MCALIB - An Automated Tool to Quantify Floating Point Rounding Error Sensitivity

Philip Leong
Computer Engineering Laboratory
School of Electrical and Information Engineering
The University of Sydney
(work by Michael Frechtling)
Overview

- Introduction
- Theory
- Implementation
- Results
- Conclusion
Overview

› Introduction
› Theory
› Implementation
› Results
› Conclusion
Focus on architectures, tools and applications for embedded systems

- Develop novel computer architectures and computing techniques.
- Improve designer productivity
- Cloud, cluster, multicore, microprocessor, FPGA, GPU

Applications

- Computational Finance
- Signal Processing
- Biomedical Engineering
› Machine learning
  - Applications to computational finance (hedging of FX risk with Westpac)
  - Hardware acceleration (online, non-periodic sampling)

› Biomedical engineering
  - Movement disorders (real-time, mobile analysis of motion, EMG and electrophysiology)

› Rounding error analysis
Collaboration with Movement Disorders Unit at Westmead Hospital and Graham Brooker (ACFR)

Causality of freezing in PD
- Combine EMG/accelerometer/pressure pad/video
- Understand sequence of events leading up to freezing
Dynamic error analysis methods effective at detecting rounding error

Implementation limited

- Often requires significant modification to existing source code
- Non-scalable
- Significant expertise required for implementation

Implementation of automated solution

- Monte Carlo arithmetic (D.S. Parker UCLA) for runtime validation of sensitivity to FP rounding errors
- Changes to software and storage are not required
Overview

› Introduction
› Theory
› Implementation
› Results
› Conclusion
IEEE FP not associative

- Using 8 significant digits

- \((11111113+\text{-}11111111)+7.5111111=9.5111111\)
- \(11111113+(-11111111+7.5111111)=10.000000\)
IEEE rounding errors are biased

\[ rp(x) = \frac{622 - x \cdot (751 - x \cdot (324 - x \cdot (59 - 4 \cdot x)))}{112 - x \cdot (151 - x \cdot (72 - x \cdot (14 - x))))} \]

Plot \( rp(x) - rp(u) \) for \( x = (u + \epsilon), \ldots, (u + 300\epsilon) \), \( u = 1.60632, \epsilon = 2^{-22} \)

- single precision IEEE
(a-b) is inaccurate when a ≈ b
- The major source of loss of significance

e.g. Quadratic formula to solve \(7169x^2 - 8686x + 2631 = 0\)

Intel processor, OSX, gcc 4.2.1 (ANSI C),

IEEE double: \(6.06243866321686e-01\) \(6.05361657461268e-01\)

IEEE single: \(6.06197e-01\) \(6.05408e-01\)
Monte Carlo Arithmetic

› Define floating point operation $x \otimes y$ in terms of real operation $x \cdot y$ (where $\in \{+, -, \times, \div\}$)

› Randomization: if $t$ is the floating point fraction precision, $e$ is exponent of $x$ and $\xi = U(-0.5, 0.5)$

$$
\text{randomize}(x) = \begin{cases} 
  x & \text{if exact within } t \text{ digits} \\
  x + 2^{e+1-t} \xi & \text{otherwise}
\end{cases}
$$

› Random rounding:

$$
\text{random\_round}(x) = \text{round}(\text{randomize}(x))
$$

› Monte Carlo Arithmetic:

$$
x \otimes y = \text{round}(\text{randomize}(\text{randomize}(x) \cdot \text{randomize}(y)))
$$

› Results different each time program is run => Monte Carlo simulation
Standard error $\sigma/\sqrt{n}$ gives a measure of the error in the mean.

