1. Context and purpose of our work

- Tool for automatically detecting and remediying anomalies in scientific floating-point programs
  - large-scale scientific single/multi-threaded applications have been growing rapidly
  - anomalies may cause rare but critical bugs that are hard to nonexpert to find or fix [1]
  - detection and remedy either at C code level or at run-time

- What are the usual anomalies?
  - rounding error accumulations
  - conditional branches involving floating-point comparisons
  - convergence misbehavior
  - Fortran: constants converted in full double precision accuracy if written with d_n notation
  - under/overflows, resolution of ill-conditioned problems
  - return result may be completely wrong
  - cancellation, benign or catastrophic, ...

- General flowchart of the framework

2. Usual approaches for finding anomalies in floating-point programs?

- Some techniques for detecting these usual anomalies [1],[3]
  - altering rounding mode of floating-point arithmetic hardware
  - may not normally be usable to remedy the problems
  - extending precision of floating-point computation
  - may increase run time significantly (due to the use of software interface)
  - using interval arithmetic
  - may produce a certificate, but run time is the greatest

- General principle find a local minimal set of changes on a given C code, so that the computed result remains within a given threshold of a known and more accurate result (exact, higher precision, ...)
  - implementation like binary search

- Delta-debugging algorithm [5]

- General flowchart of the framework

3. Detection of anomalies using delta-debugging algorithm and code transformations

- General flowchart of the framework

4. Some examples

- Inaccurate computation of the arc length of a given function [1]
  \[ g(x) = x + \sum_{k \geq 2} x^k \cos(x^k) \quad \text{over} \quad (0, \pi). \]
  - summing for \( x \in (0, \pi) \) divided into \( n \) subintervals
    \[ \sqrt{h} + (g(x_k + h) - g(x_k))/h, \quad \text{with} \quad h = \pi/n \text{and} \quad x_k = kh. \]
  - only 1 change is necessary: found in \( n \) sec.

- Bug in dgeps subroutine of LAPACK

  - I have the following problem with dgeps. For version 3.1.1 and newer, I get a reasonable result, for version 3.2.1 and later I get info=3.

  - the only difference between LAPACK 3.1.1 and 3.2.x: some call to dlarfp replaced by dlarfp
    - which call(s) to dlarfp made the program fail?

5. Conclusion and future work

- Current work on the automatic debugging of scientific floating-point applications
  - 1. CIL for applying transformations on a given C code,
  - 2. delta-debugging algorithm for finding a minimal set of effective changes to be applied on a given C code to improve its accuracy.

- Future work
  - implementation of other transformations (eg FloatToFF: float to float-floating)
  - application of these automated techniques to bug reports of widely used library (eg LAPACK), and automation of techniques that are originally done by hand
  - detection of some infinite loops, exception handling, ...
  - automatic and careful addition of an adjustable "fuzz" (small numerical value) on one side of the comparison that goes astray due to the subtleties of floating-point arithmetic
  - automatic user’s program scanning and modification when a constant is not converted to full expected precision because of difficulties of the programming language

Some references