Make Computer Arithmetic Great Again?

Jean-Michel Muller

CNRS, ENS Lyon, Inria, Université de Lyon

France

ARITH-25

June 2018

An apparent contradiction

- low number of paper submissions to Arith these last ≈ 5 years
- few PhD defenses, few academic positions (at least in North America)

  and yet...
  - just in the last 9 months, great arithmetic-related papers in IEEE Trans. VLSI, IEEE Trans. Computers, ACM TOMS, Mathematics in Computer Science, Numerical Algorithms, Mathematics of Computation, BIT... by authors who don't or rarely frequent the Arith conferences;
  - people in the industry still design arithmetic operators, they also have new needs: deep learning, certified and /or reproducible calculations (e.g. for automated transportation), mixed precision...

Why don't we see these people?

Why don't we see these people?

- because it is seldom "pure" arithmetic, merely arithmetic **and** something;
- because if they submitted to Arith, we might well reject their papers!

Why don't we see these people?

- because it is seldom "pure" arithmetic, merely arithmetic **and** something;
- because if they submitted to Arith, we might well reject their papers!

(and we probably have)

Why don't we see these people?

- because it is seldom "pure" arithmetic, merely arithmetic **and** something;
- because if they submitted to Arith, we might well reject their papers!

(and we probably have)

Necessary expertise in related areas:

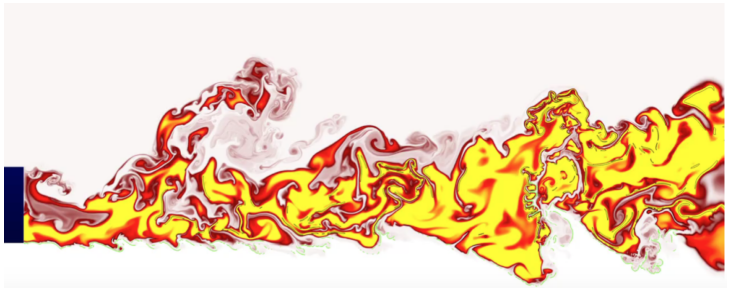
- our community has expertise on circuit design;
- we have low expertise in numerical analysis, compilation, formal proof, finite field arithmetic...

We used to have

- a few formats: single precision, double precision;
- a few applications: numerical simulation, financial calculations.

Is there a thing such as an "Universal solution"?

Numerical simulation

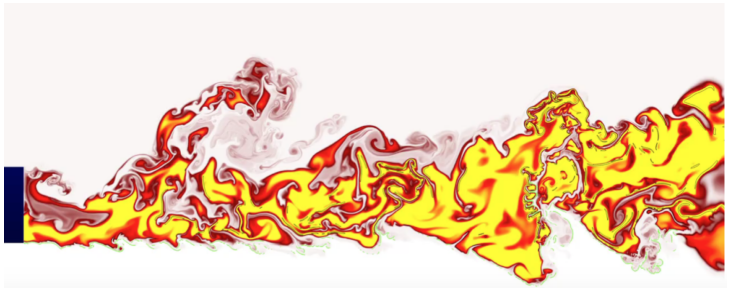


Embedded computing

Entertainment

Is there a thing such as an "Universal solution"?

Numerical simulation



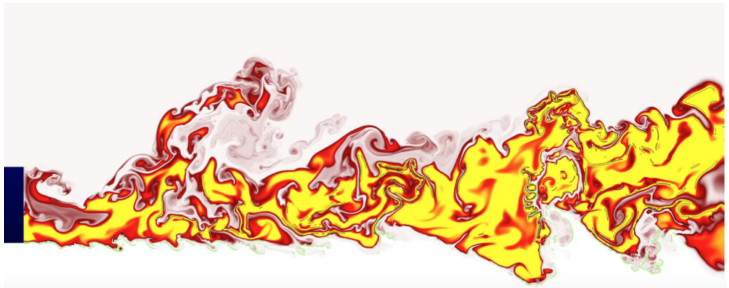
- trillions of operations
- floating-point (dynamic range, speed, accuracy)
- crash? just start again the simulation (but not too often)

Embedded computing

Entertainment

Is there a thing such as an "Universal solution"?

