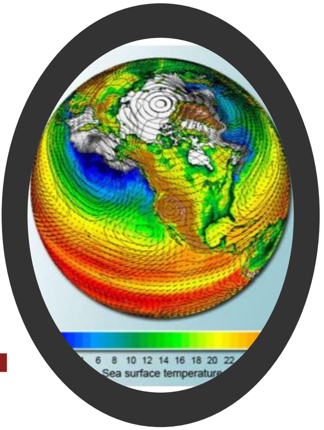
SCHOOL OF COMPUTING
UNIVERSITY OF UTAH

SOFTWARE ANALYSIS RESEARCH LABORATORY

ANALYSIS AND SYNTHESIS OF FLOATING-POINT ROUTINES

Zvonimir Rakamarić

FLOATING-POINT COMPUTATIONS ARE UBIQUITOUS



CHALLENGES

- FP is “weird”
  - Does not faithfully match math (finite precision)
  - Non-associative
  - Heterogeneous hardware support
- FP code is hard to get right
  - Lack of good understanding
  - Lack of good and extensive tool support
- FP software is large and complex
  - High-performance computing (HPC) simulations
  - Machine learning

FP IS WEIRD

- Finite precision and rounding
  - x + y in reals ≠ x + y in floating-point
- Non-associative
  - (x + y) + z ≠ x + (y + z)
  - Creates issues with
    - Compiler optimizations (e.g., vectorization)
    - Concurrency (e.g., reductions)
- Standard completely specifies only +, -, *, /, comparison, remainder, and square root
  - Only recommendation for some functions (trigonometry)

FP IS WEIRD cont.

- Heterogeneous hardware support
  - x + y*z on Xeon ≠ x + y*z on Xeon Phi
    - Fused multiply-add
  - Intel's online article "Differences in Floating-Point Arithmetic Between Intel Xeon Processors and the Intel Xeon Phi Coprocessor"
- Common sense does not (always) work
  - x "is better than" log(e^x)
  - (e^x-1)/x "can be worse than" (e^x-1)/log(e^x)
    - Error cancellation

FLOATING-POINT NUMBERS

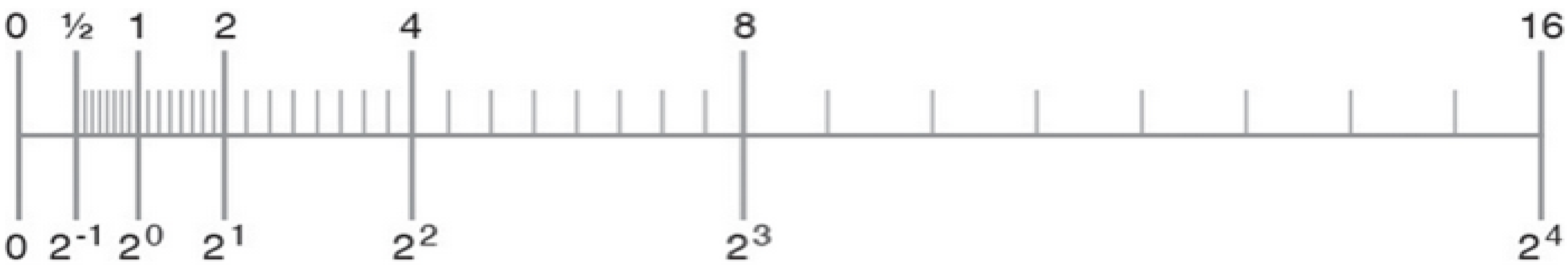
- IEEE 754 standard
- Sign (s), mantissa (m), exponent (exp):

$$(-1)^{s} * 1.m * 2^{exp}$$

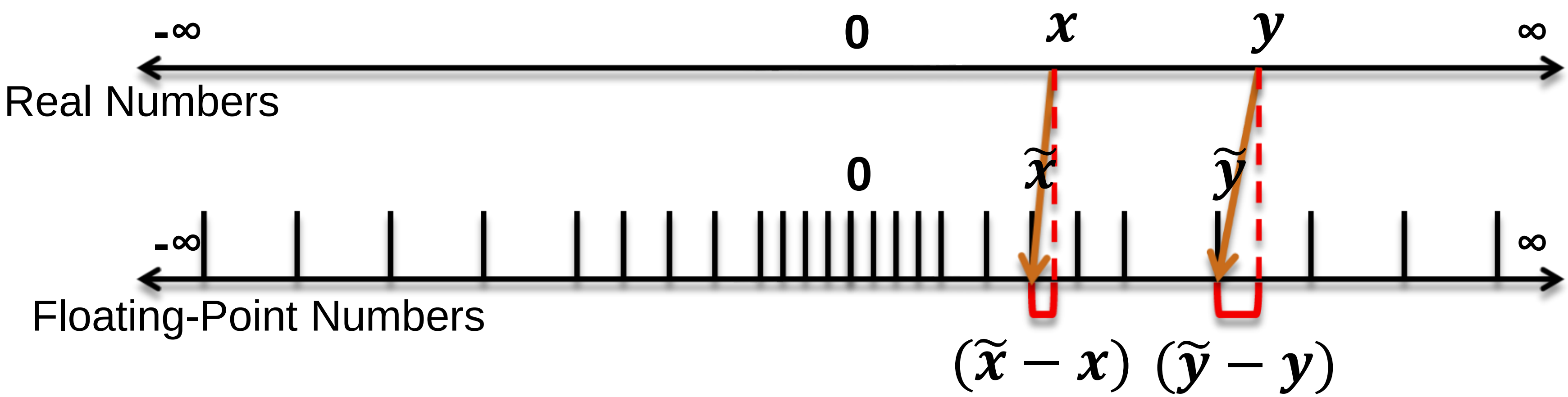
- Single precision: 1, 23, 8 bits
- Double precision: 1, 52, 11 bits

