Farewell to Floats: A Personal History of the Rise and Fall of IEEE Std 754™

Prof. John L. Gustafson
Visiting Scholar, Arizona State University

March 2, 2023

The Dark Ages of Floating-Point Arithmetic

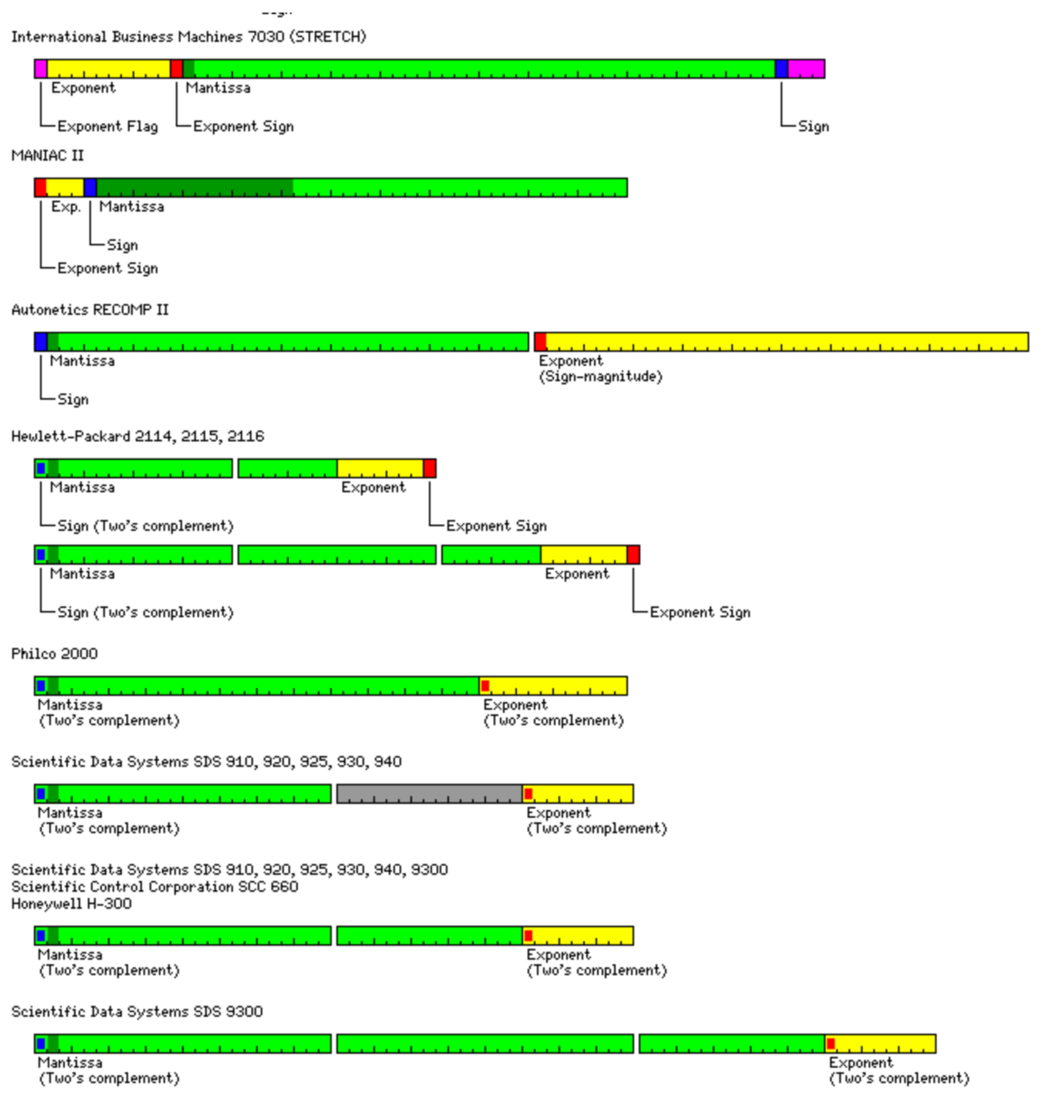
1950s: John von Neumann opposes floating-point hardware, saying it would lead to lazy ignorance of rounding errors.

1960s: IBM use a base-16 float format with severe “wobbling accuracy.”

No vendor even guaranteed that plus-minus-times-divide would produce **correctly-rounded results**!

“Single-precision” meant *36-bit*, until IBM’s 32-bit System/360 took over the market for computers (1965)

Float formats were all over the map. It was a zoo.



International Business Machines 7030 (STRETCH)
Exponent
Mantissa
Exponent Flag
Exponent Sign
Sign
MANIAC II
Exp.
Mantissa
Sign
Exponent Sign
Autonetics RECOMP II
Mantissa
Exponent (Sign-magnitude)
Sign
Hewlett-Packard 2114, 2115, 2116
Mantissa
Exponent
Sign (Two's complement)
Exponent Sign
Mantissa
Exponent
Sign (Two's complement)
Exponent Sign
Philco 2000
Mantissa (Two's complement)
Exponent (Two's complement)
Scientific Data Systems SDS 910, 920, 925, 930, 940
Mantissa (Two's complement)
Exponent (Two's complement)
Scientific Data Systems SDS 910, 920, 925, 930, 940, 9300
Scientific Control Corporation SCC 660
Honeywell H-300
Mantissa (Two's complement)
Exponent (Two's complement)
Scientific Data Systems SDS 9300
Mantissa (Two's complement)
Exponent (Two's complement)

Vendors **liked** the "lock-in" effect.

The significand and exponent sign bits could be *anywhere.*

Base-2, base-8, base-10, base-16

Different exponent sizes for the same number of total bits

No one even *expected* portability.

See: http://quadibloc.com/comp/cp0201.htm

A computer in a *hospital*?? Are you crazy?

1961: A cardiologist in Iowa wants to put a computer in Methodist Hospital.

No private hospital then had a computer. He says a computer could monitor EKGs, manage patient meals,… endless uses.

1963: He goes to IBM NY to close the deal.

His 8-year-old son asks, “Can I come?” and his parents say, “Why not?”

John E. Gustafson, MD

So I went along.

I saw acres of tape drives, laser research, engineers doing rock climbing in the cafeteria… *I was hooked*.

Dad is 98 now and still winning at contract bridge (he’s a Grandmaster twice over).

Grumpy Old Men…

1970: For me, it all started with this question…

What is

$i!$

?

