Reconfigurable Computing

Measuring Sensitivity to Rounding Error

Philip Leong (梁恆惠)
Computer Engineering Laboratory
School of Electrical and Information Engineering
The University of Sydney

http://phwl.org/talks

THE UNIVERSITY OF SYDNEY

MCALIB

THE UNIVERSITY OF SYDNEY

Overview

- Introduction
- Theory
- Implementation
- Results
- Conclusion

Overview

- Introduction
- Theory
- Implementation
- Results
- Conclusion

THE UNIVERSITY OF SYDNEY

Rounding Error Analysis

- 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

Rounding Error Analysis

› Monte Carlo Programming:
  - C library implementing MCA supported by source to source compilation
  - Variable precision MCA supporting both single and double precision IEEE formats
  - Inspect the accuracy of floating point variables in existing programs
  - Impose new semantics on existing arithmetic primitives

Theory

THE UNIVERSITY OF SYDNEY

Associativity

- IEEE-754 operations are not associative

$$(a + b) + c \neq a + (b + c)$$

- Simple example (Knuth) using 8 significant digits:

$$(11111113 - 11111111) + 7.5111111 = 9.5111111$$
$$11111113 + (-11111111 + 7.5111111) = 10.0000000$$

Rounding Errors

- IEEE-754 rounding errors are biased:
- Simple example:

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

- Test $rp(x) - rp(u)$ using the following conditions:

