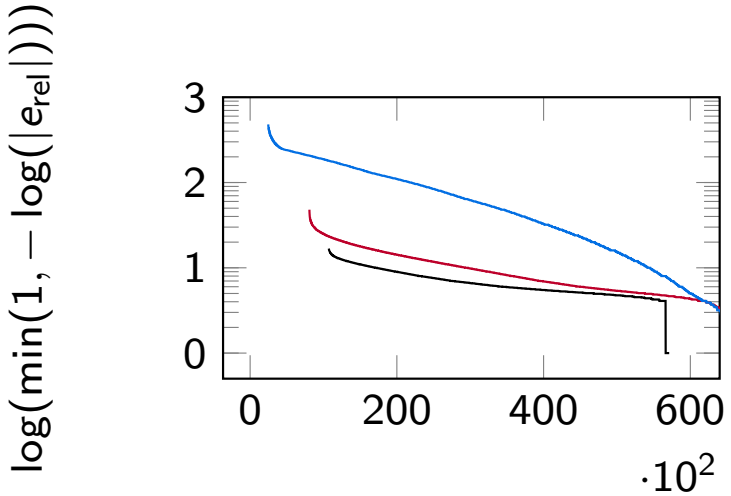
Binary Closure



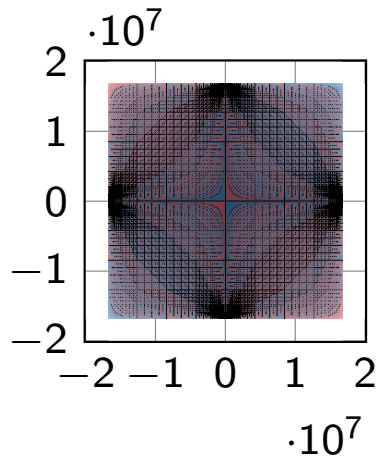
(a) posit8 addition



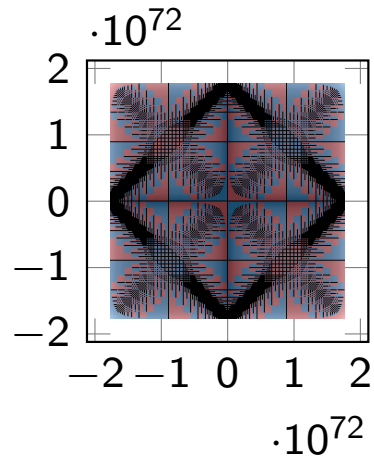
(b) takum8 addition



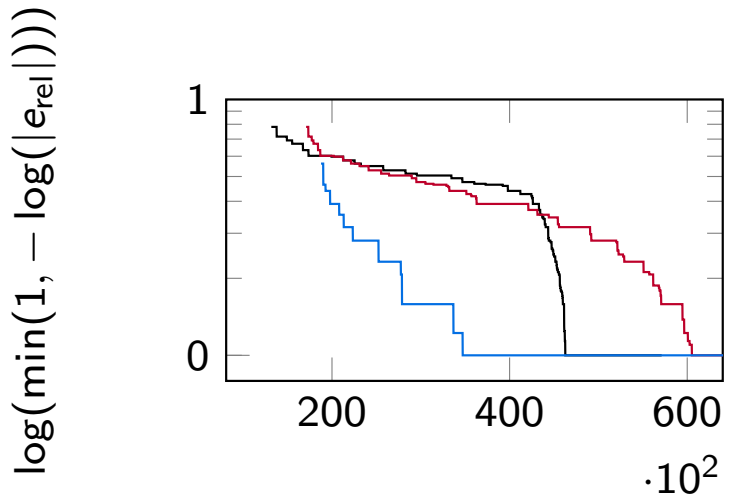
(c) sorted relative precision vs. float8



(d) posit8 mult.