None of the math teachers in my high school could tell me.
Like, what is it as a *numerical value*??

Progress: I Discovered the Gamma Function

$z! = \Gamma(z+1)$. Aha!

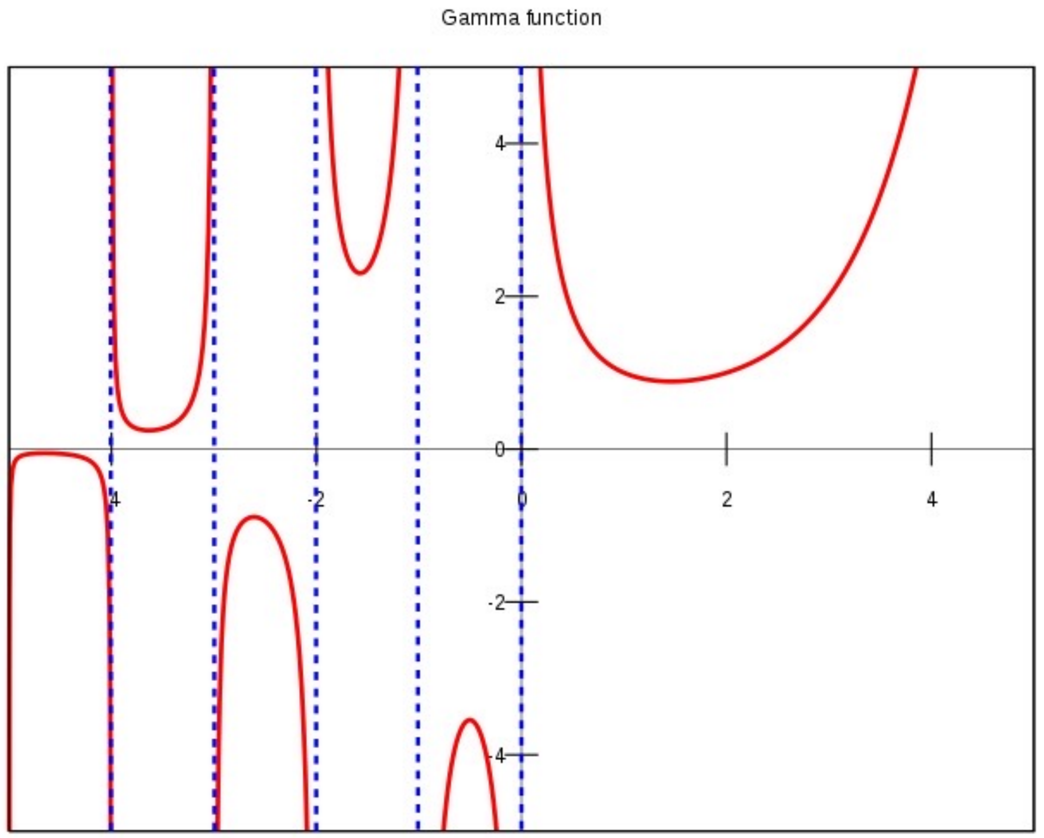
$\Gamma(z) \equiv \int_0^\infty t^{z-1} e^{-t} dt$. But…
how do you evaluate that for $i+1$?

$\Gamma\left(\frac{1}{2}\right) = \left(-\frac{1}{2}\right)! = \sqrt{\pi}$. Really?? Wild.

More progress when I found this:

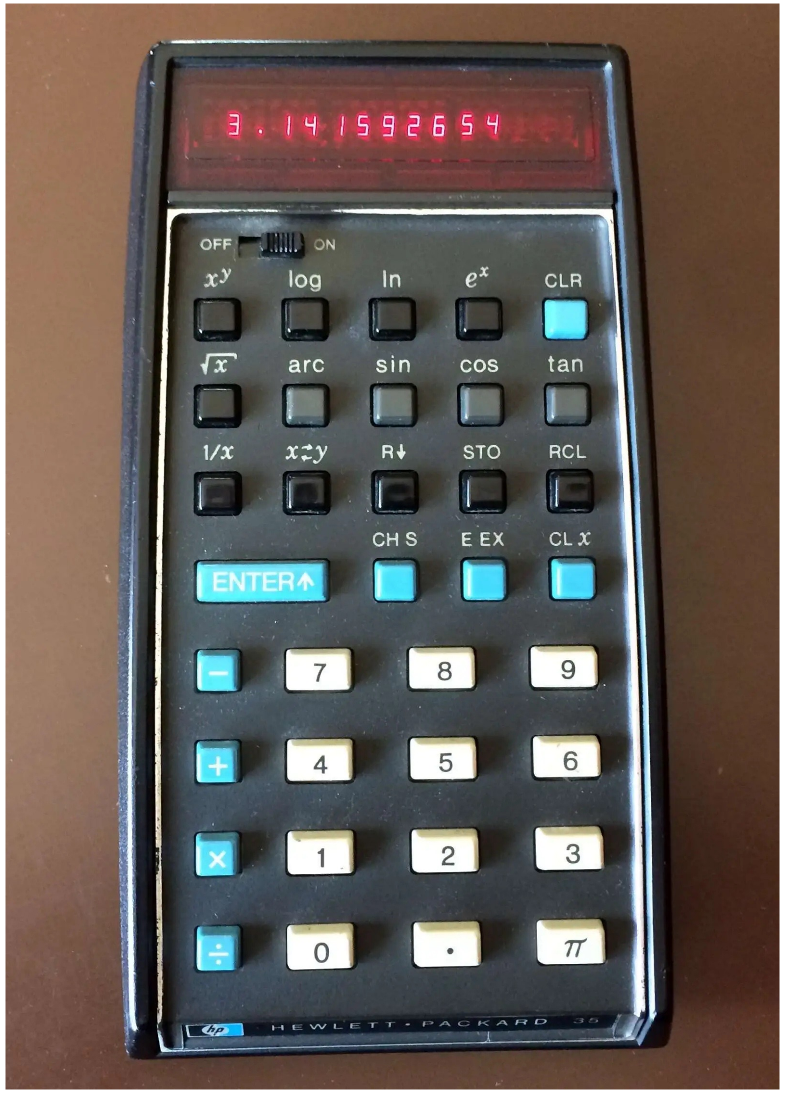
$$\Gamma(z) = \frac{1}{z} \prod_{n=1}^{\infty} \left[\frac{1}{1+\frac{z}{n}}\right] \left(1+\frac{1}{n}\right)^z$$

Looks calculable, except for that ∞ part… and convergence is ***slow.***



1972: Hewlett-Packard introduces the HP-35

- The first “electronic slide rule”
- US$395 (that’s **US$2827** today)
- I wonder if it would allow calculation of $i!$ using $z! \sim \sqrt{2\pi z}\left(\frac{z}{e}\right)^{z}$.
- Other teenagers wanted their own car. *I wanted one of these.*



Trimmed trees, hedges, weeded at $3/hour

- 17-year-old me scrimps and does odd jobs, and finally *gets one*.
- Huge academic advantage
- Decimal floating-point… and the function routines *had bugs*. Like, exp(ln(2.02)) = 2.
- I learn to be *very* suspicious of floating-point calculations.

