Challenges in Model-Based Simulation

Walid Taha and **Robert Cartwright**
Halmstad University, Halmstad,Sweden
Rice University, Houston, Texas

Precise Simulation is the Future

- CPS is now well-established, but only dimly understood by most information technologists
- Manifestation of *ubiquitous computing* [1988]
- CPS innovation is much harder and more interesting than mere software innovation; CPS transcends the digital vision of IT futurists.
- CPS modeling can be viewed as an ambitious and particularly challenging form of program design *provided*:
  - framework accommodates physical components
  - computational model supports real numbers

Remainder of Talk

- Traditional floating-point computation
  - IEEE standard
  - 32 bit -> 64 bit -? 128 bit
- Basic properties of rationals and real numbers
- Interval arithmetic
- Exact real arithmetic

Floating Point Numbers

What should every software/CPS developer know about floating point arithmetic?

- Floating point is scientific notation with fixed number of digits in the fraction and bound on the minimum and maximum exponents.
- Floating point numbers $\neq$ Real numbers
  - The set of "double-precision" floating point number is finite.
  - Nearly every arithmetic operation involves rounding.
- Floating point representation gives NO bound on accumulated rounding error
- Error bounds on floating point (numerical) algorithms are typically weak and unstated. Many methods introduce discretization error.
- Choosing numerical methods is an art.
- Universal numerical (floating point) algorithms would help but may be unattainable.

Properties of Floating Point Arithmetic

- In real arithmetic

  $10^{40} + 20 - 10 - 10^{40} = 10$

- In double precision IEEE arithmetic

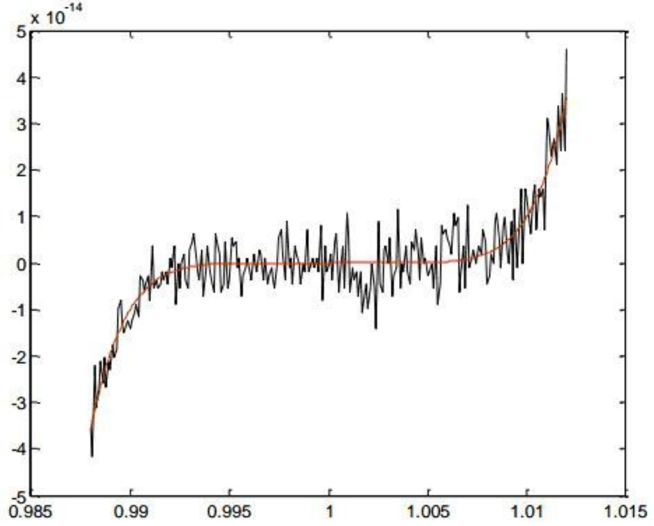
  $10^{40} + 20 - 10 - 10^{40} = 0.$

- Even worse, graphing the polynominal in MatLab (by Cleve Moler in MATLAB)

  $x^7-7x^6+21x^5-35x^4+35x^3-21x^2+7x-1$

  yields the figure in black to the right

- Good numerical software for a particular problem typically avoids these pathologies.



Challenges

- Incorporating support for higher precision floating point arithmetic in new processors
    - IEEE standard for single and double precision floating point arithmetic has made numerical software more portable and more accurate, but given the number of transistors on modern chips, quad (128 bit) precision should be standard.
    - Some hardware support (microcode?) for the efficient implementation of higher precision arithmetic would be even better.
- Improved support for accurate floating point arithmetic in higher level languages. Java does not support directed rounding; it should.

Properties of Rational and Real Numbers

- Rationals
  - pairs of unbounded integers
  - countably infinite set
  - grow very fast
  - not closed under most library functions (sqrt, sin, cos)
- Reals
  - closed under standard library functions (few exceptions)
  - no finite canonical representations
  - common infinite representations are not canonical
  - stream representations are subtle; redundancy required
  - uncountably infinite set

Challenges

- Floating point numbers are rational, but not all rational numbers (even when bounded in size) can be represented exactly as floating point numbers. Rationals generally correspond to real numbers with infinite repeating radix representations:
  - Is there an enhanced form of floating point arithmetic that better supports rationals?
  - How does it compare with higher-precision floating point?
- To what extent can a computer support exact computation with real numbers?

Interval Arithmetic

- A real number x is represented by an interval [x',x''] where x' ≤ x ≤ x'' and x', x'' are rationals. Interval arithmetic is the most widely used *self-validated computation model*.
- In practice, representing x' and x'' as floating point numbers facilitates much better performance, but rounding must be directed.
- Extending basic arithmetic to intervals is easy, but comparison operations may be indeterminate, e.g., [1,2] < [1.5, 2.5], which is treated as an aborting error.
- Interval arithmetic often over-estimates the error, but it particularly misses canceling dependencies, *e.g.*, x – x where x is bound to an interval [a,b] where a < b. Then x - x = [a – b, b – a] rather than 0.
- The results of computations that use the values of variables more than once can be improved by splitting the interval into two equal pieces, doing the computation for each piece and taking the union of the results.

Interval Arithmetic cont.

- Splitting example: x – x where x = [-1, 1].
  - Without splitting, x – x = [-2, 2].
  - With splitting at 0, we get x – x = [-1, 1] because the result is [-1, 1] if x = [0, 1] and [-1, 1] if x = [-1, 0].
  - Through repeated splitting, the interval result for x-x can be made as small as desired (> 0).
- Many interval analysis algorithms perform repeated interval splitting until a particular error bound is achieved; these algorithms are similar in character to adaptive quadrature.

Challenges

- Are there good strategies for combining splitting and higher precision?
- More elaborate models that propagate correlation information like affine arithmetic exist. They are much more complex to implement and have been less widely used. What are the trade-offs between interval arithmetic and more elaborate alternatives?
- Self-validated computation models could play a very important role in precise CPS simulation, but this research area has received comparatively little attention among researchers in numerical computation.

Exact Real Arithmetic

- A countable subset of the reals called the *constructive* reals can be implemented on the computer. Moreover, every real number with a computable radix expansion is constructive. The constructive reals form a countable subset of the real numbers. But:
  - Radix representations require redundancy (more digit values than the base). Example: add 0.44444 … and 0.55555 ..... In the computed result, what is the value of the digit preceding the decimal point?
  - Comparison (<=) diverges if the two numbers are equal.
  - The overhead involved in computing with exact reals is very large. Constructive real number can be represented in a variety of ways: as a lazy stream of digits, as an lazy continued fraction, or as a computable function from rational tolerances to rational approximations. Other representations are possible. On intriguing scheme might be to use iterated interval arithmetic, doubling the floating point precision on each iteration.

Challenges

- Can parallelism (multi-core, many-core, distributed memory) be used to effectively speed up exact real computation in general?
- Exact real arithmetic potentially eliminates round-off error. But what about discretization error? Can we bound it? Enclosure methods try to do this.

Summary

- Accurately simulating physical components is the new frontier for computing research
- The CPS design process is amenable to the same hierarchical approach to system development as software provided we can develop satisfactory simulation frameworks for physical components.
- Advanced approaches to supporting accurate computation involving real numbers are still in their infancy and we do not yet know what approaches will be most practical.
- More interdisciplinary cooperation between hardware designers, language researchers and CPS researchers would be very helpful.