$$u = 1.60631924$$

$$x = u, (u + \epsilon), \ldots, (u + 300\epsilon)$$

$$\epsilon = 2^{-24}$$

Rounding Errors – IEEE754

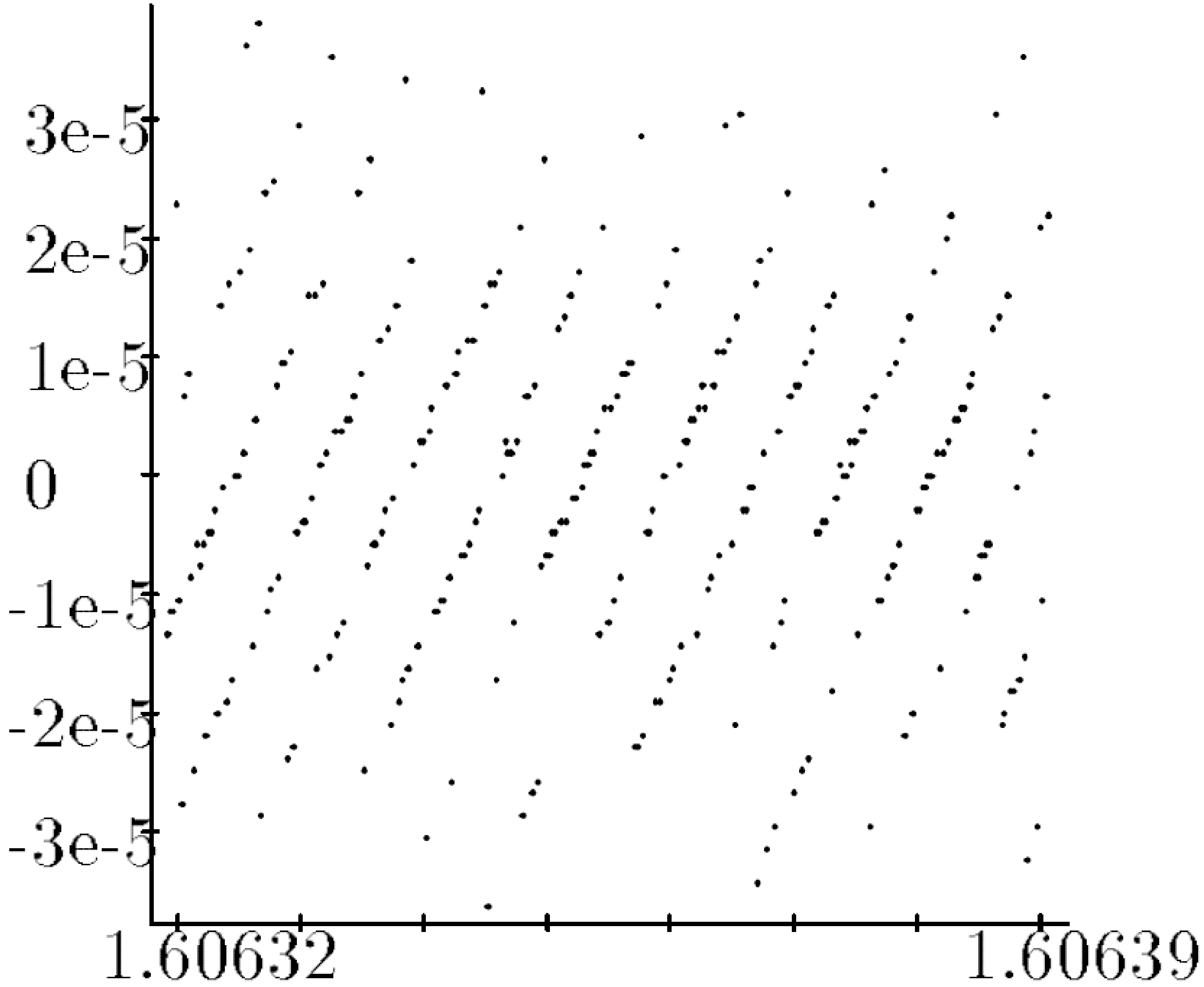


Figure: D.S. Parker UCLA

Catastrophic Cancellation

- Catastrophic cancellation is a major loss of significance in FP operations
  - Occurs when subtracting similar values
- Consider $\hat{x} = \hat{a} - \hat{b}$ where $\hat{a} = a(1+\delta_a)$ and $\hat{b} = b(1+\delta_b)$

$$\begin{aligned}
\left|\frac{x-\hat{x}}{x}\right| &\leq \left|\frac{(a-b)-(\hat{a}-\hat{b})}{a-b}\right| \\
&\leq \left|\frac{[a-a(1+\delta_a)]-[b-b(1+\delta_b)]}{a-b}\right| \\
&\leq \left|\frac{-a\delta_a+b\delta_b}{a-b}\right| \\
&\leq \max(|\delta_a|,|\delta_b|)\frac{|a|+|b|}{|a-b|}
\end{aligned}$$

- Relative error is highest when $|a-b| \ll |a|+|b|$

Monte Carlo Arithmetic

- MCA implemented using the inexact function:

$$\begin{aligned}\text{inexact}(x, t, \xi) &= x + 2^{e_x - t}\xi \\ &= (-1)^{s_x}(m_x + 2^{-t}\xi)2^{e_x}\end{aligned}$$

- Where:
  - $x \in \mathbb{R},\ x \neq 0$
  - t is a positive integer representing the virtual precision
  - $\xi \in U(-\frac{1}{2}, \frac{1}{2})$

Monte Carlo Arithmetic

- Define floating point operation $\circ \in \{+, -, \times, \div\}$ in terms of the inexact function:

$$x \circ y = \mathrm{round}(\mathrm{inexact}(\mathrm{inexact}(x) \circ \mathrm{inexact}(y)))$$

- Results are different each time the program is run -> multiple trials turns execution into a Monte Carlo Simulation.

- Results may be analyzed statistically

MCA Associativity

	(11111113. $\oplus$ −11111111.) $\oplus$ 7.5111111			11111113. $\oplus$ (−11111111. $\oplus$ 7.5111111)		
n	$\hat{\mu}$	$\pm$	$\hat{\sigma}/\sqrt{n}$	$\hat{\mu}$	$\pm$	$\hat{\sigma}/\sqrt{n}$
10	9.62506	±	0.11484	9.40092	±	0.27888
100	9.49476	±	0.04241	9.42260	±	0.06533
1000	9.51095	±	0.01295	9.49816	±	0.02042
10000	9.50977	±	0.00411	9.51206	±	0.00645
100000	9.51014	±	0.00129	9.51396	±	0.00204
1000000	9.51093	±	0.00041	9.51159	±	0.00065
10000000	9.51112	±	0.00013	9.51111	±	0.00020

Standard error $\sigma/\sqrt{n}$ gives a measure of the accuracy of mean.

Notice convergence to the exact sum value 9.5111111.