Early 1970s: HP forms a calculator division

- HP hires a consultant, this guy:
- A Canadian mathematician named William “Velvel” Kahan
- Already known for brilliant numerical analysis papers
- Also known for his *weaponized sense of humor*
- His first designs are the HP-45, HP-67, and HP-97

Kahan's HP-97 got me through Caltech

- It was *programmable*.
  A numeric PC.
  With a printer!
- Other students used the campus mainframe for homework.
- High-quality 10-decimal floats, 10^{-99} to 10^{99}
- My girlfriend resented the attention I gave *it* instead of her.

1975: Trying to calculate $i!$ leads me to a math breakthrough

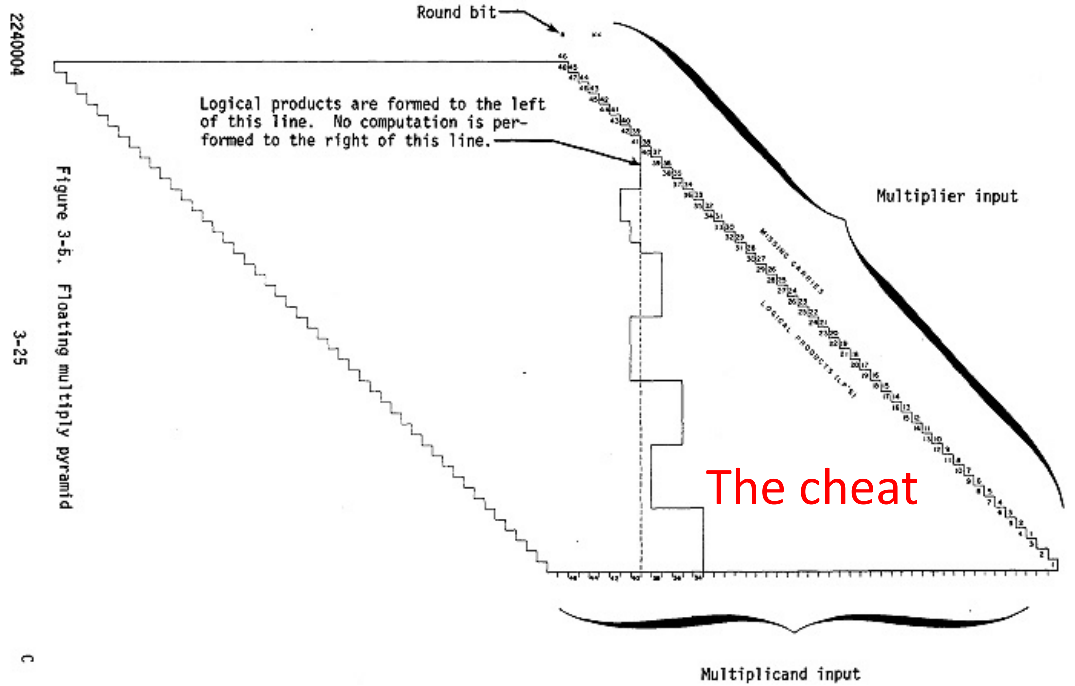
- I discover a *fast-convergence* infinite product for the Gamma function in my sophomore year.
- Caltech gives me the Eric Temple Bell Award for it.
- Applied mathematician John Todd shows me how to publish a math paper (my first).
- It allows me to calculate $i! \approx 0.498 - 0.155i$, after ten years wondering.

$$\Gamma(z) = z^{z/2-1} \Pi \left[(1+n^{-1})(1-(z+n)^{-1})^{-1}\right]^{z/2}(1+z\,n^{-1})^{-1}.$$

1977: Cray-1 had 64-bit floats, 16-bit exponent

Cray-1 fails $a \times b = b \times a$, to save transistors.

To quiet complaints, Cray *sorts* the inputs so $a \le b$. Problem “solved.”


2240004
Round bit
Logical products are formed to the left of this line. No computation is performed to the right of this line.
Multiplier input
MISSING CARRIES
LOGICAL PRODUCTS (LP'S)
The cheat
Figure 3-6. Floating multiply pyramid
3-25
Multiplicand input

Also in 1977…

DEC introduces the VAX-11/780.

The Apple II is introduced, at $666. No floating-point.

TRS-80 personal computer comes out. Floating-point built into ROM. Yes!

Once again, *I had to have one*.

Dog eating my last algorithm

1978: I discover a kindred spirit as a grad student

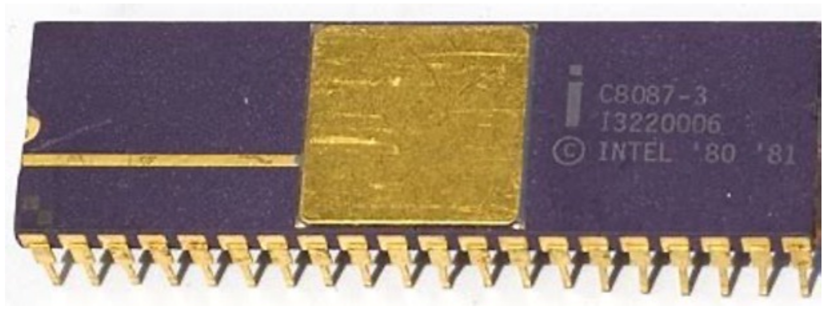
At Caltech we find a common interest in doing *much* better math on computers.

He says he's building "SMP", a "Symbolic Manipulation Program."

Stephen Wolfram

Years later, Steve Jobs will suggest that *Mathematica* would be a cooler name for it. Wolfram agrees.

