Verification of Numerical Results, using Posits, Valids, and Quires

Gerd Bohlender, Karlsruhe Institute of Technology

CoNGA – Singapore, March 28, 2018

Outline

- Floating-Point Arithmetic
- Pure vs. Applied Mathematics, Problems
- Posits
- Interval Arithmetic, Valids
- Exact Dot Products, Quires
- Numerical Algorithms with Verified Results
- Conclusion

IEEE Floating-Point Arithmetic

- IEEE Standard 754, issued 1985, revised 2008
- Widely adopted industry standard
- Easier interchange of data and programs
- Exception handling with NaNs etc. has advantages and disadvantages
- Results may differ on different platforms
- Results of algorithms may be unreliable

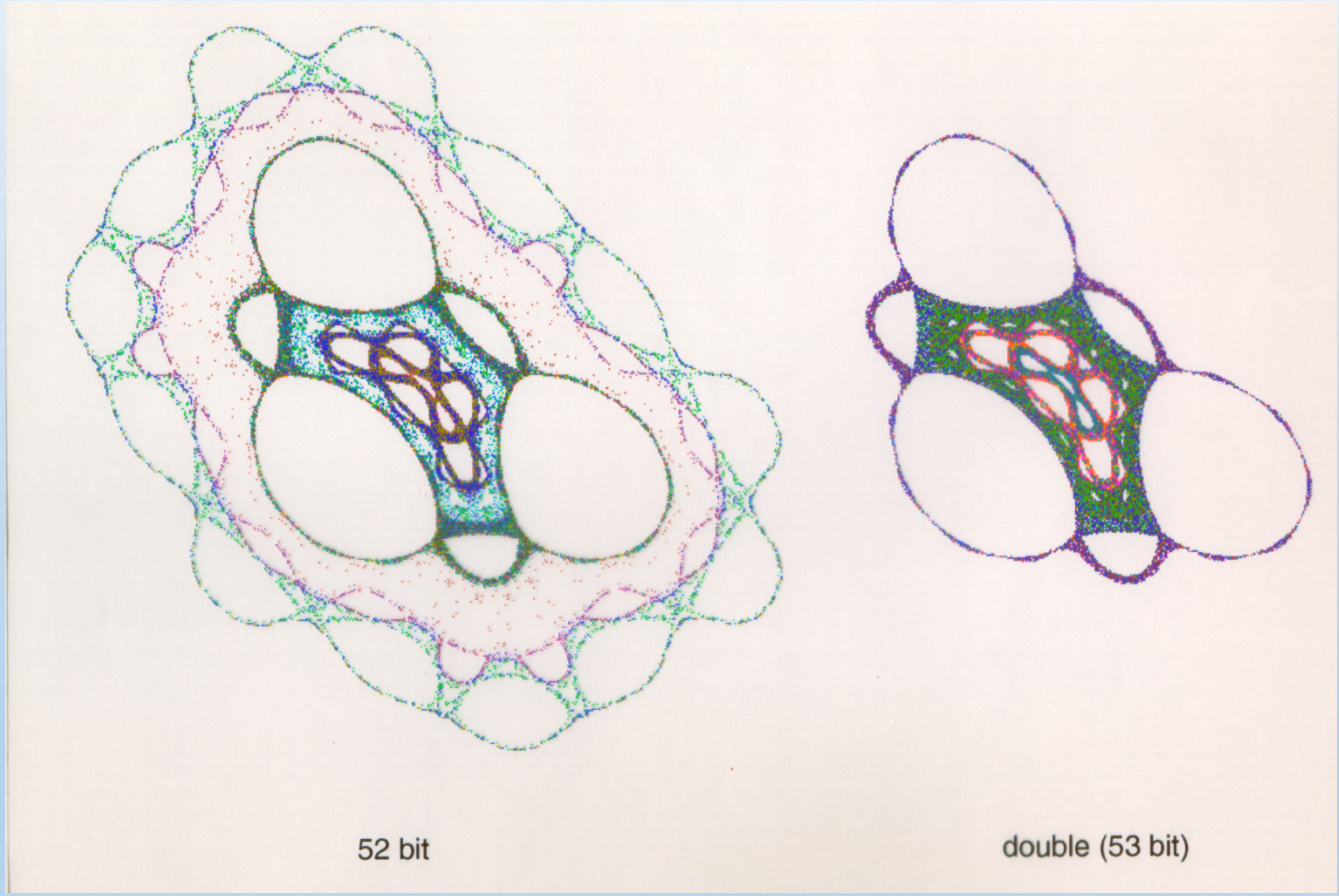
Pure vs. Applied Mathematics

- **Pure Mathematics**: for all real numbers x, y, z the associative law (x+y)+z = x+(y+z) holds
- **Applied Mathematics / Computed Results**: x is an approximate solution of the given problem with an error estimation eps – neglecting higher order terms and influence of rounding error
- Strong contrast between these concepts