Notice convergence to the exact sum value 9.511111.
Zero expected rounding error

\[ rp(x) = \frac{622 - x \cdot (751 - x \cdot (324 - x \cdot (59 - 4 \cdot x)))}{112 - x \cdot (151 - x \cdot (72 - x \cdot (14 - x)))} \]

Plot \( rp(x) - rp(u) \) for \( x = (u + \epsilon), \ldots, (u + 300\epsilon) \), \( u = 1.60632 \), \( \epsilon = 2^{-22} \)

- single precision IEEE
- single precision MCA
- histogram for MCA

Figure: D.S. Parker UCLA
For large values most of the digits of the result will be different.
Overview

› Introduction
› Theory
› Implementation
› Results
› Conclusion
Translation of C FP operators to MCA operations
- Compiler to translate any C-based source code.
- MPFR library to facilitate MCA operations.
- Storage requirements of all FP variables remain unchanged.
- Variable precision MCA – arbitrary precision of MCA operations at any point during execution.
 › C Intermediate language (CIL) by Necula (UCB) used to translate C FP operations to calls to MCALIB library
   - Translations to C source code defined in set of OCaml modules
   - FP operations translated by first lowering source to single assignment statement form, then converting FP operations to calls to MCALIB library
   - E.g. the FP multiplication:

   - Translated to the following $a = b \times c$; library function:

     $$ a = _\text{floatmul}(b, c); $$
Overview

› Introduction
› Theory
› Implementation
› Results
› Conclusion
Previous work was limited in analysis
- Determining number of significant figures in results
- Qualitative analysis of mean, standard deviation

We define sensitivity to rounding error using two measurements
- Number of significant figures lost due to rounding, $K$
- Minimum precision to avoid an unexpected loss of significance, $t_{min}$
Orthogonal polynomials used in approximation theory

Focus on Chebyshev polynomials of the first kind:

\[ T_n(z) = \cos(n \cos^{-1}(z)) \]

May be expanded to:

\[ T_{20}(z) = \cos(20 \cos^{-1}(z)) \]
\[ = 52488z^{20} - 2621440z^{18} + 5570560z^{16} \]
\[ - 6553600z^{14} + 4659200z^{12} - 2050048z^{10} \]
\[ + 549120z^8 - 84480z^6 + 6600z^4 \]
\[ - 200z^2 + 1 \]
Example Results – Chebyshev Polynomial

› Expanded form automatically translated to use MCALIB

› Testing performed using virtual precision, (t), values between 1 and 53 using a step of 1

› N = 1000 executions performed for each t step

› For each t value, results are summarized by calculating relative standard deviation
Example Results – Chebyshev Polynomial

Chebyshev Polynomial – Analysis for $z = 1.0$

- MCA Data
- MCA Outliers
- Linear Model
- Ideal
- $t_{\text{min}} = 21$
- $K = 24.3 \pm 0.02$

Relative Std. Dev. vs. Virtual Precision ($t$)
Sensitivity to rounding error detected
- Worst case result occurs at $z = 1.0$
- Loss of significance for worst case input of 24.3 digits, minimum required precision of 21 bits
- Single precision FP is insufficient

Can determine precision required to obtain results normally expected from single precision FP ($p=24$)
- Use worst case result, $K = 24.3$
- Determine optimized precision:

$$\lceil p + K \rceil = 49$$
Example Results – Chebyshev Polynomial

Chebyshev – Results of Precision Analysis

<table>
<thead>
<tr>
<th>Type</th>
<th>t</th>
<th>μ</th>
<th>Θ</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single</td>
<td>24</td>
<td>0.9985</td>
<td>1.2119e+00</td>
</tr>
<tr>
<td>Optimized</td>
<td>49</td>
<td>1.0000</td>
<td>3.4492e-08</td>
</tr>
</tbody>
</table>

Comparison of Single and Optimized Precision Results for Chebyshev Polynomial (using $z = 1.0$)

TABLE 3
Tool can compare different algorithms for sensitivity to rounding.
Introduction

Theory

Implementation

Results

Conclusion
MCALIB gives quantitative measurements of sensitivity to rounding error
- Takes arbitrary C source and generates summary graph

Applications
- Detect catastrophic cancellation
- Determine if single precision FP operators are sufficient for a given application
- Determine minimum precision requirements
- Enable aggressive optimization of floating point wordlengths in hardware designs
Future Work

› Family of automated rounding error analysis tools
  - Floating to fixed point conversion
  - Range analysis
  - Mixed precision analysis
  - Interval Arithmetic

› MCA operator analysis
  - Formal definition of MCA arithmetic
  - Proof of correctness of implementation