Numerical simulation



- trillions of operations
- floating-point (dynamic range, speed, accuracy)
- crash? just start again the simulation (but not too often)

Embedded computing

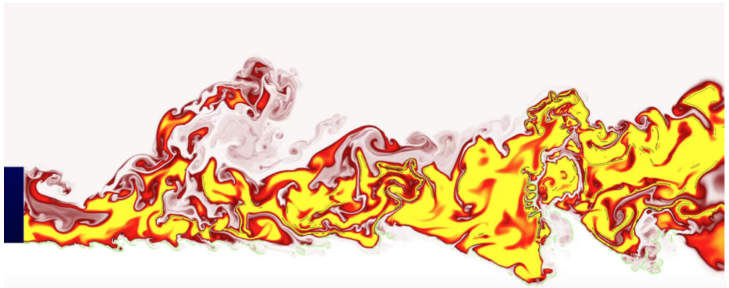
- speed: no need to be faster than real time;
- crash? ahem...

→ certified calculations.

Entertainment

Is there a thing such as an "Universal solution"?

Numerical simulation



- trillions of operations
- floating-point (dynamic range, speed, accuracy)
- crash? just start again the simulation (but not too often)

Embedded computing

- speed: no need to be faster than real time;
- crash? ahem...

→ certified calculations.

Entertainment

- supermario's pizza does not need to carefully follow the laws of physics;
- fluidity matters.

Playing with different formats ?

See Nick Higham's talk at Arith 24.

single precision (a.k.a. binary32)
double precision (a.k.a. binary64)

$$\Rightarrow \begin{cases} \text{“quarter precision”} \\ \text{“half precision” (binary16)} \\ \text{binary32} \\ \text{binary64} \\ \text{binary128 (quad)} \end{cases}$$

Combinatorial explosion of all the possible arithmetic operators of the form $\texttt{Format 1} \times \texttt{Format 2} \rightarrow \texttt{Format 3}$.

Cleverly using these formats:

Locate when
low precision
puts us
at an unacceptable risk.
Numerical analysis
abstract interpretation

$\longleftrightarrow$

Locate when
big precision
totally destroys
performance.
Compilation
Computer architecture

Reproducible computing: useful and sometimes dangerous

Reproducibility in computer arithmetic: examples are

- Reproducible Basic Linear Algebra Subprograms at Berkeley (`https://bebop.cs.berkeley.edu/reproblas/`);
- paper on reproducible summation by P. Ahrens, H.D. Nguyen and J. Demmel.

Main arguments in favor of reproducibility:

- consistence (parallelism: sometimes evaluating the same expression in different places is cheaper than transmitting it);
- debugging is difficult if we cannot reproduce errors;
- contractual/legal reasons.

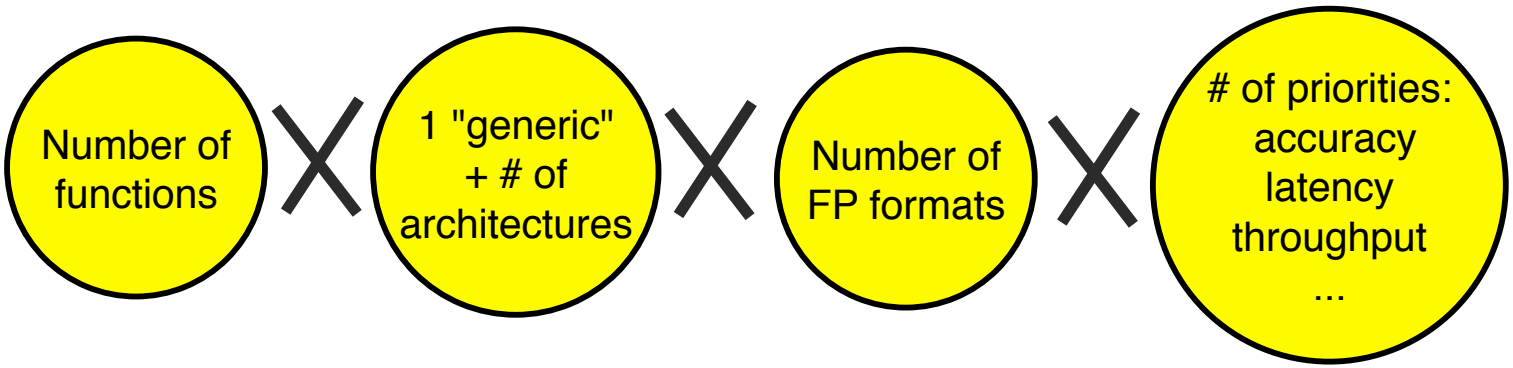
Significant demand (from HPC) and interesting problems $\rightarrow$ need to work on these issues.