Problems with Floating-Point Arithmetic

- **Ex. 1:** Evaluate $p^4 - 4\,p^2q^2 + q^4$ for $p = 665857, q = 470832$
- Result in double precision: $-3.3554432 \cdot 10^7$
- Correct result: **1**
- **Ex. 2:** Evaluate sum $10^{20} + 1 - 10^{20}$
- Result in double precision: **0**
- Correct result: **1**
- **Ex. 3:** Linear system 2x2, solution requires 7 arithmetic operations
- All digits of result are wrong

Fractal, Influence of Arithmetic



- Identical fractal
- Identical number of steps
- Identical coloring scheme
- Different floating-point formats

- Which graph is better?

Problems with Floating-Point Arithmetic

- Supercomputers don’t solve such trivial problems
- Tianhe-2 has 34 Petaflops – what does this mean?
- **Imagine**: print all computed values (in small font on both sides of paper)
- Height of pile of paper for one second of calculations ???

Problems with Floating-Point Arithmetic

- Supercomputers don't solve such trivial problems
- Tianhe-2 has 34 Petaflops – what does this mean?
- **Imagine**: print all computed values (in small font on both sides of paper)
- Height of pile of paper for one second of calculations ???
- 3.4 million kilometres = 9 x distance from earth to moon !!!
- How reliable are these results? Who can check them?
- Special tools on computer are needed!

Posits, a New Floating-Point Format

- Flexible trade-off between exponent bits (regime + exponent) and mantissa bits
- Higher accuracy than IEEE 754 for numbers close to 1
- Special representation for $\pm\infty$
- No other special cases like NaN, gradual underflow
- Better results in typical applications [Gustafson, Yonemoto]
- However, results may still be unreliable

Valids, Interval Bounds for Values

- Pair of equal-size posits for lower and upper bounds of value, e.g. $a \in [a_{low}, a_{high}]$
- Each posit ends in a ubit, indicating closed bounds [] or open bounds () or a mix of both (], [)
- Representation of connected sets of real numbers. These can be closed, open, half-open, bounded, or unbounded.
- Arithmetic for valids is algebraically closed under elementary operations and many standard functions, i.e. for all valids A, B, and all relevant functions F, $A + B, A - B, A * B, \frac{A}{B}, F(A)$ are valids
- Calculus is free of exceptions

Naïve Application of Intervals / Valids

- Interval evaluation $F(X)$ of a function $f(x)$: replace all floating-point operations by interval operations
- Usually **overestimates** the range of the function on interval X
  $range(f, X) = \{f(x) | x \in X\}$
- **Example**: $f(x) = x - x \quad (= 0),\ X = [0,1]$,
  $range(f, X) = [0,0]$, but $F(X) = [0,1] - [0,1] = [-1,1]$
- **Help**: subdivide interval (under some conditions):
  $F([0, 0.5]) \cup F([0.5,1]) = [-0.5,0.5] \cup [-0.5,0.5] = [-0.5,0.5]$
- Further subdivision: nearly arbitrarily high accuracy
- **But**: higher costs, less efficiency, especially in multidimensional cases

Better Application of Intervals / Valids

- Do **not** compute everything with valids – too expensive, too inaccurate
- Transform expressions to form which is better to evaluate
- Avoid (if possible) variables to occur more than once
- Apply exact dot products / quires (next slides)
- Use better algorithms (next slides)
- Do major part of calculation in floating-point and only remainders (between exact and computed values) in interval arithmetic

Exact Dot Products, Quires

- Generalization of fused multiply and add
- Fused sum $\sum_{i=1}^{n} a_i$ and fused dot product $\sum_{i=1}^{n} a_i \ * \ b_i$ computed with only one final rounding for any floating-point numbers / posits a_i, b_i
- Round result to nearest posit $p = round\ (\sum_{i=1}^{n} a_i \ * \ b_i)$
- Round result to enclosing valid $v = [v_{low}, v_{high}] = roundout\ (\sum_{i=1}^{n} a_i \ * \ b_i)$
- Store intermediate values exactly, e.g.