Figure: D.S. Parker UCLA

- Zero expected rounding error:

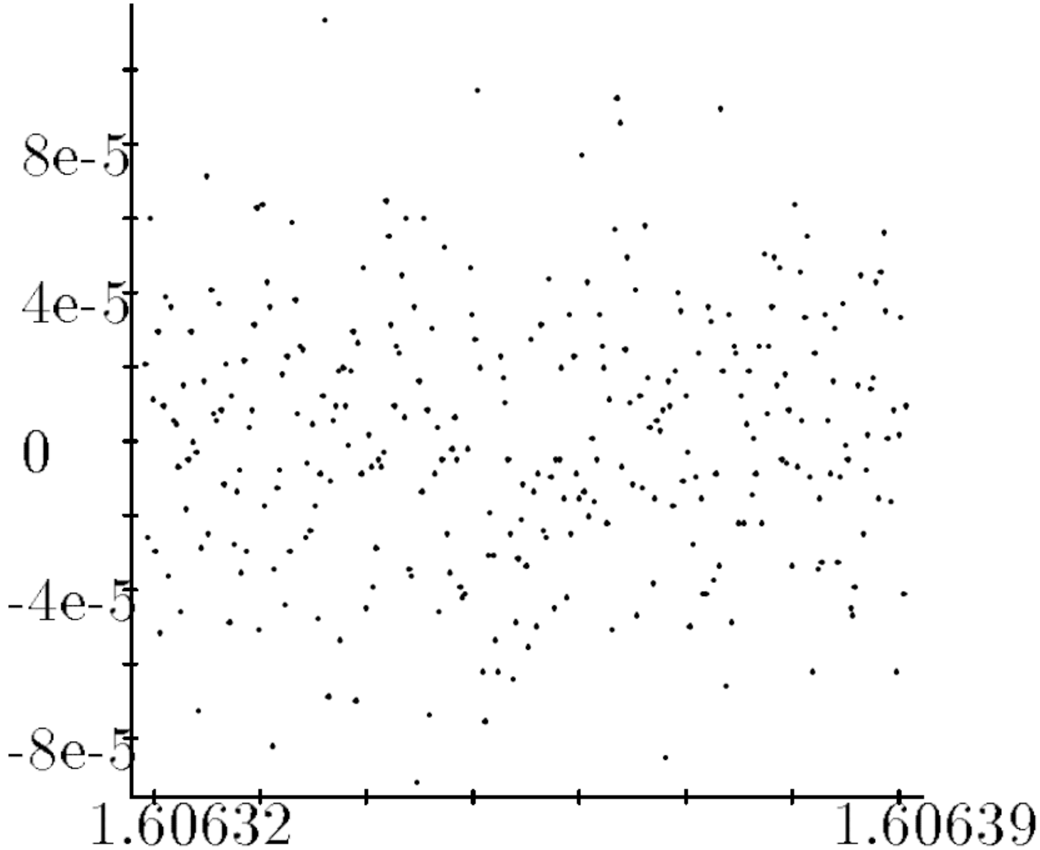


Figure: D.S. Parker UCLA

MCA Catastrophic Cancellation

x	$\boxed{+3.495683 \times 10^{0}}$
$x' = \mathsf{randomize}(x)$	$+3.49568320391695941600884\ldots$
y	$\boxed{+3.495681 \times 10^{0}}$
$y' = \mathsf{randomize}(y)$	$+3.49568191870795420835463\ldots$
$(x' - y')$	$+0.00000228520900520765421\ldots$
$\mathsf{round}(x' - y')$	$\boxed{+2.2852090 \times 10^{-6}}$

Catastrophic cancellation with input randomization.
Boxed values are 8-digit decimal floating-point values.

- For cancellation most digits of each result will be different which we can detect

Figure: D.S. Parker UCLA

Implementation

MCALIB Tool

- 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
  - Run time control of MCA implementation type – can select input (precision bounding) perturbation, output (random rounding) perturbation.

