It's Time for Unums – an Alternative to IEEE 754 Floats and Doubles

Thomas Risse
Institute of Informatics and Automation
Hochschule Bremen, City University of Applied Sciences, Germany
e-mail risse@hs-bremen.de

***Abstract*—In the eighties of the last century standardization of representation of and arithmetic with floating point numbers by IEEE 754 promised a reliable and efficient way to handle scientific computing independently of the hardware. However, IEEE 754 implied lots of numerical anomalies which seemingly had to be accepted fatefully. Now, Unums constitute a self-descriptive representation of floating point numbers which is closed under the basic arithmetic operations, which ends rounding errors and which allows for efficient scientific computing taking into consideration todays technological preconditions.**

I. INTRODUCTION

In the eighties of the last century it was a milestone to standardize representation of and arithmetic with floating point numbers. With the standard IEEE 754 of 1985, amended 2008 computation with floating point numbers, i.e. any scientific computation, should no longer be dependent on a specific hardware. However, IEEE 754 implies lots of numerical anomalies; it hides inaccuracies by rounding and truncation as well as underflow and overflow; it presumes given numbers of bits for mantissa and exponent in fixed formats.

Now, Unums [2] cover the whole real line $[-\infty, \infty]$ by alternating between exact (rational) numbers and open intervals of reals between these exact numbers. Thus, firstly, rounding, underflow, overflow are avoided. Secondly, a clever application of interval arithmetic together with built in operations like an exact dot product render every mathematical operation on Unums provably true.
Thirdly, Unums with varying numbers of bits for the mantissa and for the exponent on average require less memory space and bus bandwidth. Fixed format representation of Unums in 64 bit registers allow for fast arithmetic operations on Unums.
Lastly, with Unums different algorithms for typical numerical problems can be designed which can utilize the multitude of available floating point processors in parallel.

II. IEEE 754 AND ITS DEFICIENCIES

Mathematical properties of basic operations in the field of rational numbers like associativity $(-10^{16} + 10^{16}) + 1 = 1 \neq 0 = -10^{16} + 10^{16} = -10^{16} + (10^{16} + 1)$ or distributivity are no longer guaranteed when computing with IEEE 754 floating point numbers. This is maybe the severest deficiency because it prevents simplest parallelization. But there are more deficiencies that come with IEEE 754.

A. IEEE 754 computations are not trustworthy!

There is no indication whether or not a computation with IEEE 754 is exact even if all arguments are. E.g. 1 and 10 are integers which in IEEE 754 are represented exactly, but $\frac{1}{10}$ has a periodic binary representation and thus cannot be represented exactly with finite many bits. There is no indication whether some operation yields an exact result or not. IEEE 754 computations may violate mathematical laws like associativity or distributivity. With IEEE 754 floats $a + b == a$ not necessarily implies $b == 0$ but $|b| \ll |a|$ only.

B. Underflow and Overflow

Without notice, numbers greater than the biggest representable rational number, maxreal are represented by plus infinity, numbers smaller than –maxreal by minus infinity. Similarly, non-zero numbers very close to zero, i.e. those in (-smallestsubnormal,smallestsubnormal) are represented by zero, again without notice.

C. Fixed Number of Bits for Mantissa and Exponent

Independent of the requirements of a problem IEEE 754 specifies the number of bits for mantissa and exponent for e.g. single precision (floats in four bytes) and double precision (doubles in eight bytes) floating point numbers. (To be on the safe side usually formats with too many bits are chosen. Also, NaN has wastefully many representations in IEEE 754.) All these formats have been specified quite arbitrarily. There is no way to let the computer adjust these numbers dynamically.

D. Cancellation, Truncation, Rounding

Because cancellation the function evaluations of $f(x) = (x-1)^n$ and the mathematically identical $g(x) = \sum_{k=0}^{n} \binom{n}{k} (-1)^{n-k} x^k$ differ significantly for arguments close to 1.
For example, computing iterated square roots of 2 will eventually lead to $\sqrt[2^n]{2} = 1$.
There are quite lot examples where errors caused by IEEE 754 arithmetic had catastrophic consequences [1], [2].

E. Anomalies

There is an abundance of anomalies occurring when doing arithmetic with IEEE 754 floats: absurd erroneous solutions to ill conditioned linear equations (Bailey's nightmare in [2]), evaluating multivariate functions (Rumps royal pain in [2] or Kahans examples in [2]), ignoring discontinuities (Kahan's smooth surprise in [2]), or divergence instead of convergence to some fix-point of

a recursive sequence. To give only one example, define $H(x) = E(Q(x)^2)$ with $E(z) = \frac{e^z-1}{z}$ where $E(0) := 1$ and $Q(y) = \left|x - \sqrt{x^2+1}\right| - \frac{1}{x+\sqrt{x^2+1}}$. Then for IEEE 754 floats, doubles and even extended doubles with 128 bits $H(\vec{x})$ for $\vec{x} = (15.0,16.0,17.0,9999.0)$ returns $(0,0,0,0)$ whereas the correct result is $(1,1,1,1)$.