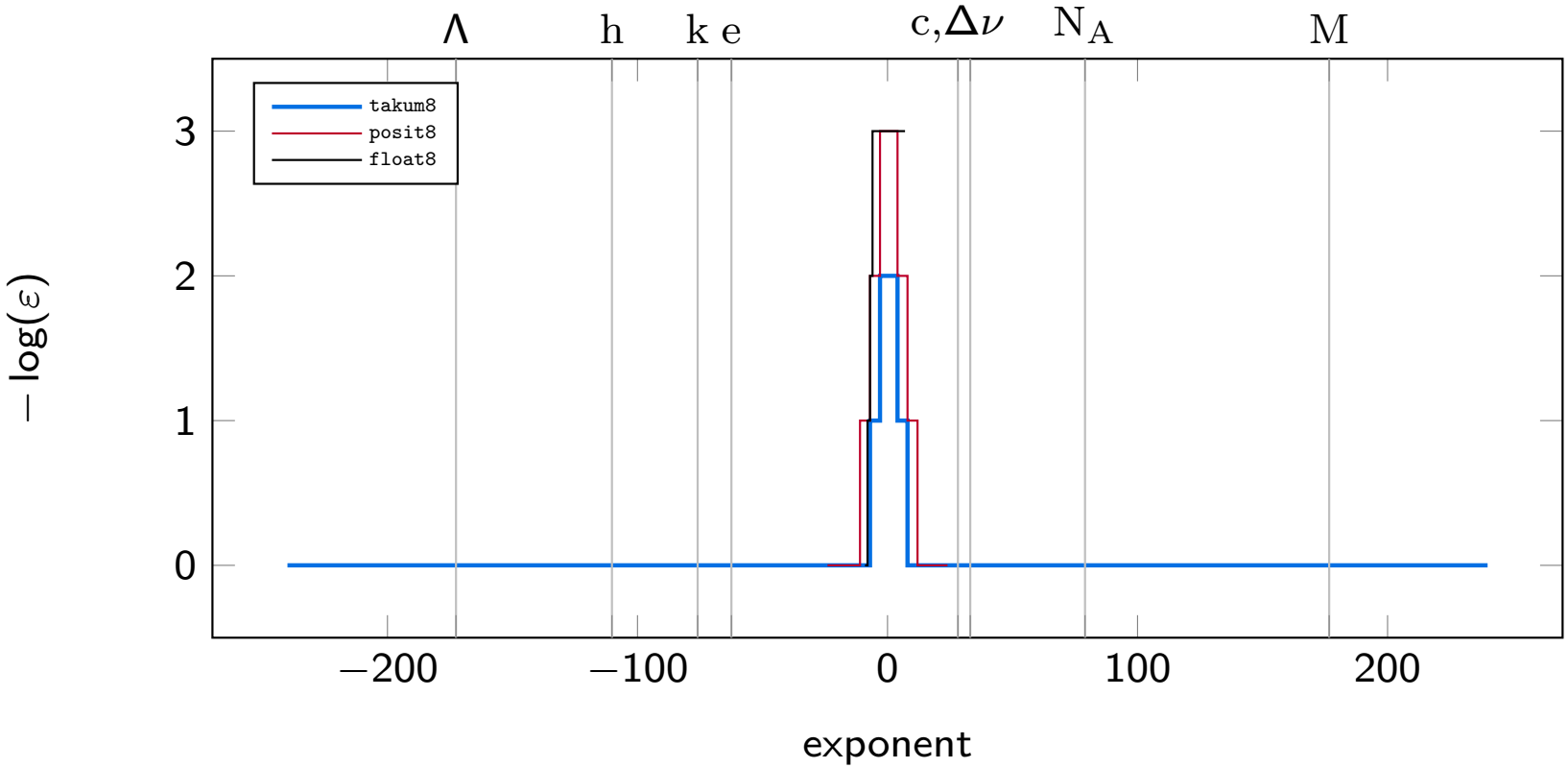
(e) takum8 mult.