Reproducible computing: useful and sometimes dangerous

However...

- obtaining very different results when running the same program twice is a sign that something weird is going on (of physical, numerical or programming origin). This is an useful warning, not to be disabled systematically;
- the legal reasons are fine, but there may as well be legal reasons against reproducibility: be ready to explain to a court that you deliberately delivered a less accurate result.

Libraries of math functions



$=$ thousands of function programs

- impossible to debug, maintain, keep consistent, improve...
- and physicists would like many other functions

First solution: computer-assisted library design

Metalibm project (`http://www.metalibm.org`). Two versions

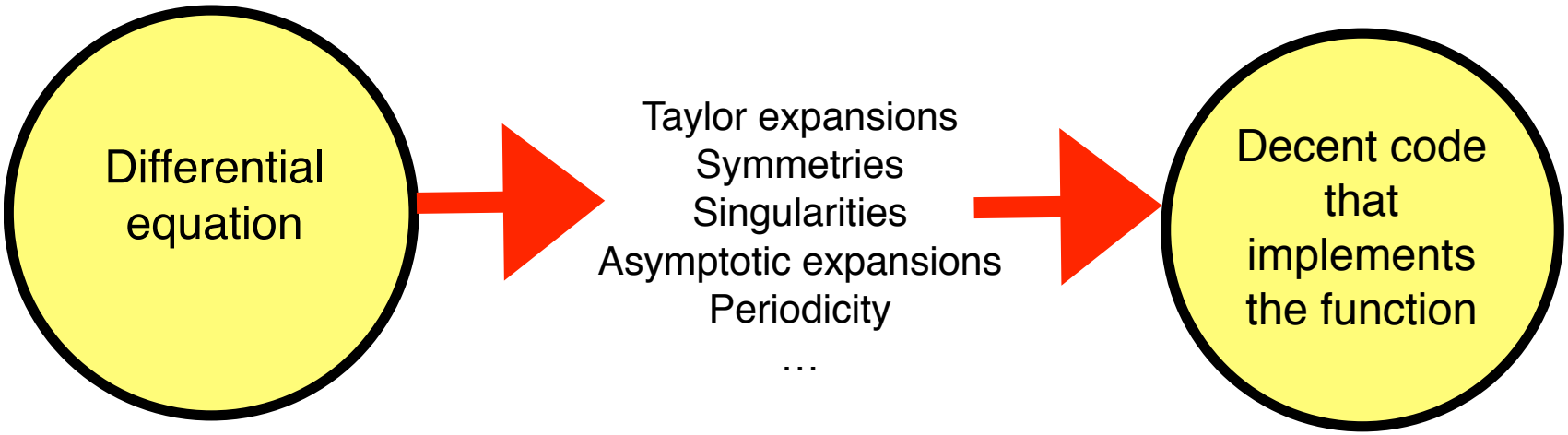
- fully automated for the end user;
- assistance for the specialist.

Metalibm builds upon tools such as

- Sollya (`http://sollya.gforge.inria.fr`): get approximations with many possible constraints;
- Gappa (`http://gappa.gforge.inria.fr`): tight bounds to polynomial evaluation errors and formal proofs.