FLOATING-POINT NUMBER LINE

- 3 bits for precision
- Between any two powers of 2, there are $2^3 = 8$ representable numbers



ROUNDING IS SOURCE OF ERRORS



FLOATING-POINT OPERATIONS

- First normalize to the same exponent
  - Smaller exponent -> shift mantissa right
- Then perform the operation
- Losing bits when exponents are not the same!

UTAH FLOATING-POINT TEAM

1. Ganesh Gopalakrishnan (prof)
2. Zvonimir Rakamarić (prof)
3. Ian Briggs (staff programmer)
4. Mark Baranowski (PhD)
5. Rocco Salvia (PhD)
6. Shaobo He (PhD)
7. Thanhson Nguyen (PhD)

Alumni: Alexey Solovyev (postdoc), Wei-Fan Chiang (PhD), Dietrich Geisler (undergrad), Liam Machado (undergrad)

RESEARCH THRUSTS

Analysis

- Verification of floating-point programs
- Estimation of floating-point errors
  1. Dynamic
     - Best effort, produces lower bound (under-approximation)
  2. Static
     - Rigorous, produces upper bound (over-approximation)

Synthesis

- Rigorous mixed-precision tuning

Constraint Solving

- Search-based solving of floating-point constraints
- Solving mixed real and floating-point constraints

RESEARCH THRUSTS

Analysis

- Verification of floating-point programs
- Estimation of floating-point errors
  1. Dynamic
     - Best effort, produces lower bound (under-approximation)
  2. **Static**
     - **Rigorous, produces upper bound (over-approximation)**

Synthesis

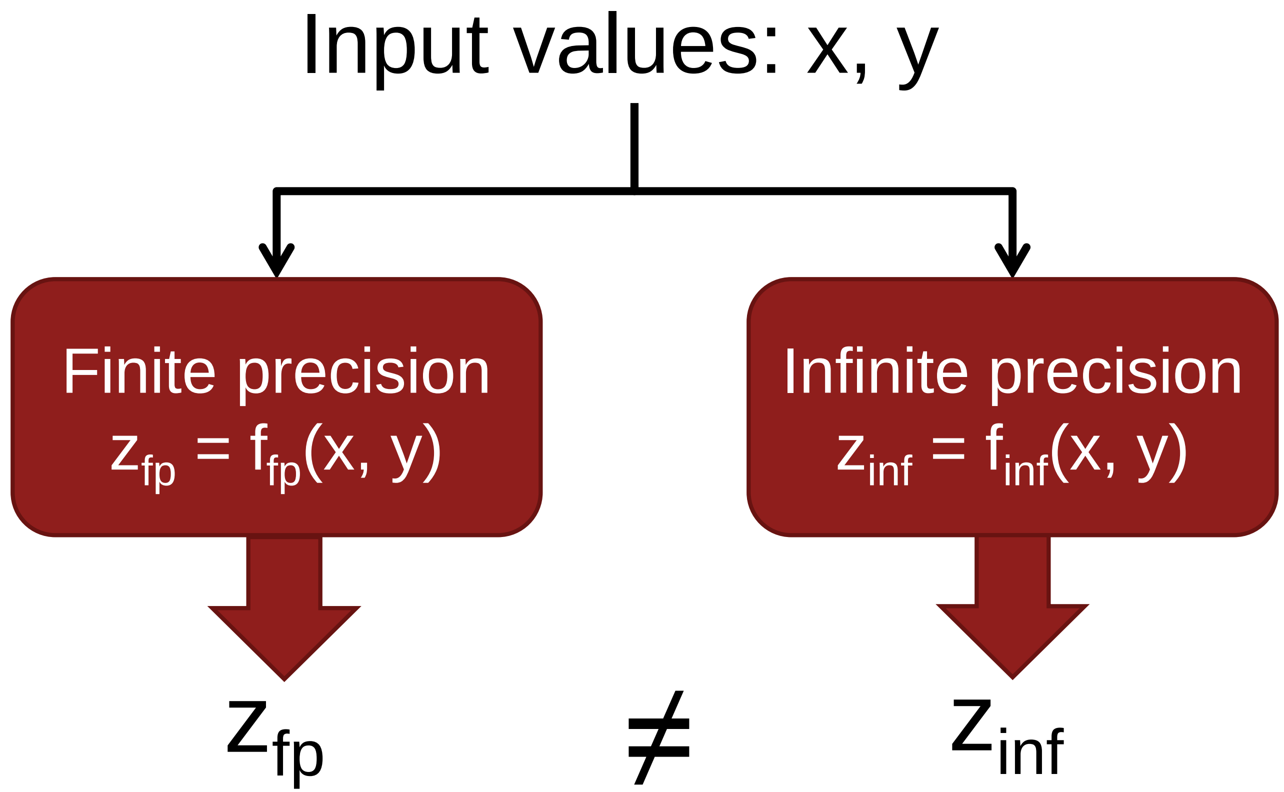
- **Rigorous mixed-precision tuning**

Constraint Solving

- **Search-based solving of floating-point constraints**
- Solving mixed real and floating-point constraints