The origin of IEEE 754



1977: Intel's John Palmer is green-lighted to co-architect the i8087 coprocessor with Bruce Ravenel. Kahan brought in as consultant.

John Palmer, 1946–2017

1980: Intel announces the ambitious chip. $150. Its 64-bit double used an 11-bit exponent. 50 kFLOPS.

Word of i8087 creates industry panic over Intel clout, and demand for an IEEE standards committee to make playing field fair.

Oct 1979: Kahan, Palmer et al. announce a *proposed standard* for floating-point.

Why 11-bit or 16-bit exponent for 64-bit floats?

Because exponent size is $\log_2$(precision) + constant? NO

Because studies were done that showed that this produced the dynamic range needed for scientific computing? NO

Because that size means all single-precision products $a \times b$ can be stored exactly in double-precision? NO

Because big significand *multipliers* were really hard to build but big exponent *adders* were cheap and easy? BINGO

Donald Knuth on Underflow to Zero

"It has unfortunately become customary in many instances to ignore exponent underflow and simply to set underflowed results to zero with no indication of error. This causes a serious loss of accuracy in most cases (indeed, it is the loss of *all* the significant digits), and the assumptions underlying floating point arithmetic have broken down…"

The Art of Computer Programming: Seminumerical Algorithms,
Vol.2, First Edition, 1969

Internal IEEE 754 War over *Gradual Underflow*

- Decision misstep: Choosing underflow to **zero**, not ±*minReal.*
- Result: $X - Y$ can be zero when $X \neq Y$. Major contradiction.
- David Goldberg introduced denormal floats as a band-aid. Kahan backed him. The heated arguments lasted **six years**.

David Goldberg

William Kahan

An i8087 Throwback Idea: A Separate “Sign Bit”

IBM 701, 1953

Negative integers were originally stored in “sign-magnitude” form, imitating the way humans write + and – before digit strings.

$$\begin{tikzcd} + \arrow[d] & 5 \arrow[d] \\ 0 & 101 \end{tikzcd} \qquad \begin{tikzcd} - \arrow[d] & 5 \arrow[d] \\ 1 & 101 \end{tikzcd}$$

BAD idea. Why? Well, here’s one reason:

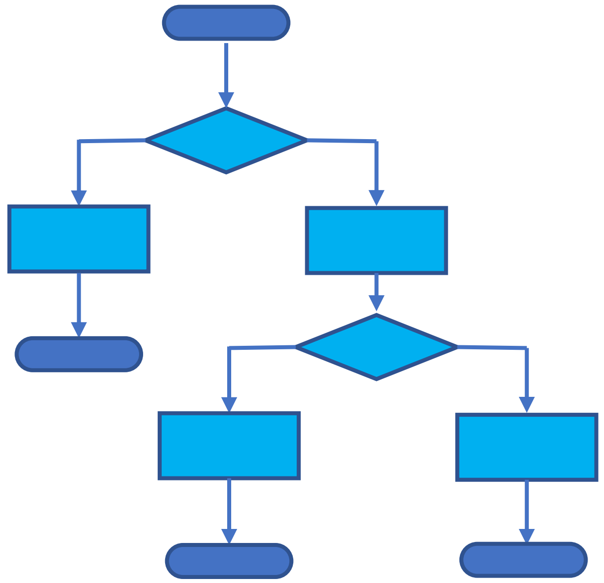
$$\begin{tikzcd} + \arrow[d] & 0 \arrow[d] \\ 0 & 000 \end{tikzcd} \qquad \begin{tikzcd} - \arrow[d] & 0 \arrow[d] \\ 1 & 000 \end{tikzcd}$$

Welcome to the joys of “negative zero.”

Remember adding signed numbers in school?

To add nonzero signed integers *m* and *n*:

```
Are they the same sign or different sign?
    If they are the same sign, add their magnitudes.
        Apply that sign to the resulting sum, DONE.
    Else if they have different signs,
        Find out which magnitude is bigger.
        If m has bigger magnitude,
            Subtract n's magnitude from m's magnitude.
            Apply m's sign to the result. DONE.
        Else
            Subtract m's magnitude from n's magnitude.
            Apply n's sign to the result. DONE.
```



Floating Point Systems

FLOATING POINT SYSTEMS, INC.

1981: FPS introduces a 64-bit attached processor (11-bit exponent… hmm…) as the IEEE 754 debates rage.

1982: Post-PhD, I join Floating Point Systems. I learn about Kahan and meet him at conferences.

1984: I’m asked to lead a “skunk works” project to create a massively parallel hypercube supercomputer that will fill a building.

1986: IBM’s Alan Karp announces his defiant \$100 bet that massive parallelism cannot get to 200× speedup; Digital’s Gordon Bell joins in offering \$1000 Prize for best parallel speedup.

Ulrich Kulisch and Interval Arithmetic

- FPS also introduced me to a 1981 book by Kulisch and Miranker proposing *interval arithmetic* to control rounding error.
- It involves doing *exact dot products* and bounding the calculations. FPS built hardware for this as an R&D project.
- Parallel processing gives *different answers* unless you use exact dot products to restore the associative property, $(a + b) + c = a + (b + c)$.
- Epiphany. **Finally**, a way to end rounding error!
- Hmm… what’s the catch?

Kulisch battles to get the EDP into IEEE 754

- The "exact register" for doubles is, like, 4,664 bits wide! (Because of oversized exponents)
- How does it handle exceptions like NaN and ∞? Answer: It doesn't.
- Kahan frequently flips between "Everything must be perfect" to "Perfection is impractical." (With scathing humor for each.) He refused to believe the exact approach was actually *faster*.

Prof. Ulrich Kulisch
Karlsruhe Institute of Technology

Kahan rejected it from IEEE 754, saying *extended double* suffices.

Here’s the magic that IEEE 754 threw away

Any expression using + – × / can be written as L*x* = *b* where L is lower triangular and the last x_n is the desired calculation. Example:

$$\boxed{f = (a + b) \times c - d / e}$$

$$\begin{aligned} x_1 &= a \\ x_2 &= x_1 + b \\ x_3 &= c \times x_2 \\ x_4 &= d \\ e \times x_5 &= x_4 \\ x_6 &= x_3 - x_5 \end{aligned} \quad \Longrightarrow \quad \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 & 0 & 0 \\ 0 & c & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -1 & e & 0 \\ 0 & 0 & -1 & 0 & 1 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{pmatrix} = \begin{pmatrix} a \\ b \\ 0 \\ d \\ 0 \\ 0 \end{pmatrix}$$

Exact dot products allow you to find *f* to within 0.5 ULP!