MCALIB – Source to Source Compilation

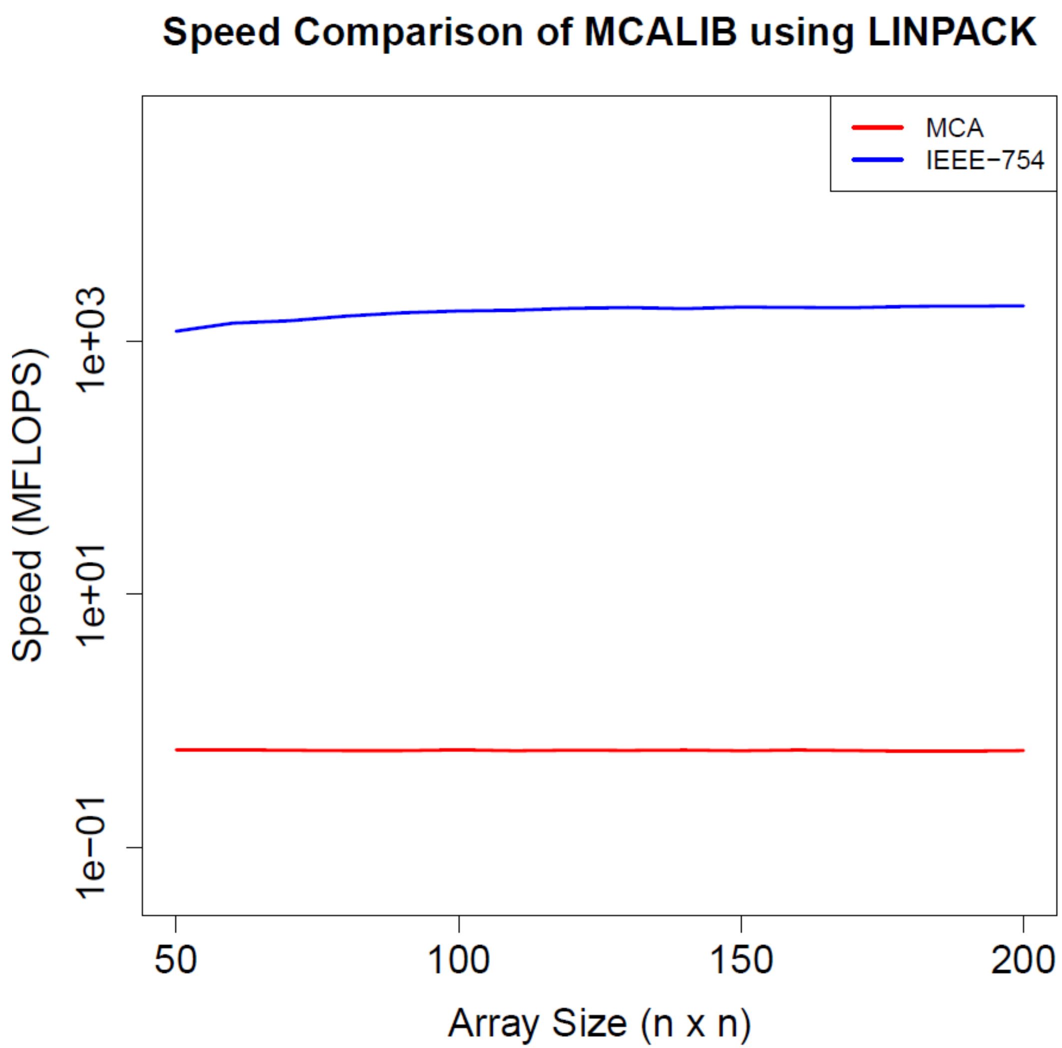
- C Intermediate language (CIL) by Necula (UCB) used to translate C FP operations to calls to MCALIB library
  - Translations to C source code defined in set of OCaml modules
  - FP operations translated by first lowering source to single assignment statement form, then converting FP operations to calls to MCALIB library
  - E.g. the FP multiplication:

```
a = b * c;
```

  - Translated to the following call to the MCALIB library function:

```
a = _floatmul(b, c);
```

MCALIB – Performance Decrease



Results

Expected number of significant digits

- For a p-digit binary floating point system, the log relative error is proportional to p
  - This is the ideal case

$$\delta \leq 2^{-p}$$
$$p \geq -\log_2(\delta)$$

- Sterbenz noted that the number of sigificant digits in result is linear with p
- Parker showed total significant digits in set of MCA results

$$s' = \log_2 \frac{\mu}{\sigma}$$

MCA Results

- 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

$$
\begin{aligned}
K &= t - s' \\
&= t - \log_2\left(\frac{\mu}{\sigma}\right) \\
&= \log_2(\Theta) + t
\end{aligned}
$$

Where $\Theta = \frac{\sigma}{\mu} \rightarrow \mu \neq 0$ is the **Relative Standard Deviation (RSD)**