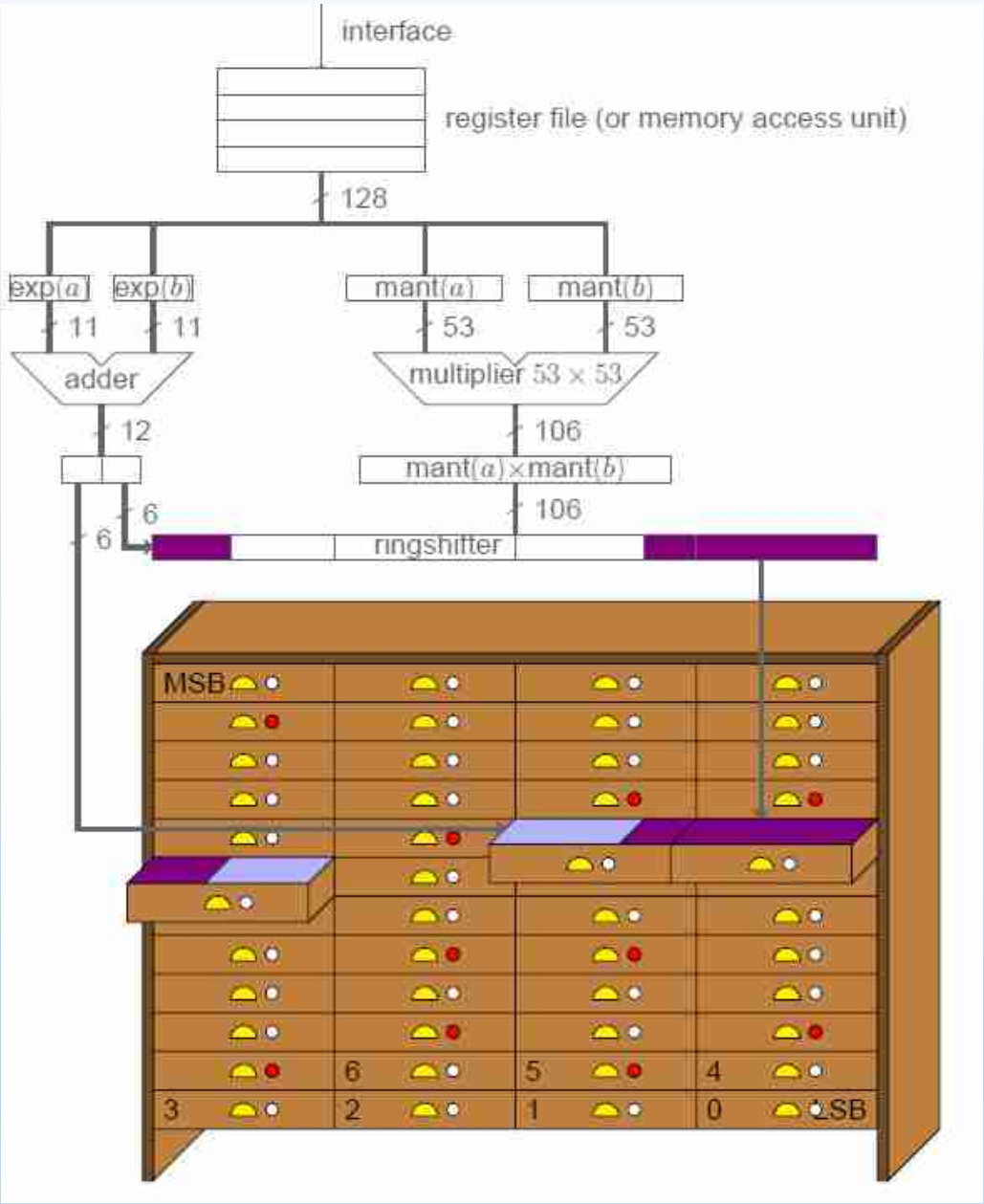
  $s = \sum_{i=1}^{n} a_i \ * \ b_i\ ,\ \ s$ exact