Another i8087 decision that got into IEEE 754: "Floats overflow to ±∞." Why that's a bad idea:

Calculate $\dfrac{n^2}{\sqrt{n^3+1}}$ for n = 42.

Exact answer to three decimals: 6.47⋯

Answer using 16-bit IEEE 754 Standard floats:

0.00 **FAIL**

By the way: answer using 16-bit posits: 6.48⋯

Folly: Use *processor flags* to indicate a problem

Set $x = 2.6469783 \times 10^{-23}$ (as a 32-bit float, should be `0x1A000001`). Compute $y = \text{sqrt}(x^2)$. Should get something close to x, right?

An IEEE 754 compliant system will give you $3.7433921 \times 10^{-23}$, a relative error of 41% with not even a single correct digit.

A typical GPU will return 0.

In 38 years, not one language has provided support for viewing the IEEE 754 processor flags. Programmers only see underflow when the result is 0 and shouldn’t be.

Folly: The 9 Quadrillion Names for NaN

IEEE 754 doubles allow 9,007,199,254,740,990 bit patterns that indicate an answer is indeterminate, or Not-a-Number (NaN).

Is NaN = NaN? The Committee flunked Aristotelian Logic 101 and decided, *no*. A thing is not always equal to itself!

Kahan hoped vendors would *encode the program address where the error happened*. Only HP tried doing that, and only once.

Nothing ruins a number system more than having redundant ways to represent the same thing.

The IEEE 754 battle rages for *six years*

1979: Kahan immediately loses three “must have” demands:

- Decimal, not binary
- Perfect reproducibility across systems
- 128-bit “extended double” precision

Intel (Palmer) insists the i8087 will **define** IEEE 754.

1982: IBM PC introduced, creates $2B market overnight, including demand for better floating-point performance. At least *kiloflops*.

1985: IEEE Std 754™ ratified, with grumbling about denormals.

Principles for Computer Arithmetic Design

- No **covert** use of extra precision. Ever.
- No redundant bit patterns to mean the same value
- Math libraries get correctly-rounded answers for ALL inputs.
- Support for exact sums, exact dot products (associativity)
- No hidden “modes” or invisible “processor flags”
- Map reals monotonically to signed integer format
- Stable when 0 bits appended, making precision change trivial
- Don’t underflow or overflow; saturate to ±*minReal,* ±*maxReal.*

IEEE 754 floats failed *every single one* of these principles.

nCUBE

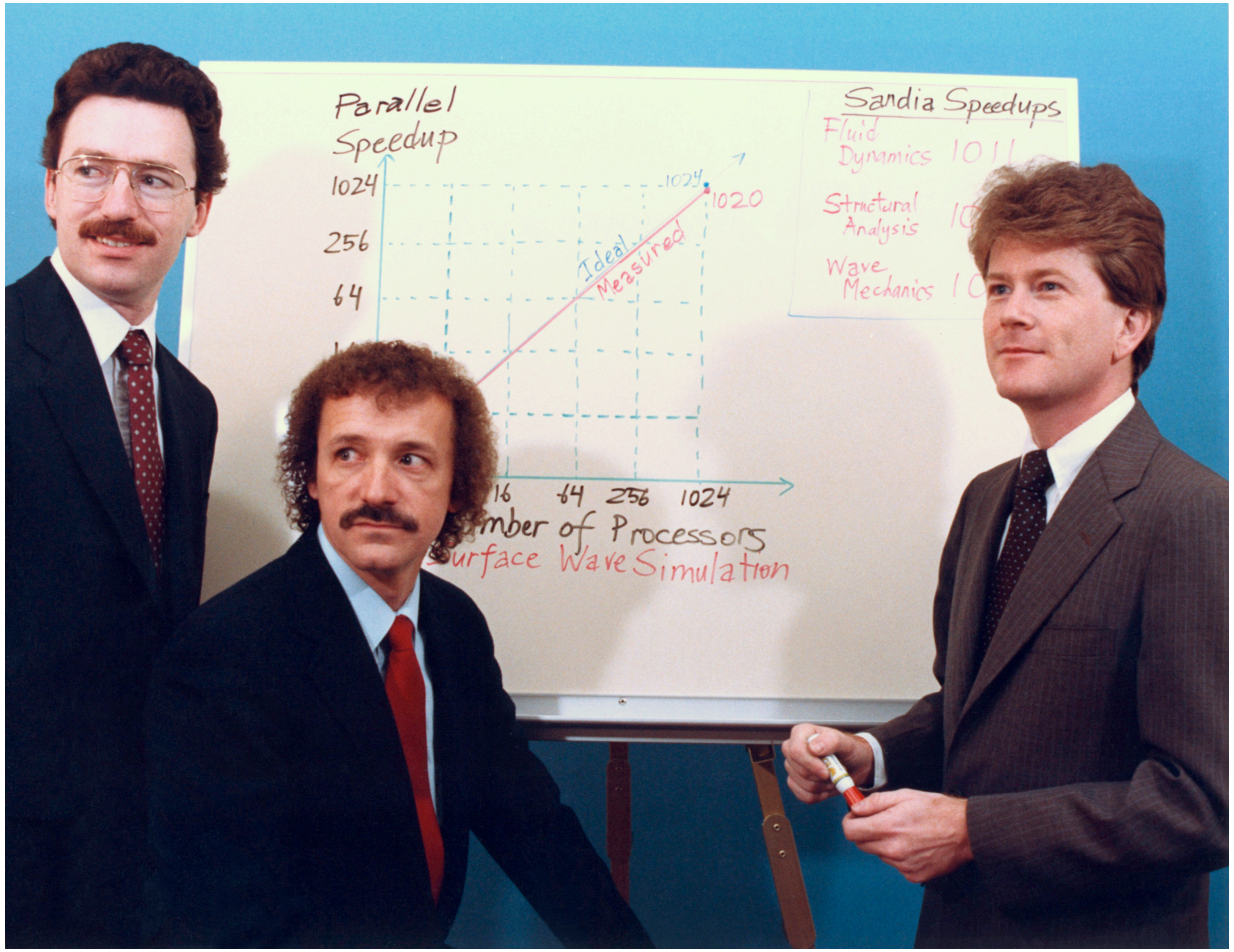
1986: Delays in the Inmos transputer trash my FPS hypercube project… but nCUBE has a full-custom system that ***works***.