  - Minimum precision to avoid an unexpected loss of significance, t_{min}

Example – Error Detection & Optimization

- Chebyshev polynomial - 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:

$$\begin{aligned} T_{20}(z) &= \cos(20 \cos^{-1}(z)) \\ &= 52488z^{20} - 2621440z^{18} + 5570560z^{16} \\ &- 6553600z^{14} + 4659200z^{12} - 2050048z^{10} \\ &+ 549120z^{8} - 84480z^{6} + 6600z^{4} \\ &- 200z^{2} + 1 \end{aligned}$$

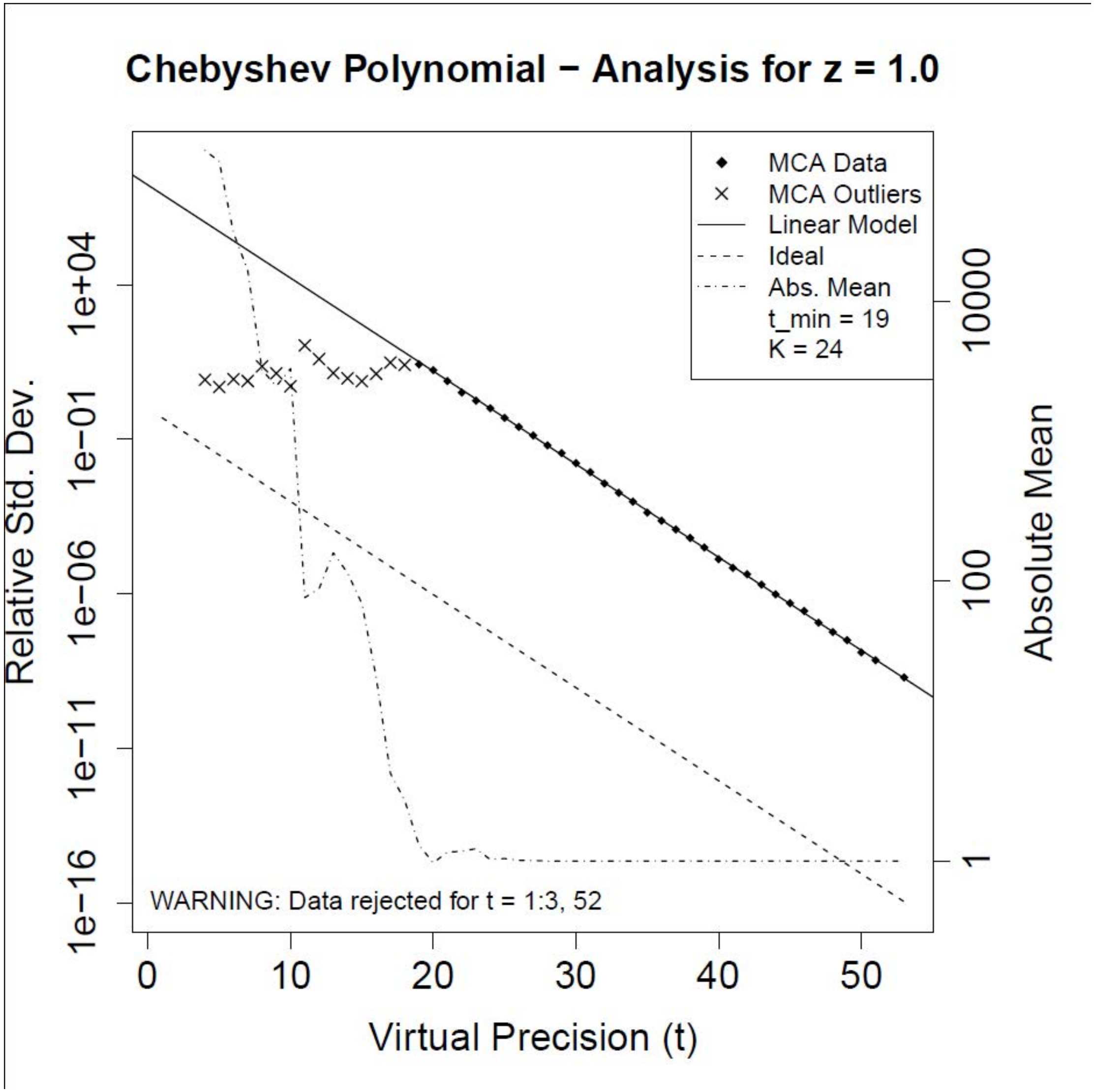
THE UNIVERSITY OF SYDNEY

Example – Error Detection & Optimization

- Expanded form automatically translated to use MCALIB
- Testing performed using virtual precision, (t), values between 1 and 53 using a step of 1
- N = 100 executions performed for each t step, (min. sample size).
- For each t value, results are summarized by calculating relative standard deviation
- Normality not assumed – Anderson-Darling test used to check normal distribution of results, (results grouped by t). Non-normal data sets removed from computation of K and t_{min}.
- Absolute mean plotted to ensure user is warned if mean approaches zero