  $s_1 = round(s), \quad s_2 = round(s \ - s_1)$

  „quadruple precision“ result as sum of two posits

Software and Hardware Solutions for Quires

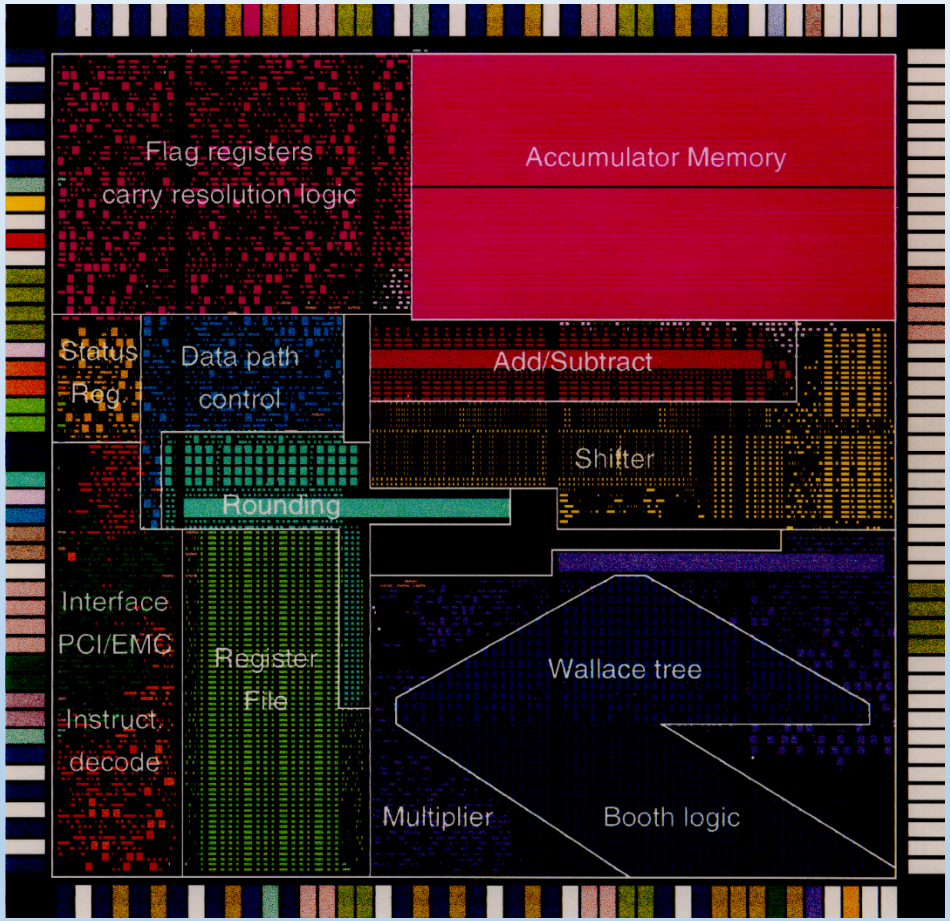
- Many options available for high performance implementations
- Storage in long fixed-point accumulator; pipelined parallelized operation in one clock cycle
- Quires: exact arithmetic operations are much faster than memory access
- Alternative implementations: e.g. sum and remainder operation [Bo 1978], [Bo 1990], many recent publications e.g. by [Rump]



Dot Product symbolized by a drawer

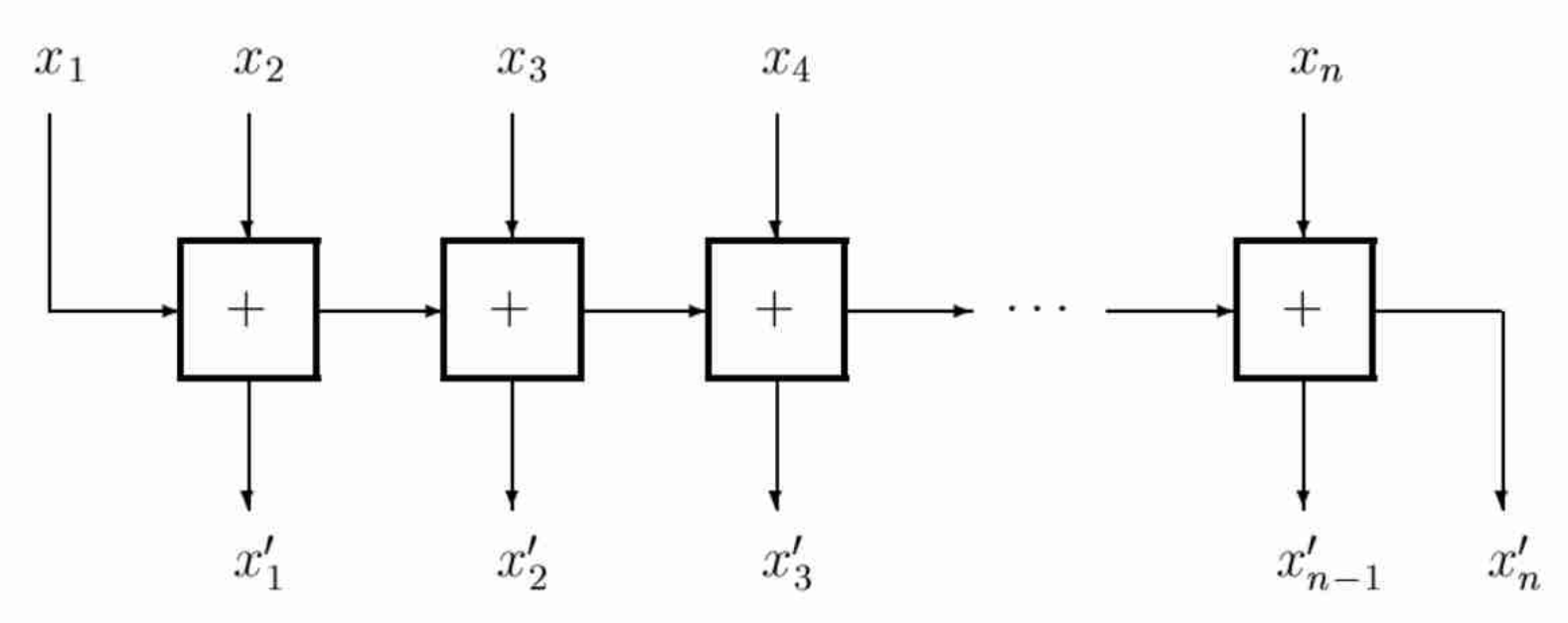
- Only 3 words have to be computed in any accumulation
- Other words may be affected by carries or borrows
- These words are not modified in storage, but flags (red / white dots) are used to store this information