I join nCUBE, reporting to the CEO… John Palmer! Palmer fills me in on all the gory details of how IEEE 754 came to be.

Sandia buys a full-size nCUBE with 1024 processors. I realize this is my chance to prove that Amdahl was wrong: *parallelism works*.

1987: To use the 1024-processor system, I need to become a customer. So I join Sandia and conspire to beat the Karp and Bell challenges.

1987–1989: Sandia National Labs



Robert Benner, Gary Montry, and me after winning the first Gordon Bell

- Speedup of 1020 from 1024 processors, and a theoretical explanation of why Amdahl's law didn't stop us
- Parallelism changes rounding errors slightly…
- **Disruption**: Industry embraces parallel computing at last.
- At age 33, I find that a law has been named after me.
- Great! Now I can get funding to start my own lab!

1990–2000: The Scalable Computing Lab

- DOE-funded lab within Ames Lab at Iowa State
- Quickly grew to ~20 people, won R&D 100 Awards
- Found a validated arithmetic approach for radiation transfer modeling (radiosity graphics)
- Found a validated arithmetic approach for Laplace’s Equation, which had defied interval methods
- “If I can find enough examples, I’m going to write a book on how to fix computer arithmetic.”

Two-Body Problem Forces Return to Industry

1999: I marry Dr. Phyllis Crandall of DOE’s Los Alamos National Lab. I’m still at DOE’s Ames Lab. DOE has serious funding challenges and a hiring freeze.

2000: Sun Microsystems is hiring anyone who can fog up a mirror. Phyl and I join Sun and meet interval arithmetic fanatic Bill Walster, son of a tent revivalist minister.

2005: Sun drops *hardware* support for IEEE 754 exceptions; some customers howl.

2008: IBM Gets the Fused Multiply-Add into 754

- IBM already had put this into their processors.
- Sticking it into the IEEE 754 Standard (2008) would cleverly force all their competition to redesign their CPUs at great expense.
- Sun and Intel screamed “bloody murder!” but IBM eventually won.
- *Should compilers fused multiply and add opportunistically and covertly?*

If you do, and you evaluate

$$\sqrt{x^2 - y^2}$$

when $x = y$, half the time you will get NaN, not zero!

Intel’s view of IEEE 754 as of 2009

2009: I join Intel, propose a Technical Strategic Long-Range Plan (TSLRP) to improve floats.

It gets approved. This is a Big Deal. I suggest fundamentally re-vamping IEEE 754. Intel’s CTO, Justin Rattner, says:

I think, but do not say out loud:

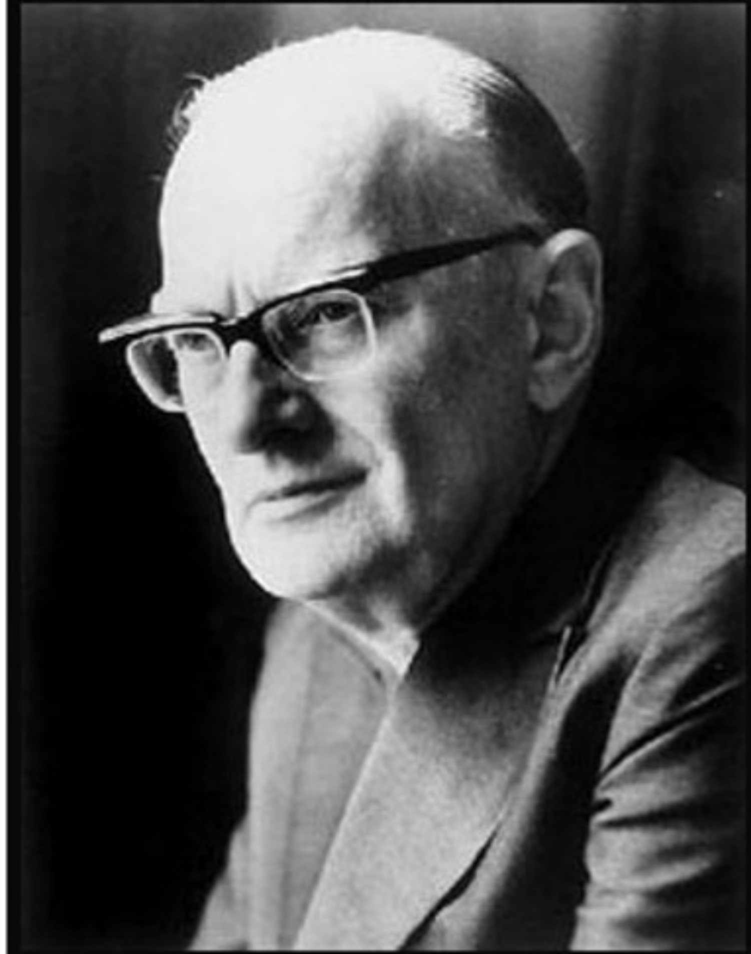
Fix the IEEE 754 Standard?

Can we get help from “the father of IEEE 754,” Bill Kahan? Probably not.

“Linguistically legislated exact reproducibility is unenforceable.”

“Faster matrix multiplication is usually too valuable to forego for unneeded exact reproducibility.”*

from “How Java’s Floating-Point Hurts Everyone Everywhere”