(f) sorted relative precision vs. float8

Evaluation

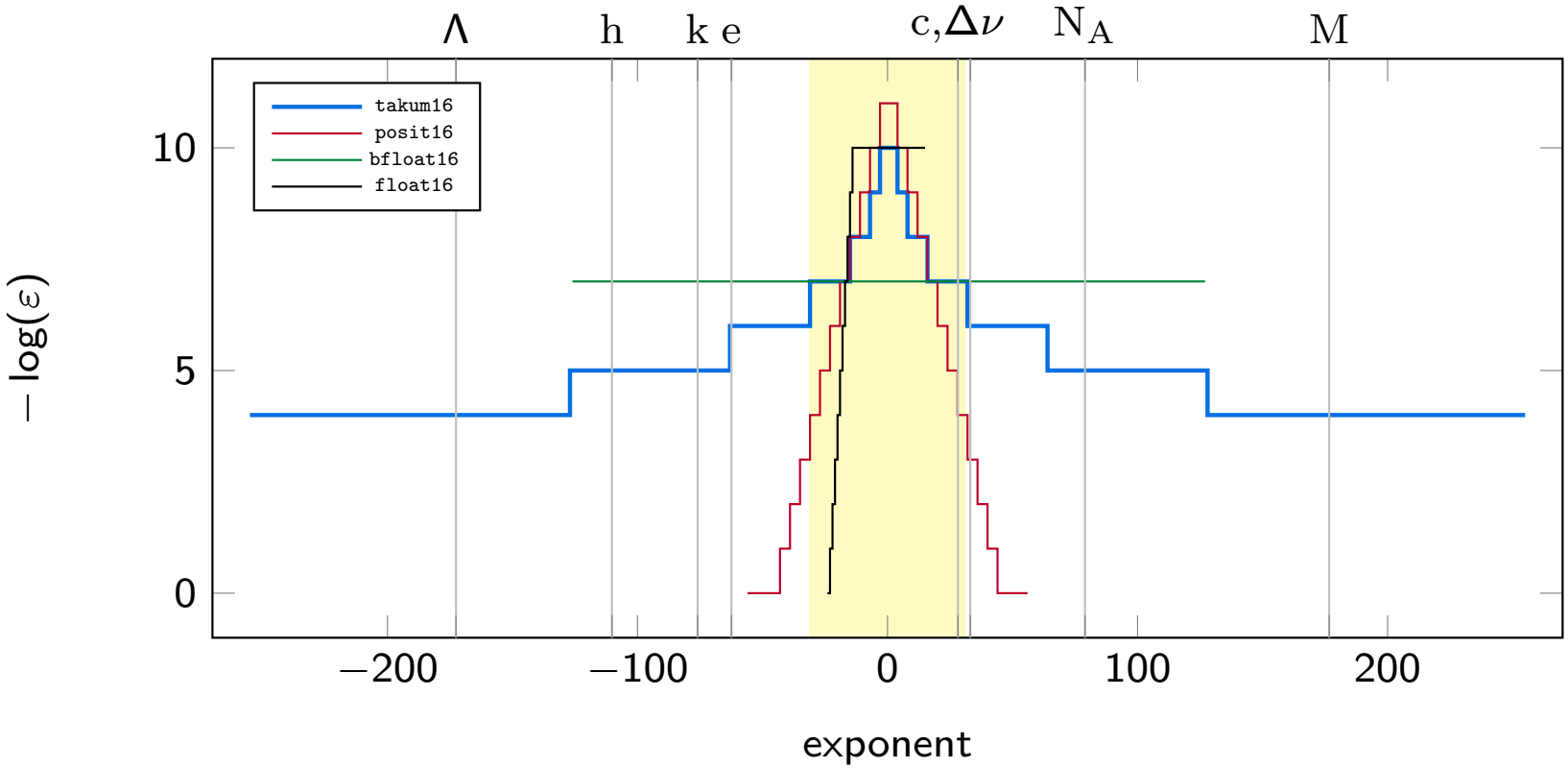
Relative Error (8 bits)