Coprocessor XPA 3233 for Exact Dot Products



- Prototypes built in 1995
- Coprocessor for Intel processor
- Faster than conventional non-exact dot products
- Intended as demonstration of technology
- Similar development 2017 presented on ARITH 24 in London [Koenig / Berkeley]

Alternative Implementation of Quire

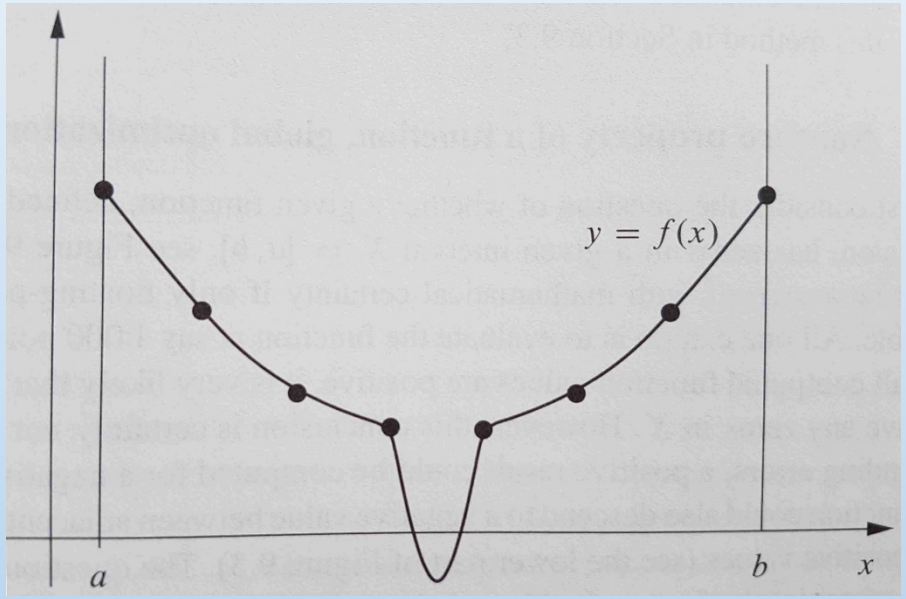


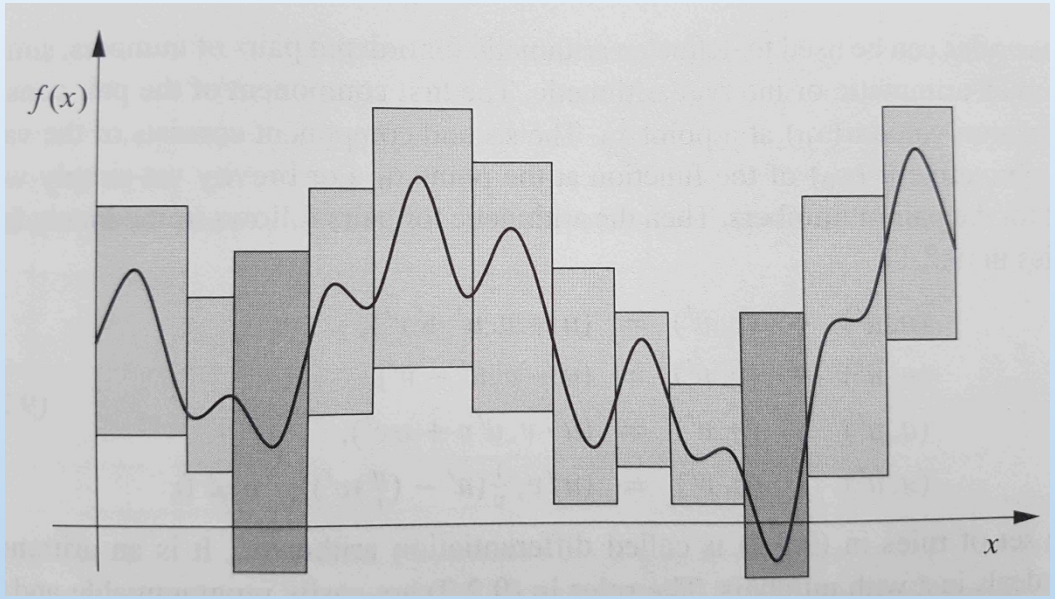
- Repeated sum with exact remainder [Bo 1978], graphics taken from [Bo 1990]
- $x_1^{(k)}$ converges very quickly to quire result (rounded value of exact sum)
- Pipelining possible