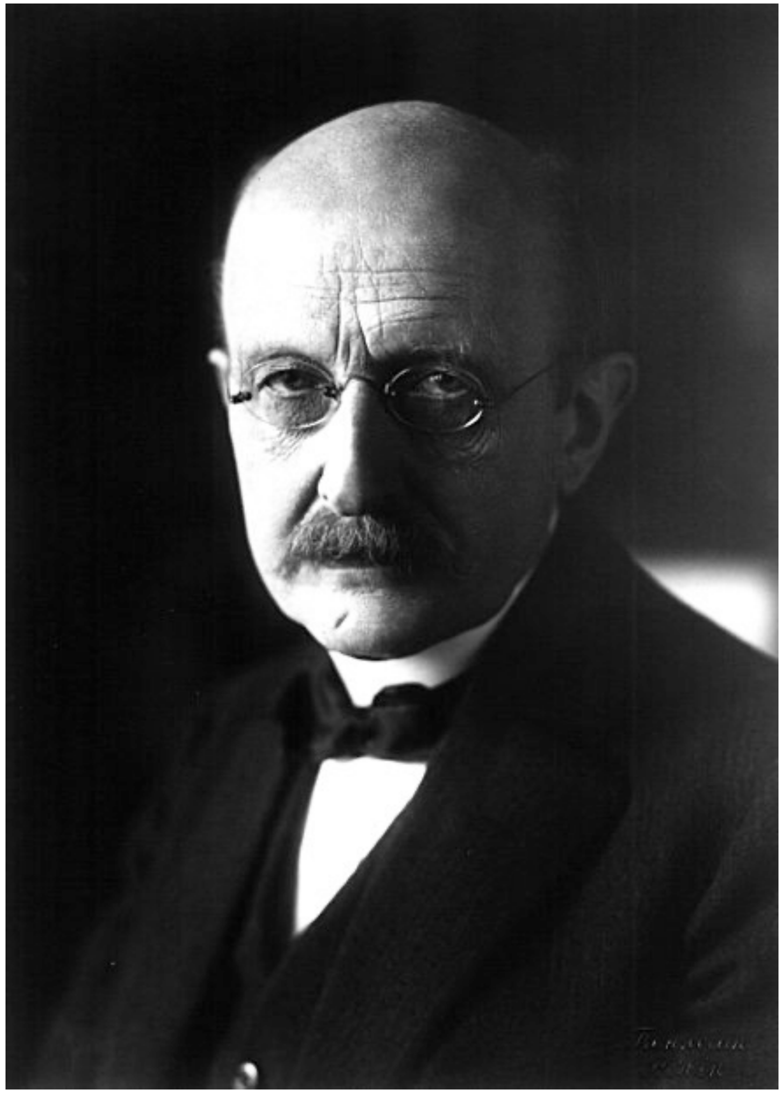


If an elderly but distinguished scientist says that something is possible, he is almost certainly right; but if he says that it is impossible, he is very probably wrong.

(Arthur C. Clarke)

Plancks’ Principle

“A new scientific truth does not generally triumph by persuading its opponents and getting them to admit their errors, but rather by its opponents gradually dying out and giving way to a new generation that is raised on it.”


Max Planck (1858–1947)

Playbook for a successful technology disruption

- Do NOT patent the idea. Open-source everything.
- Introduce it to small companies who have nothing to lose and everything to gain. (Big companies always resist change.)
- Amass a pile of *real-world examples* where the idea is a win.
- Also create a *theory* of why the idea works or is better.
- **Sit back and enjoy the chaos.** Change takes about ten years.

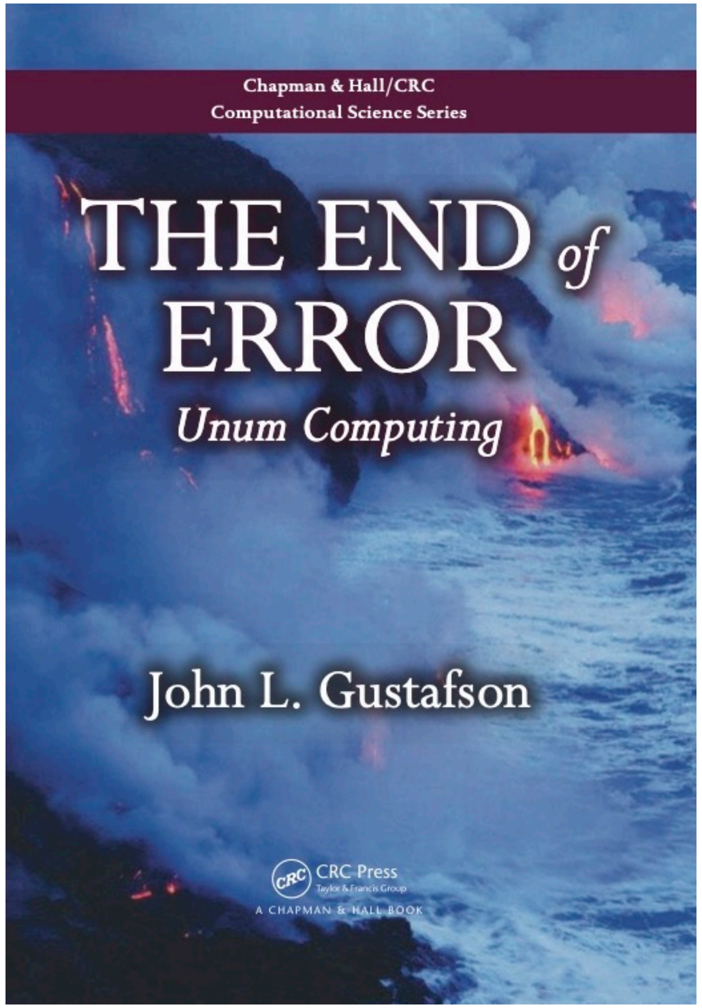
Open source everything? Check.

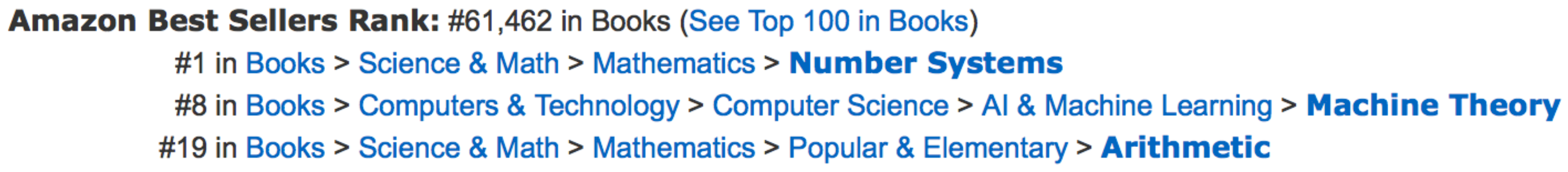
2013–2015: I give myself a sabbatical to write a book, hence no corporate IP rights. In *Mathematica.* Email with Kahan goes dead.

Kahan reportedly goes ballistic at the *title* of the book. (That was intentional.)

That photo on the cover is a Hawaii volcano… **boiling the ocean**.

Feb 2015: *End of Error: Unum Arithmetic* hits #1 in its Amazon category.