ERROR ANALYSIS

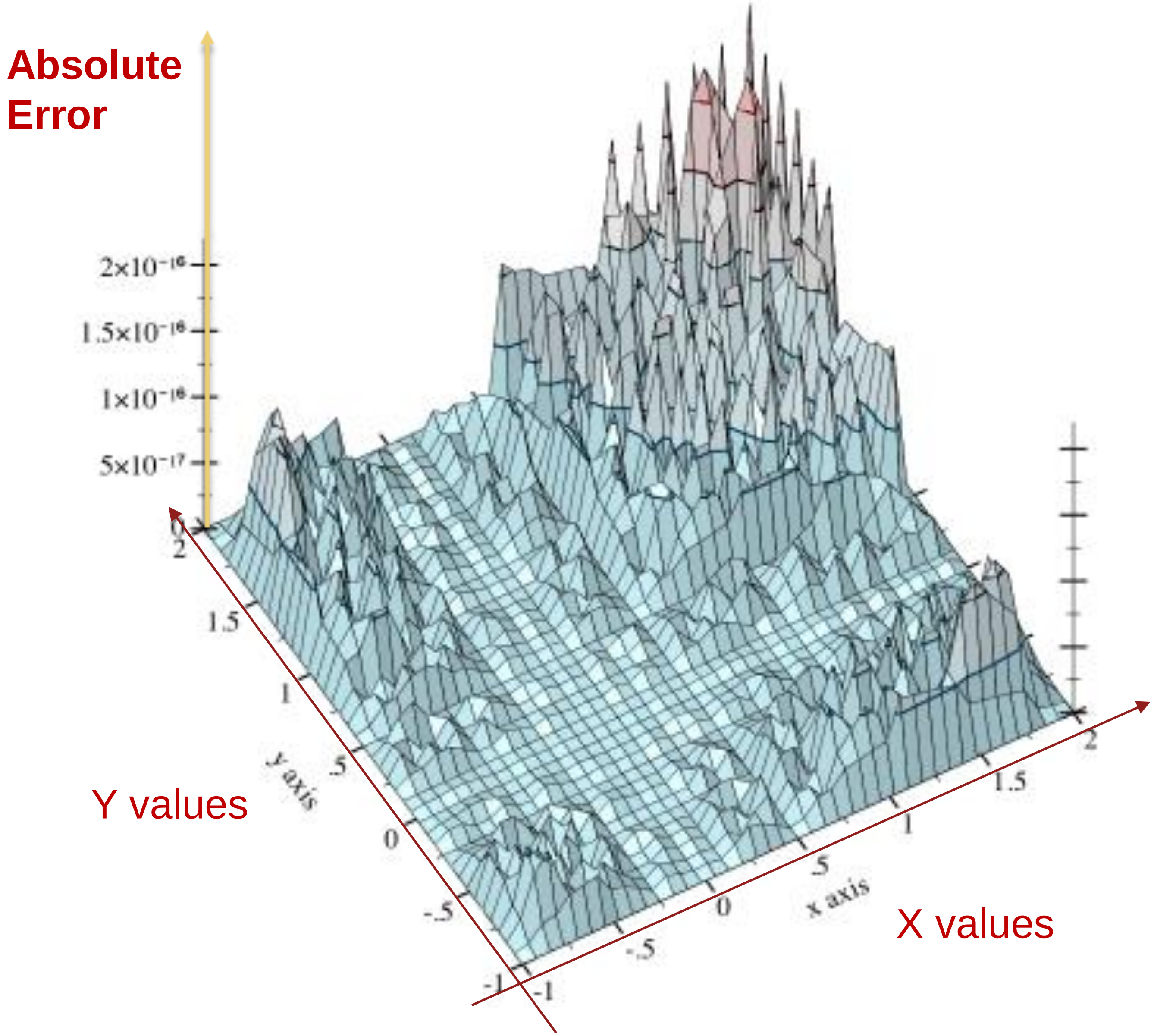
FLOATING-POINT ERROR



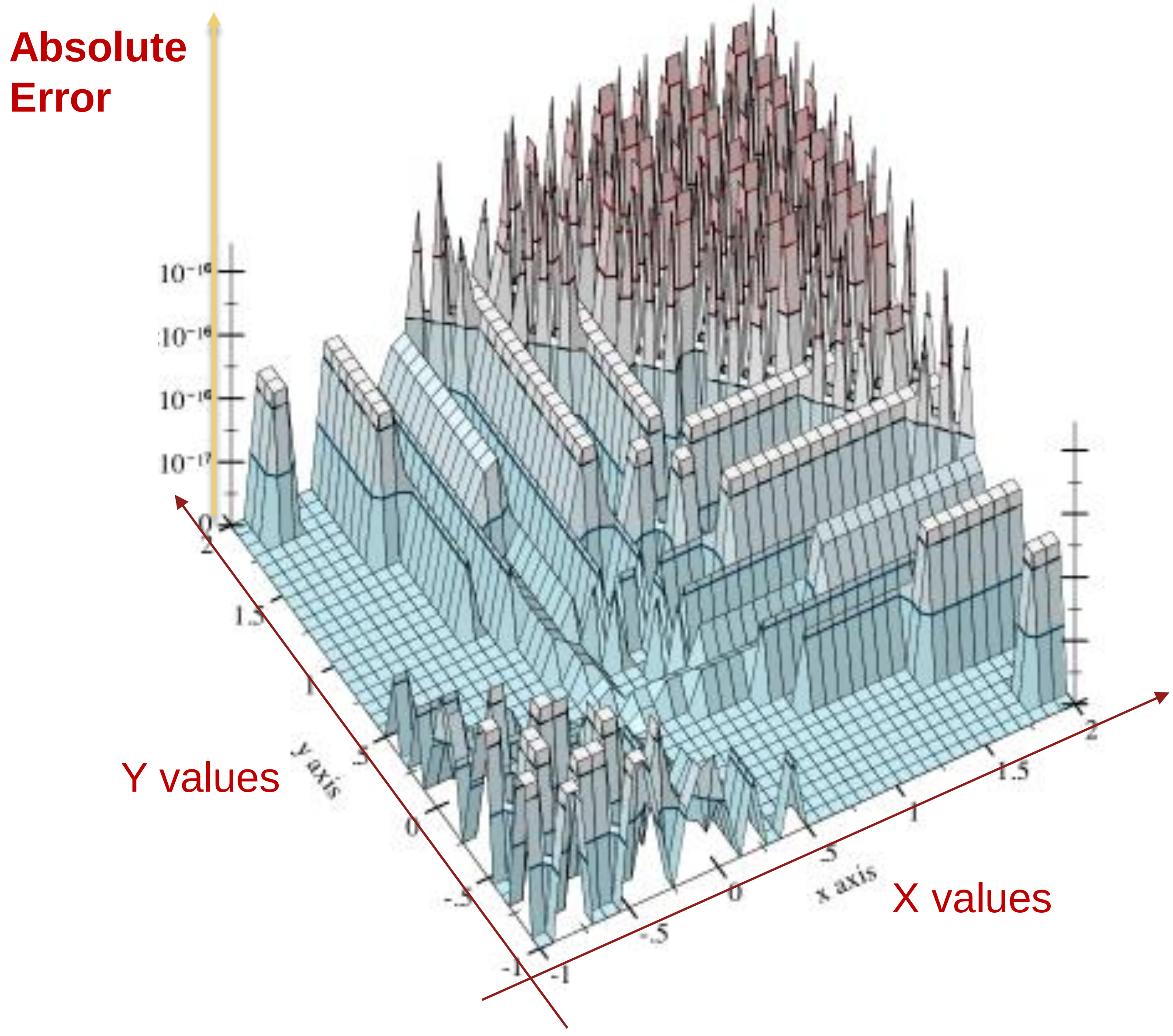
Absolute error: $| z_{fp} - z_{inf} |$

Relative error: $| (z_{fp} - z_{inf}) / z_{inf} |$

ERROR PLOT FOR MULTIPLICATION



ERROR PLOT FOR ADDITION



USAGE SCENARIOS

- Reason about floating-point computations
- Precisely characterize floating-point behavior of libraries
- Support performance-precision tuning and synthesis
- Help decide where error-compensation is needed
- “Equivalence” checking