III. UNUMS

As IEEE 754 Unums represent some rationals r by the combination of sign, biased exponent and mantissa, called *fraction*: $r = \pm fraction\ 2^{exponent}$. In addition to these three fields there is the so called *ubit* indicating whether the number is exact or whether it represents the open interval $(r, r + ulp)$ where ulp is the *Unit in the Last Place*, i.e. the difference between r and the next bigger exactly representable Unum. So Unums are either exact or open real intervals. E.g. $(\text{maxreal}, \infty)$ is just maxreal with the ubit set and $\pm\infty$ are represented by their IEEE 754 equivalents. So Unums cover $[-\infty, \infty] \supset \mathbb{R}$. By the way, with Unums there are only two representations of NaN, quiet and signaling, instead of about $2^{24} = 2^4(2^{10})^2 \approx 16(10^3)^2 = 16 \cdot 10^6$ in case of IEEE 754 floats and $2^{53} = 8(2^{10})^5 \approx 8(10^3)^5 = 8 \cdot 10^{15}$ in case of IEEE 754 doubles by allowing for any sign and mantissa.

Define *Ubounds* to be either inexact Unums or pairs of Unums. Then the set of Ubounds obviously is closed under addition, subtraction, multiplications and division.

At last, adding two more fields *esizesize* and *fsizesize* allows to specify the number of exponent bits and the number of fraction bits. (The length of these two additional fields is fixed within a so called *environment*.) These two fields render the format of Unums variable and self-descriptive. This variable length format (like strings) allows for efficient storage and transport. In the processor, Unums are unpacked into fixed length 64bit registers with fast arithmetic. So, Unums take today's technological constraints into consideration: compared to the technology of the 1980s, nowadays to transport data takes much more energy than to do floating point arithmetic [2]:

Operation	**Energy**	**Time**
64 bit multiply-add	200 pJ	1 nsec
Read 64 bit from cache	800 pJ	3 nsec
Move 64 bit across chip	2000 pJ	5 nsec
Execute an instruction	7500 pJ	1 nsec
Read 64 bits from DRAM	12000 pJ	70 nsec

It shows that the memory savings more than compensate for the overhead consisting of ubit, esizesize and fsizesize.

IV. FUSED OPERATIONS, MATH LIBRARY

There are built-in functions to compute x^y (which is an algebraic function for any rational y, i.e. for any Unum y), or to compute the dot product exactly (guaranteeing associativity of addition), to compute products (guaranteeing associativity of multiplication) and the like. And, there is a math library, e.g. for all elementary mathematical functions and their inverses.

V. UBOXES AND ALGORITHMS WITH UNUMS

So called *Uboxes* are n-tupels of Unums, i.e. hyper-cuboids in some n-dimensional solution space. We are looking for the smallest set of uboxes which solve a given problem. Corresponding methods are declarative – similar to inclusion/exclusion, region growing, lumping, grid refinement, etc. – and cope [2] with problems like

- finite integrals (quadrature),
- (linear) equations with exact and inexact coefficients,
- evaluation of polynomials; computation of zeroes, extreme points and fix points,
- computation of location of a physical pendulum where $v = v(\vartheta), a = a(\vartheta), t = t(\vartheta)$,
- two- and many-body-problems,
- inverse kinematic [4]
- mass-spring-systems, trusses, FFT, CFD etc.

Advantages in each case are exact results or guaranteed intervals and (data) parallelism to be exploited.

For example, in Kahans *smooth surprise* Unums reveal that $f(x) = \frac{1}{80}\log|3(1-x)+1| + x^2 + 1$ has a pole in $\frac{4}{3}$ because of $f\left((\frac{4}{3} - ulp, \frac{1165}{87})\right) = [-\infty, \frac{175}{64})$. In another example, consider $f(x) = 2x - \frac{x^2}{L}$ for, e.g. $L = 77$. Then, using even very coarse Unums shows the fix point $L = 77$ to lie in $(73,81)$. For more examples see e.g. [2] and [6].

VI. CONCLUSION

Unums seem to be a viable alternative to IEEE 754 floating point numbers. To use Unums guarantees mathematical laws like associativity and distributivity. Results are provable true. Further, using Unums reduces memory and bandwidth requirements. Also, there are algorithms with Unums which are easily parallelizable. All this can be checked using the open source MATHEMATICA library written by J. Gustafson, to be downloaded from www.crcpress.com/The-End-of-Error-Unum-Computing/Gustafson/p/book/9781482239867, the website of [2].

Maybe as a result of a dispute between Gustafson [3] and Kahan [5], Gustafson [4] proposes a new format, *set of real numbers*, *SORN*, which no longer extends IEEE 754 but renders that outdated standard obsolete.

REFERENCES

[1] D. Bailey: Numerical reproducibility in high-performance computing, November 2015 http://www.davidhbailey.com/dhbtalks/dhb-num-repro.pdf

[2] J. Gustafson: The End of Error – Unum Computing, CRC Press 2015

[3] J. Gustafson: The Great Debate – Unum Arithmetic Position Statement; 23rd IEEE Symposium on Computer Arithmetic, 2016 http://arith23.gforge.inria.fr/slides/Gustafson.pdf

[4] J. Gustafson: A Radical Approach to Computation with Real Numbers; 2016 www.johngustafson.net/pubs/RadicalApproach.pdf

[5] W. Kahan: A critique of John L. Gustafson's The End of Error – Unum Computation and his Radical Approach to Computation with Real Numbers; 23rd IEEE Symposium on Computer Arithmetic, 2016 http://arith23.gforge.inria.fr/slides/Kahan.pdf

[6] Th. Risse: Unums; No 142, 146 of http://www.weblearn.hs-bremen.de/risse/papers/papers.htm