Validated Results of Numerical Algorithms

- Classical numerical algorithms compute approximations of result
- Possibly error estimations, but no rigorous bounds
- New algorithms should compute
  - Rigorous validated bounds for result
  - Proof of existence of solution
  - (If possible) proof of uniqueness of solution
- This can be achieved using valids (interval arithmetic) and quires (exact dot products)

Special Algorithms for Valids and Quires



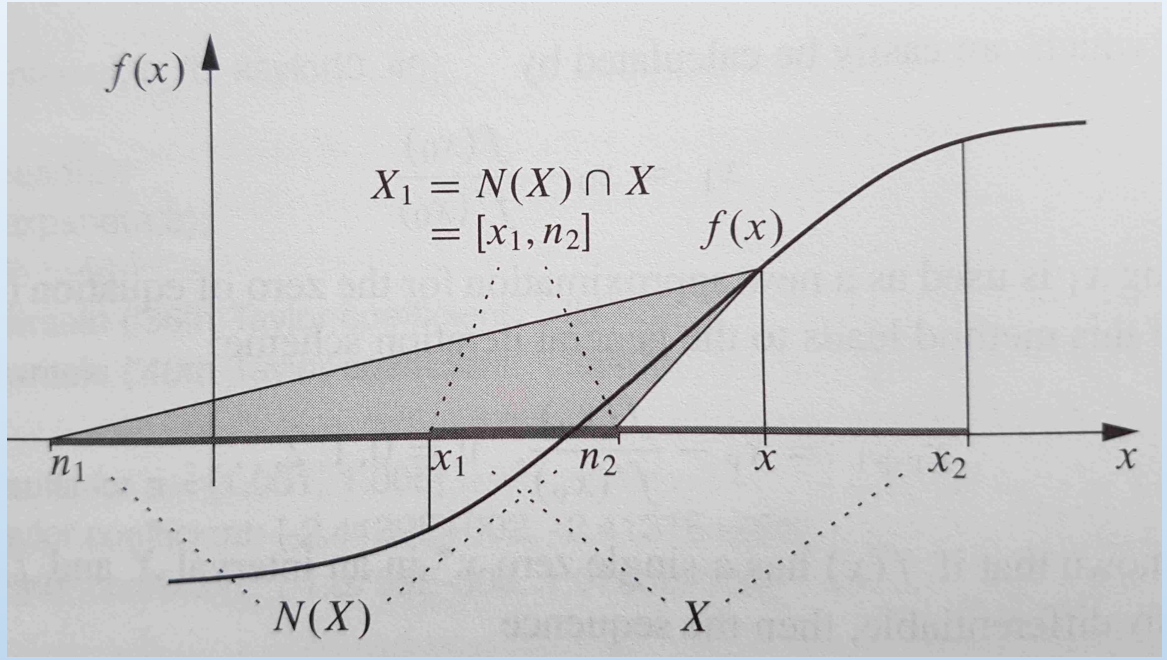


Posit / floating-point evaluation at many points:
no certainty about zeroes in range

Valid / interval evaluation:
certainty about subintervals not containing zeroes

Global optimization may use this information for fast elimination of regions

Newton Method with Valids



- If $N(X)$ is a subset of X, this proves that $f(x)$ has exactly one zero in X
- If intersection $N(X) \cap X$ is empty, this proves non-existence of zero
- Computation of a sequence X_i of more and more accurate valids containing the exact zero (provided that intersection $N(X) \cap X$ is not empty)