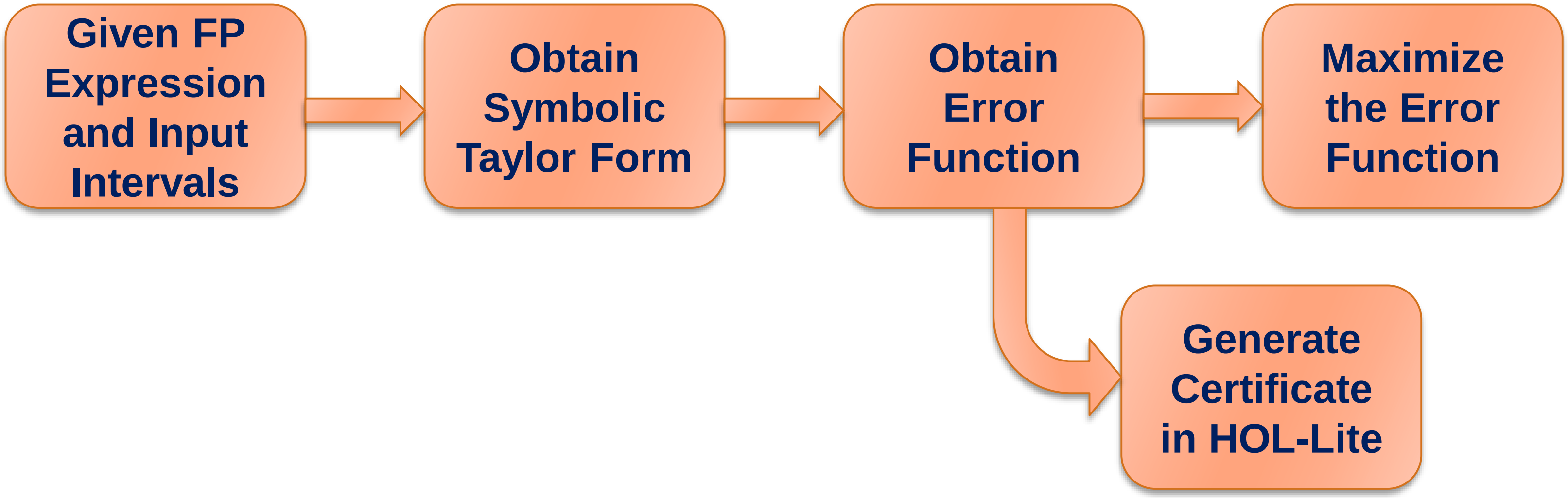
STATIC ANALYSIS

http://github.com/soarlab/FPTaylor

CONTRIBUTIONS

- Handles non-linear and transcendental functions
- Tight error upper bounds
  - Better than previous work
- Rigorous
  - Over-approximation
  - Based on our own rigorous global optimizer
  - Emits a HOL-Lite proof certificate
    - Verification of the certificate guarantees estimate
- Tool called FPTaylor publicly available

FPTaylor TOOLFLOW



IEEE ROUNDING MODEL

Consider $op(x, y)$ where x and y are floating-point values, and op is a function from floats to reals

IEEE round-off errors are specified as

$$op(x, y) \cdot \left(1 + e_{op}\right) + d_{op}$$

For normal values For subnormal values

Only one of e_{op} or d_{op} is non-zero:

$\left|e_{op}\right| \leq 2^{-24}, \left|d_{op}\right| \leq 2^{-150}$ (single precision)

$\left|e_{op}\right| \leq 2^{-53}, \left|d_{op}\right| \leq 2^{-1075}$ (double precision)

ERROR ESTIMATION EXAMPLE

- Model floating-point computation of $E = x/(x+y)$ using reals as

$$\tilde{E} = \frac{x}{(x+y)\cdot(1+e_1)}\cdot(1+e_2)$$

$$|e_1| \leq \epsilon_1, |e_2| \leq \epsilon_2$$

- Absolute rounding error is then $\left|\tilde{E} - E\right|$
- We have to find the max of this function over
  - Input variables x, y
    - Exponential in the number of inputs
  - Additional variables e_1, e_2 for operators
    - Exponential in floating-point routine size!

SYMBOLIC TAYLOR EXPANSION

- Reduces dimensionality of the optimization problem
- Basic idea
  - Treat each e as “noise” (error) variables
  - Now expand based on Taylor’s theorem
    - Coefficients are symbolic
    - Coefficients weigh the “noise” correctly and are correlated
- Apply global optimization on reduced problem
  - Our own parallel rigorous global optimizer called Gelpia
  - Non-linear reals, transcendental functions

ERROR ESTIMATION EXAMPLE

$$\tilde{E} = \frac{x}{(x+y)\cdot(1+e_1)}\cdot(1+e_2)$$

expands into

$$\tilde{E} = E + \frac{\partial \tilde{E}}{\partial e_1}(0) \times e_1 + \frac{\partial \tilde{E}}{\partial e_2}(0) \times e_2 + \mathrm{M}_2$$

where M_2 summarizes the second and higher order error terms and $|e_0| \leq \epsilon_0, |e_1| \leq \epsilon_1$

Floating-point error is then bounded by

$$\left|\tilde{E} - E\right| \leq \left|\frac{\partial \tilde{E}}{\partial e_1}(0)\right| \times \epsilon_1 + \left|\frac{\partial \tilde{E}}{\partial e_2}(0)\right| \times \epsilon_2 + \mathrm{M}_2$$

ERROR ESTIMATION EXAMPLE

- Using global optimization find constant bounds
- M_2 can be easily over-approximated
- Greatly reduced problem dimensionality
  - Search only over inputs x, y using our Gelpia optimizer

$$\forall x, y. \left|\frac{\partial \tilde{E}}{\partial e_1}(0)\right| = \left|\frac{x}{x+y}\right| \leq U_1$$

$$\left|\tilde{E} - E\right| \leq \boxed{\left|\frac{\partial \tilde{E}}{\partial e_1}(0)\right|} \times \epsilon_1 + \left|\frac{\partial \tilde{E}}{\partial e_2}(0)\right| \times \epsilon_2 + M_2$$

ERROR ESTIMATION EXAMPLE

- Operations are single-precision (32 bits)