- Almost full dynamic range at 8 bits, covers all benchmark constants

Evaluation

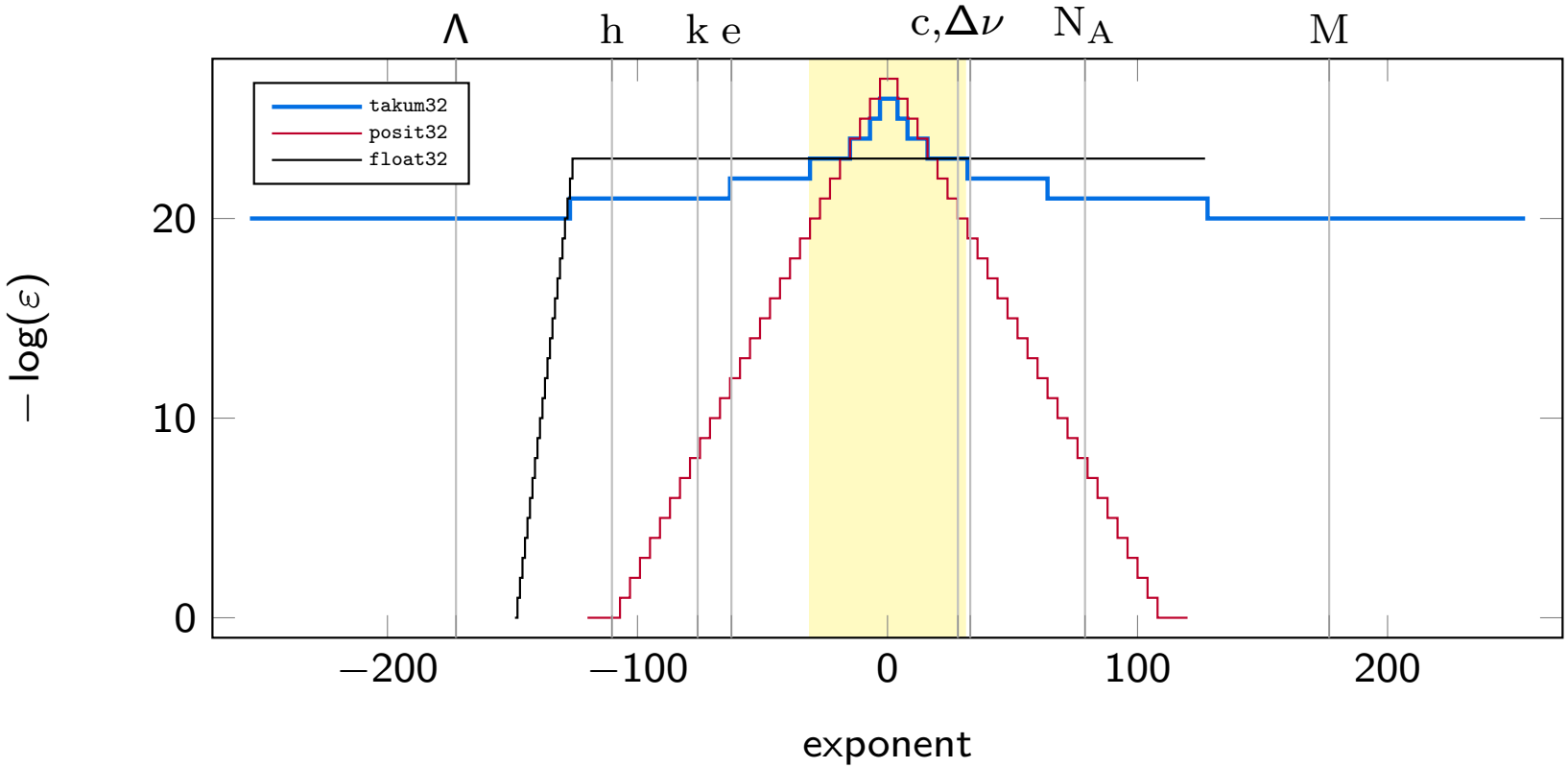
Relative Error (16 bits)



- Golden zone relative to `bfloat16` wider than posit's
- Full dynamic range; twice the range as `bfloat16`