But this is not the ultimate goal

Tools from computer algebra



- NumGfun: a Package for Numerical and Analytic Computation with D-finite Functions. ISSAC 2010.
- Mezzarobba's PhD dissertation *Autour de l'évaluation numérique des fonctions D-finies* (Ecole Polytechnique, Paris, 2011);
- complex mathematics but big reward.

Incidentally, people from computer algebra need us to speed up various algebraic computations.

Tools from computer algebra

Dynamic Dictionary of Mathematical Functions:

`http://ddmf.msr-inria.inria.fr/1.9.1/ddmf`

The Special Function $J_2(x)$

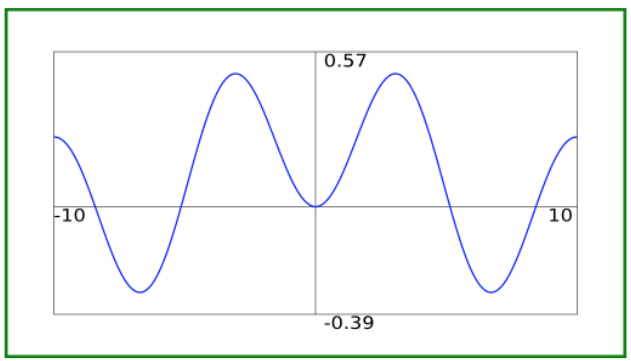
1. Differential Equation

The function $J_2(x)$ satisfies the differential equation

$$x^2 \frac{d^2}{dx^2} y(x) + x \frac{d}{dx} y(x) + (x^2 - 4)\, y(x) = 0$$

with initial values $y''(0) = 1/4$ and $y^{(4)}(0) = -1/4$.

2. Plot



min = -10 max = 10 Envoyer

3. Derivative in Terms of Lower-Order Derivatives

$$\frac{d^6}{dx^6} J_2(x) = \frac{(-x^6 + 21x^4 - 420x^2 + 2520)\, J_2(x) - 3x\,(x^4 - 35x^2 + 420)\, \frac{d}{dx} J_2(x)}{x^6}.$$

order = 6 Envoyer

4. Taylor Expansion at 0

Taylor coefficients:

$$J_2(x) = \frac{1}{4} \sum_{n=0}^{\infty} \frac{(-1)^n\, x^{2n+2}}{4^n n!\,(n+2)!}.$$

Generation of functions at compile-time

- take into account the exact context: underlying architecture, accuracy requirements, priorities (latency/throughput);
- possibly, information on input domain ($\rightarrow$ simplify/avoid range reduction), or special cases (e.g., infinities, NaNs known not to happen);
- compound functions: if you need

$$E_4(x) = \frac{x}{e^x - 1} - \ln(1 - e^{-x}),$$

  then you directly generate $E_4(x)$ instead of generating exp, ln and combining them.
- formal proof absolutely necessary (no library to heavily test beforehand);
- need to work with people from mathematics, computer algebra, compilation, formal proof...

Alternate number representations?

I am not a big fan of Unums, but I reckon J. Gustafson has a point: the considerable increase, in the last 20 years, of the ratio

$$\frac{\text{time to read/write in memory}}{\text{time to perform } +, \times, \div, \sqrt{}}$$

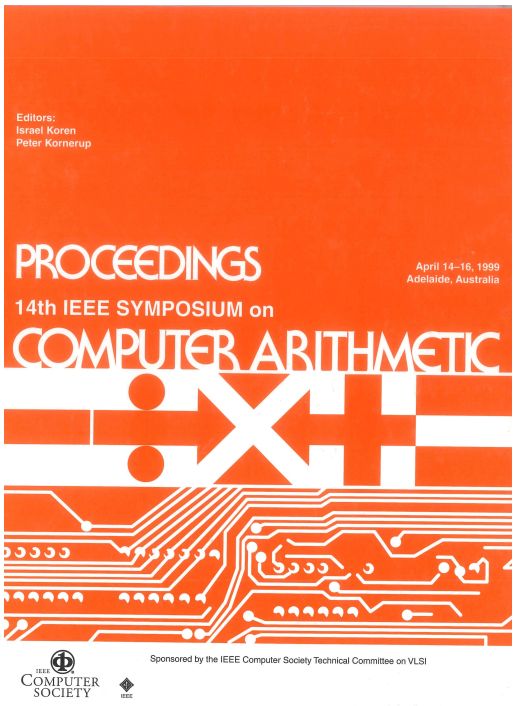
should be viewed as an interesting challenge, not as a sign that we have become useless.