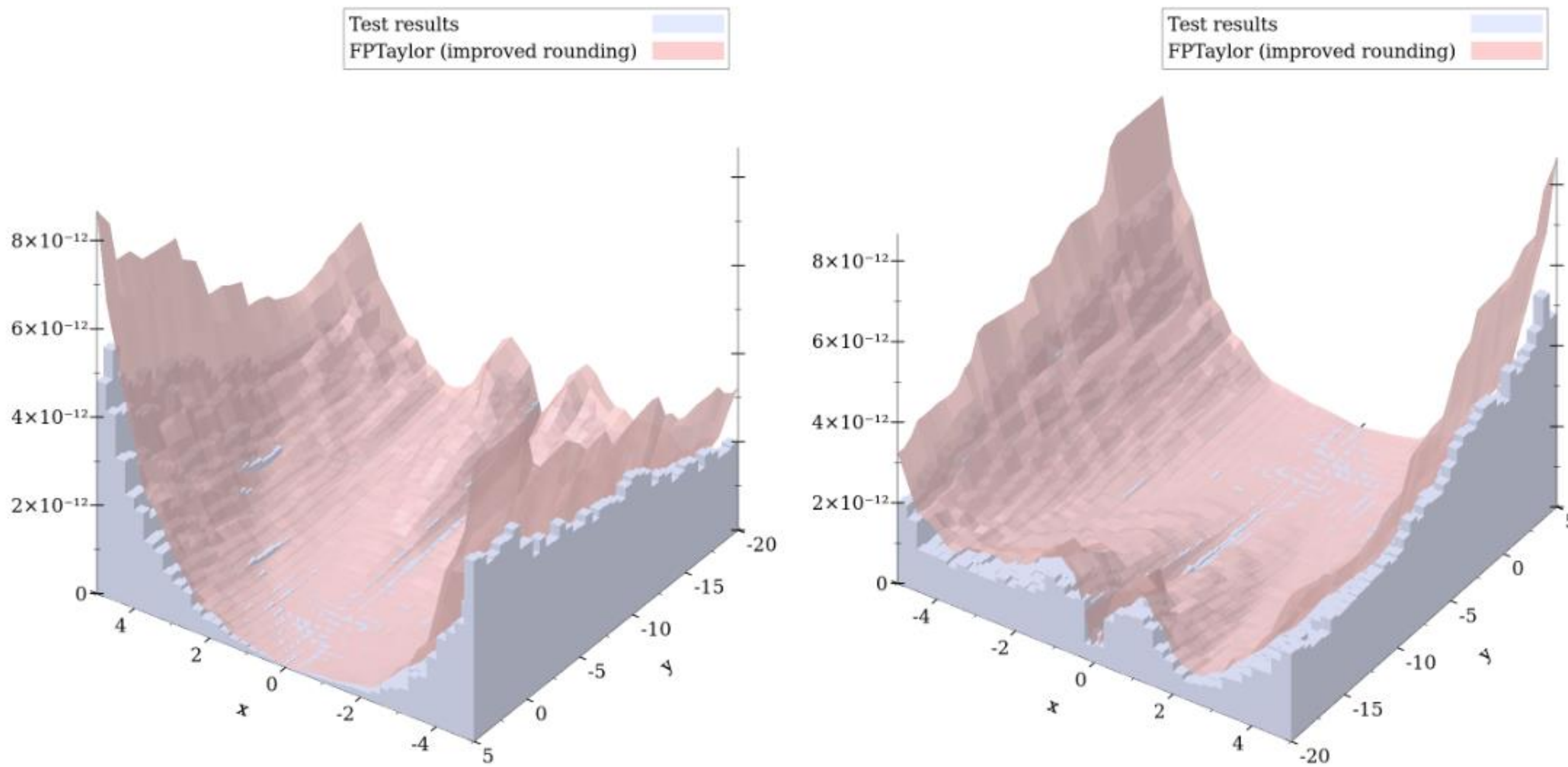
$$\left|\tilde{E} - E\right| \leq U_1 \times \epsilon_{32-bit} + U_2 \times \epsilon_{32-bit}$$

- Operations are double-precision (64 bits)

$$\left|\tilde{E} - E\right| \leq U_1 \times \epsilon_{64-bit} + U_2 \times \epsilon_{64-bit}$$

RESULTS FOR JETENGINE

jetEngine, $x_1 \in [-5, 5], x_2 \in [-20, 5]$, Double Precision



SUMMARY

- New method for rigorous floating-point round-off error estimation
- Our method is embodied in new tool FPTaylor
- FPTaylor performs well and returns tighter bounds than previous approaches

SYNTHESIS

http://github.com/soarlab/FPTuner

MIXED-PRECISION TUNING

Goal:
Given a real-valued expression and output error bound, automatically synthesize precision allocation for operations and variables