Amazon Best Sellers Rank: #61,462 in Books (See Top 100 in Books)
#1 in Books > Science & Math > Mathematics > **Number Systems**
#8 in Books > Computers & Technology > Computer Science > AI & Machine Learning > **Machine Theory**
#19 in Books > Science & Math > Mathematics > Popular & Elementary > **Arithmetic**

The Singapore Connection

Mar 2015: Singapore’s A*STAR thinks I’m onto something with unums.

A*STAR (Marek Michalewicz) invites me to come to A*STAR / NUS to advance unum research (and maybe make it hardware-friendly).

Free room and board, six-figure salary in a tropical paradise to do what I was doing anyway…

Hmm… *tough decision*. (not)

2016: The Great Debate: The Wrath of Kahan

June 2016: Furious at the success of *The End of Error: Unum Computing*, Kahan (age 83) challenges me to a **public debate** at the ARITH conference. I accept.

This is the man who, indirectly, taught me numerical analysis. My hero and my friend. I keep trying to stand on his shoulders to see farther, and he keeps trying to flick me off!

I pointed out that weaponized humor may work for Donald Trump, but has no place in a mathematical debate. The debate did *not* go well for him.

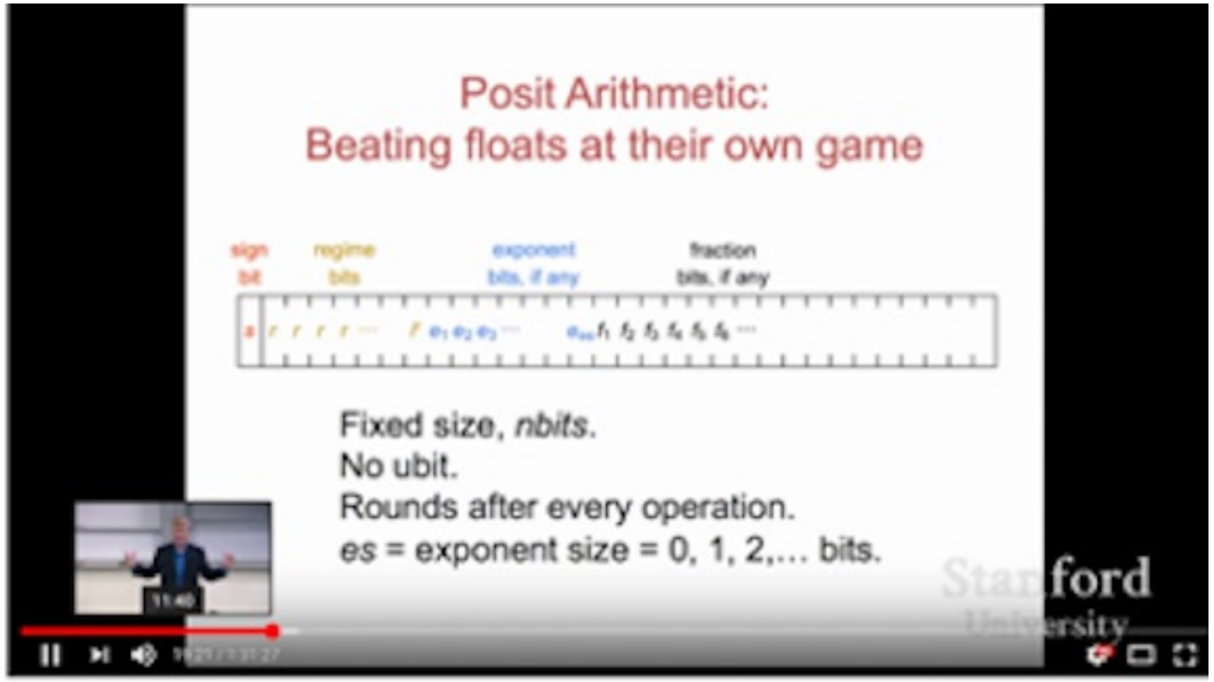
The full transcript is on the web.

Epiphany: Hardware-Friendly Unums

November 2016: RISC-V asks for a hardware-friendly version of unum arithmetic.

December 3, 2016: Got it! Yonemoto figures out how to do the decoding.

Feb 3, 2017: Presented at Stanford as part of EE380 Colloquium Seminar series.



Posit Arithmetic:
Beating floats at their own game

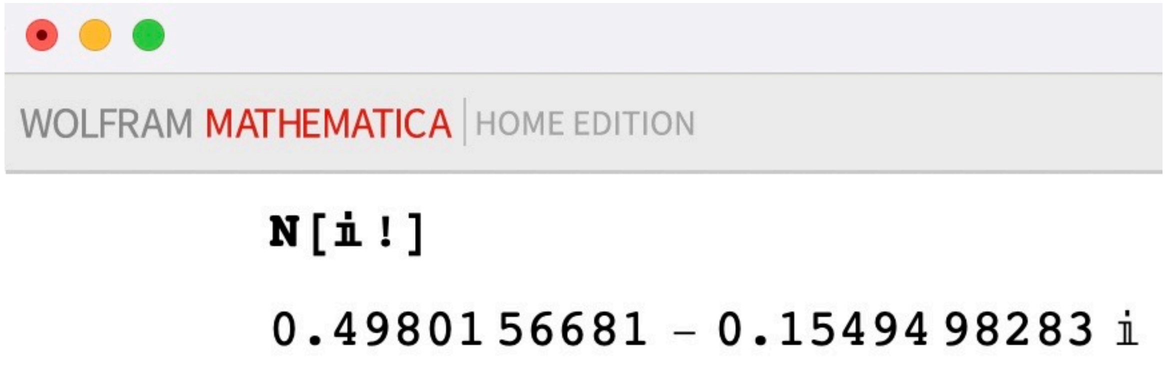
Fixed size, *nbits*.
No ubit.
Rounds after every operation.
es = exponent size = 0, 1, 2,… bits.

Stanford Seminar: Beyond Floating Point: Next Generation Computer Arithmetic

26,000+ views.
Now sit back and enjoy the chaos!

In Summary

- IEEE 754 had a good run, but *technology changes are inexorable*.
- IEEE 754 floats (of any precision) are now **obsolete**; vendors are disingenuous about noncompliance.
- I've had a front-row seat at this skirmish for a *very long time*.
- This is an opportunity for a clean-slate design (like posits).
- Oh… and I don't have to puzzle over $i!$ any more:



```
N[ⅈ!]

0.4980156681 - 0.1549498283 ⅈ
```

Farewell to floats.
Hello, CoNGA!
THANK YOU.