- we have time to do "more" things
- attach easy-to-compute additional information to FP numbers?
- develop communication-avoiding algorithms.

Other topics of interest

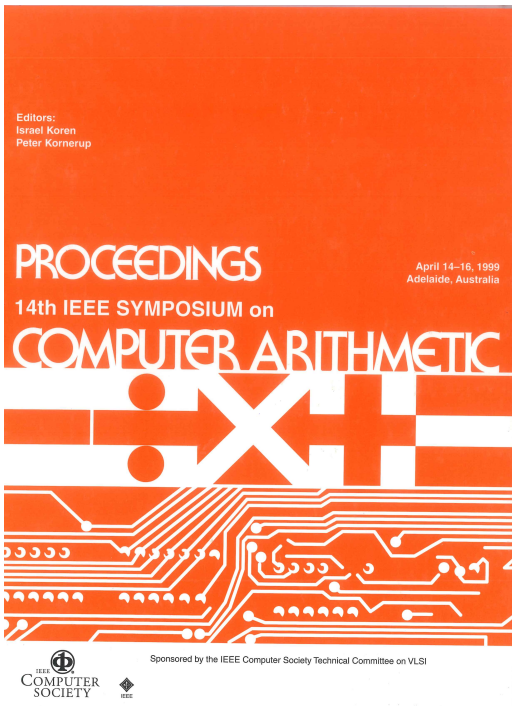
- Approximate computing. Requires more science than computing "exactly": estimate largest errors, average errors, probabilities of failures, make sure branches are taken consistently, ...
- Complex arithmetic that is i. accurate; ii. fast, and iii. overflow/underflow-safe;
- large interchange formats (we will more and more have to deal with data from sensors, previous computations, databases, ...);
- hide divisions (this is not only the compiler's task: for instance one can choose rational approximations that help);
- one day, the quantum computer will be here. Do you want physicists to re-invent the carry-skip adder or the Dadda multiplier for it?
- ...

Time travel



Arith 14, 1999.

Time travel



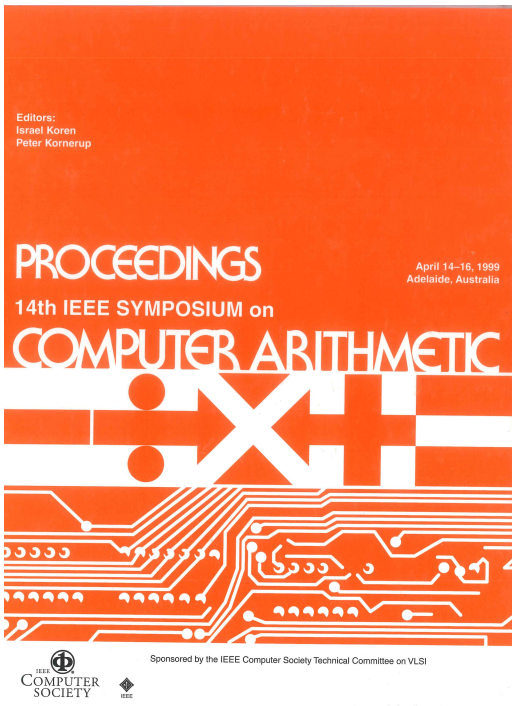
Arith 14, 1999.

Some sessions:

- Addition
- Division
- Divide and Square root
- Multiplication and Rounding
- CORDIC algorithms

Quite a change, isn't it?

Time travel



Arith 14, 1999.

Some sessions:

- Addition
- Division
- Divide and Square root
- Multiplication and Rounding
- CORDIC algorithms

Quite a change, isn't it?

Need to reflect that change (or even anticipate it) in the composition of our PC.