APPROACH

- Replace machine epsilons with symbolic variables

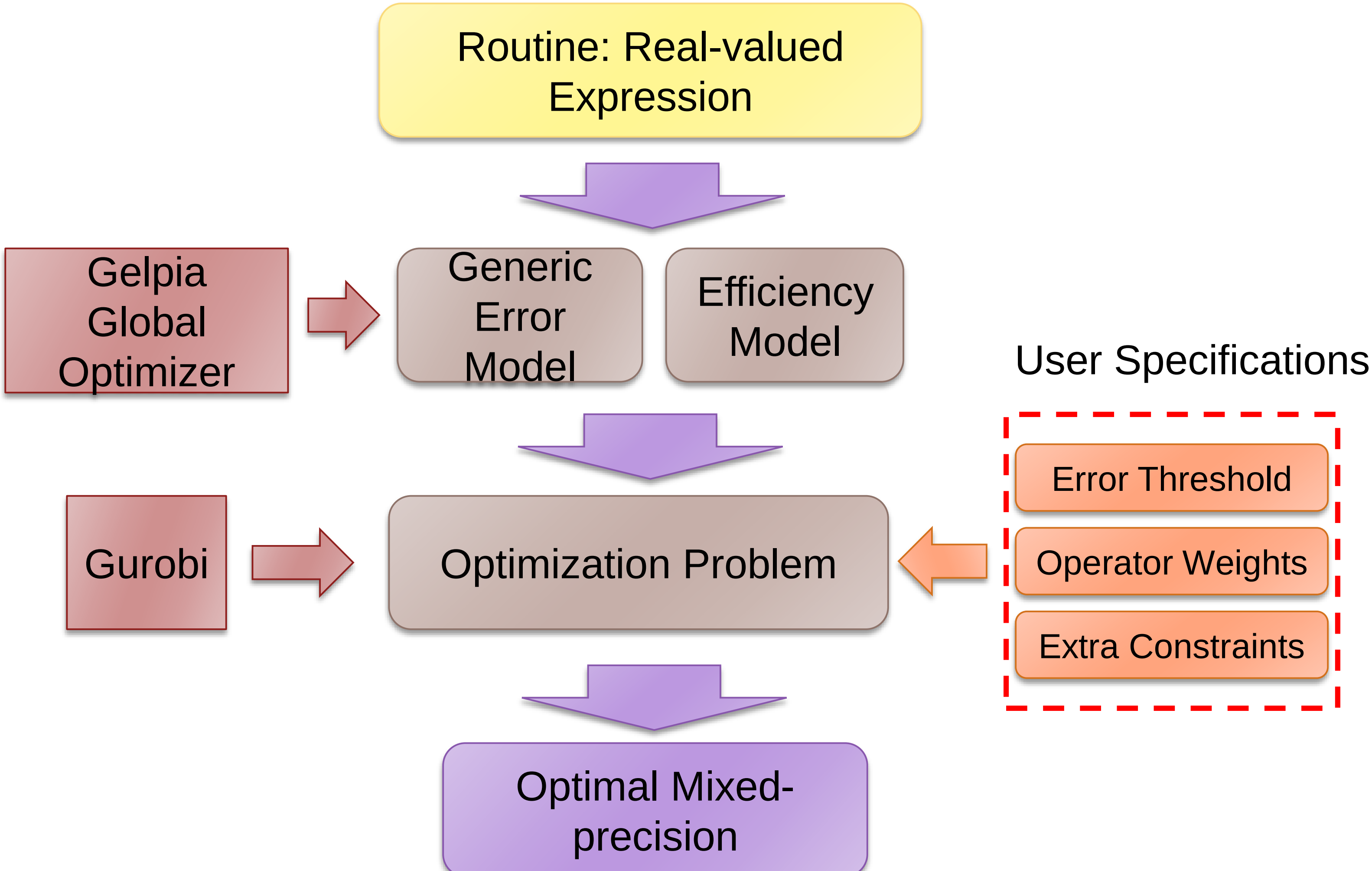
$$s_0, s_1 \in \{\epsilon_{32-bit}, \epsilon_{64-bit}\}$$

$$\left|\tilde{E} - E\right| \leq U_1 \times s_1 + U_2 \times s_2$$

- Compute precision allocation that satisfies given error bound
  - Take care of type casts