Validated Linear System Solver

- Solve linear system $\boldsymbol{Ax} = \boldsymbol{b}$ with unknown exact solution x^*
- Compute approximate inverse R, approximate solution $x^{\sim}$ using posit arithmetic
- Do **not** compute an interval inclusion of x^* directly
- But compute an interval inclusion E of the **error** $e = x^* - x^{\sim}$, using valids, giving much higher accuracy
- Compute residual $r = b - A\,x^{\sim}$ and sequence of intervals
  $$E_{i+1} = (I - R\,A)\,E_i + R\,r$$
- Computation of these expressions in conventional floating-point arithmetic could be worthless (**cancellation** in subtractions!)
- Use valids to obtain interval bounds and quires to achieve high accuracy, then under some conditions $x^* \in x^{\sim} + E_i$ (valid containing result)

Linear System Solver Using Valids and Quires

- R = inverse (A) // posit arithmetic
- X~ = R b, B = I – R A, r = b – A X~ // valid and quire
- E = R r // valid and quire
- Z = E, i = 0 // valid
- **Repeat**

  i = i + 1

  Y = blow (E, eps) // valid, make interval wider

  E = B Y + Z // valid and quire

  Until E **is subset of interior of** Y **or** i = 10 // valid

Interpretation of Result

- All operations can be implemented easily with valids and quires
- Test "E **is subset of interior of** Y" is easy to perform, some comparisons of bounds with < and > instead of $\leq$ and $\geq$
- If E **is subset of interior of** Y, this is a rigorous proof that
  - The matrix A is not singular
  - The linear system Ax = b has a unique solution
  - The true solution x^* of the linear system is contained in the interval / valid vector X~ + E
- Otherwise, a more refined algorithm is needed (e.g. Rump method, represent R by value and remainder $R = R_1 + R_2$ using quires); possibly A is singular, i.e. no solution exists

Evaluation of Polynomials and Expressions

- May be transformed in a system of linear equations
- Polynomial evaluation in form of Horner scheme

$$p(x) = ((\dots(a_n * x + a_{n-1}) * x + \dots) * x + a_1) * x + a_0$$

- Introduce auxiliary variables

$$p_n = a_n, \quad p_{n-1} = p_n * x + a_{n-1}, \dots, \quad p_0 = p_1 * x + a_0$$

- Linear system for unknowns p_i
- Value of polynomial is $p(x) = p_0$
- Similar transformation for arithmetic expressions

Application, Example Resonances in Turbines

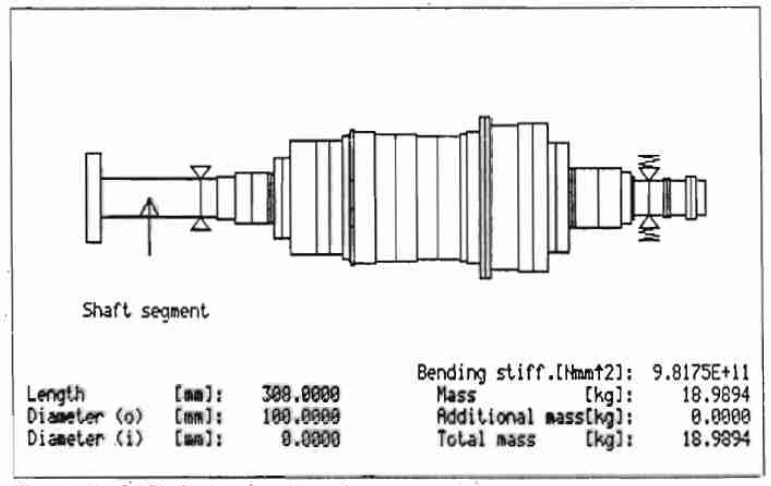


Figure 1. CAD of rotors and bearings

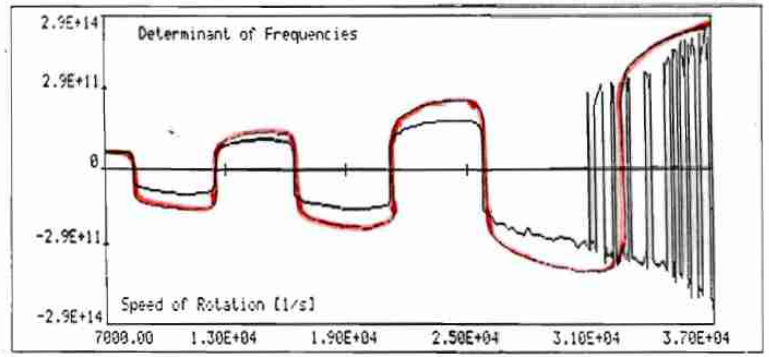


Figure 2. Residual frequency determinant

- Imprecise modelling of eigenfrequencies of rotor (in black)
- High precision modelling (in red)