Evaluation

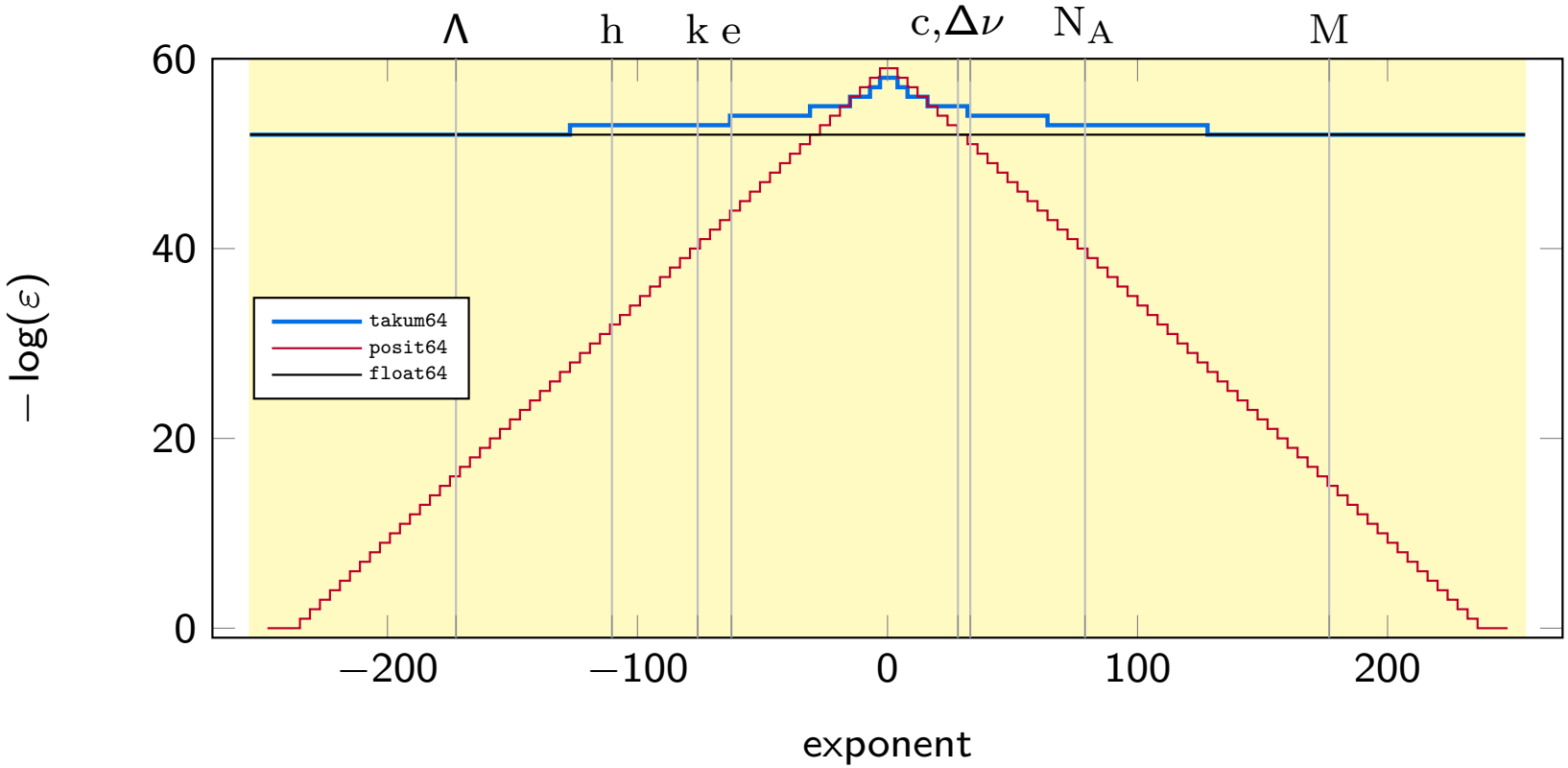
Relative Error (32 bits)



- Golden zone much larger than posit's
- At most 3 fewer fraction bits than `float32`

Evaluation

Relative Error (64 bits)



- Golden zone covers entire dynamic range
- `takum64` has up to 6 more fraction bits than `float64` in the fovea, never fewer (lossless conversion within dynamic range)
- `takum64` has 52 fraction bits where `posit64` has none

Conclusion and Outlook

- Discussed admissible dynamic ranges for general purpose number systems
- Presented takum as a new posit-like number system with bounded dynamic range
- Outlook: C implementation, benchmarks, numerical analysis, fault tolerance
- https://takum.org/