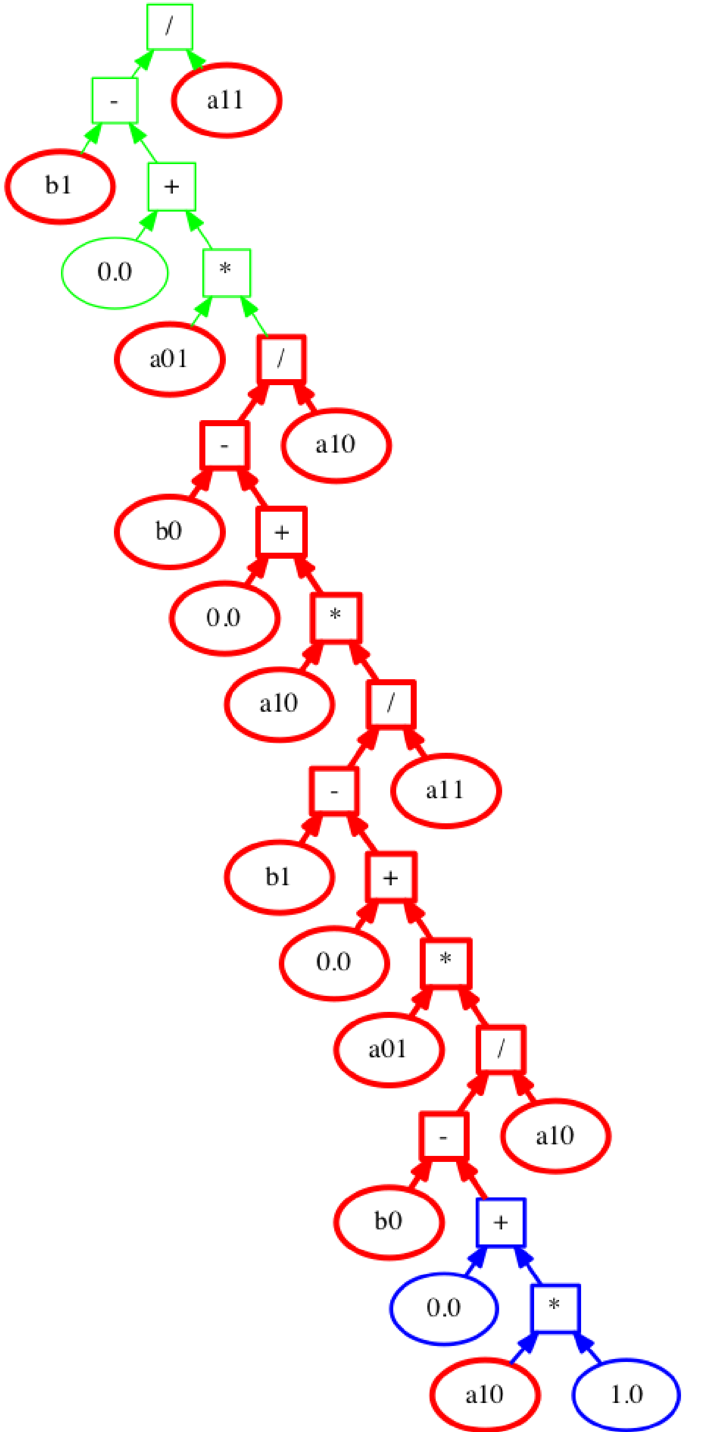
- Implemented in FPTuner tool

FPTuner TOOLFLOW



EXAMPLE: JACOBI METHOD

- Inputs:
  - 2x2 matrix
  - Vector of size 2
- Error bound: 1e-14
- Available precisions: single, double, quad
- FPTuner automatically allocates precisions for all variables and operations



SUMMARY

- Support mixed-precision allocation
- Based on rigorous formal reasoning
- Encoded as an optimization problem
- Extensive empirical evaluation
  - Includes real-world energy measurements showing benefits of precision tuning

SOLVING

http://github.com/soarlab/OL1V3R

MOTIVATION

- Poor scalability of floating-point solvers
  - Bit-blasting: formula → circuit

- Others showed that search-based solving can be effective for various SMT theories
  - Perform the search directly on theory level

- Can we achieve similar efficiency using stochastic local search on floating-points?
  - Inspired by Z3's qfbv-sls tactic for bit-vectors

STOCHASTIC LOCAL SEARCH

- Basic setting: local search + random choices

- Key ingredients
  - Score function
  - Neighborhood relation
  - Heuristics

SCORE FUNCTION

- score(expr, assignment) → rational

- Intuition: the ``degree'' of satisfiability
  - 1 = satisfiable
  - Example: s(x>2, x←1.99) > s(x>2, x←0)

- Key idea: measure a distance between *signed ordinal indices* of two floats
  - Total order on floats
  - Neighboring floats have a distance of 1

NEIGHBORHOOD RELATION

- Define neighbors of an assignment in a search step

- Several allowed mutations
  - Bit-flipping
  - ±ulp
  - (*2), (/2) – changing exponent

HEURISTICS

- Remove equality constraints when possible
  - (assert (and (= x (+ y z)) (> x 2.0)))

    → (assert (> (+ y z) 2.0))
- Use models derived from real arithmetic as initial assignments
  - (assert (> (+ y z) 2.0)) → y = 1, z = 3/2
- Variable neighborhood search
  - Refine the neighborhood relation into 3 subgroups and switch them on the fly

EVALUATION

- Compare OL1V3R with 5 state-of-the-art floating-point solvers

Tool	Version	Technique
MathSAT	5.5.4	Hybrid
CVC4	1.7	Bit-blasting
Z3	4.8.4	Bit-blasting
JFS	commit 2322167	Coverage-guided fuzzing
COLIBRI	revision 2176	Constraint propagation

RESULTS

Tool	Sat	Unsat	Unknown	Timeout	$Diff^B$	$Diff^H$
$OL1V3R^B$	115	0	2	80	-	0/16
$OL1V3R^H$	131	0	2	64	16/0	-
MathSAT	125	1	7	64	13/5	2/9
CVC4	117	1	10	69	10/9	2/15
Z3	88	0	10	99	3/32	0/43
JFS	113	0	0	84	4/8	0/20
COLIBRI	118	32	4	43	14/13	3/18

SUMMARY

- Implemented a prototype for solving floating-point constraints using SLS
  - Define key ingredients (score function, neighbors)
  - Devise custom heuristics

- Compared our tool to state-of-the-art solvers and confirmed its effectiveness