- Turbine exploded in Leipzig (Germany), killing 4 persons and injuring 6

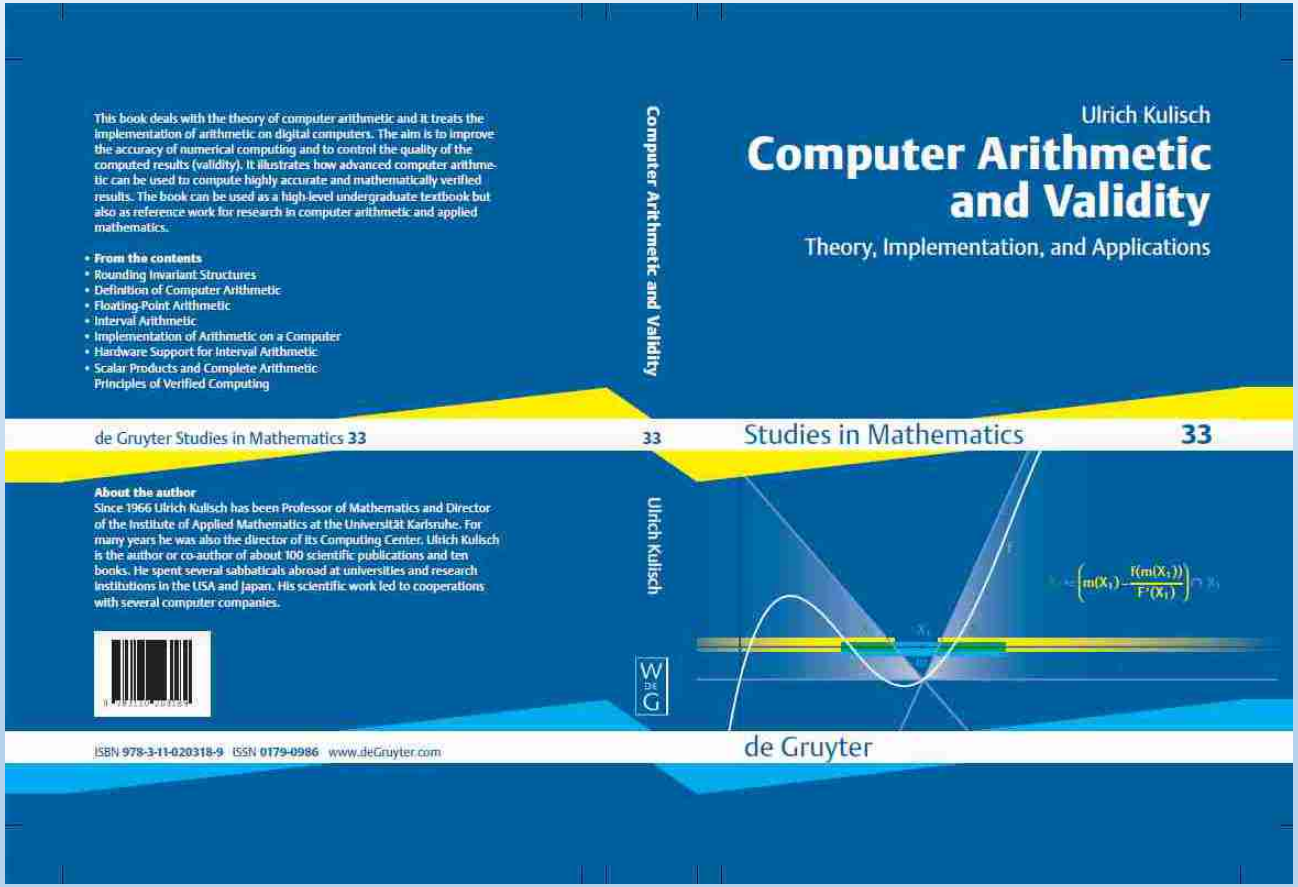
LEIPZIG

Schwarze Rauchwolken über der Baustelle der Stadtwerke in der Eutritzscher Straße

Kraftwerks-Turbine zerfetzt: vier Tote

Beim Einjustieren zerriß das Getrieb. / Öl entzündete sich / Sechs Verletzte / Ursache noch unklar

Kulisch's Book 2013



- Computer arithmetic, mathematical background
- Interval arithmetic
- Design of hardware for e.g. exact dot products / quires
- Numerical algorithms with verified results

- Prof Kulisch will celebrate his 85th birthday in May

Conclusion

- Posits bring higher accuracy than IEEE floating-point operations
- Valids add the security of interval operations
- Quires are essential for the solution of linear systems and many other standard problems of mathematics
- Implementations of valids and quires with highest performance are required, many algorithms and hardware designs are available
- Extended programming environments are required for the formulation of algorithms, e.g. C-XSC
- Verified solutions for applications may require a modification of the underlying model and according new implementation