$$\Theta = \frac{\sigma}{\mu} \to \mu \neq 0$$

Example – Error Detection & Optimization

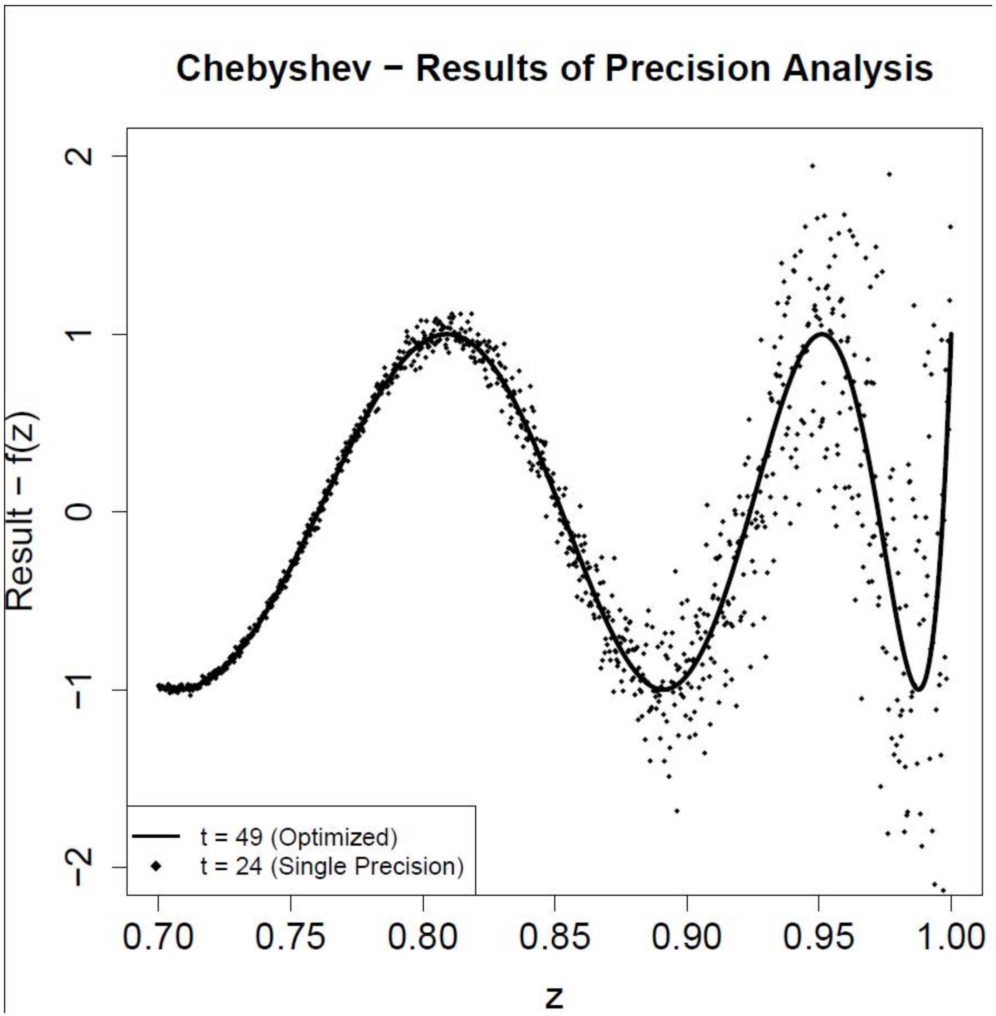


Example – Error Detection & Optimization

- Sensitivity to rounding error detected
  - Worst case result occurs at z = 1.0
  - Loss of significance for worst case input of 24.02 digits, minimum required precision of 19 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 = 24.02
  - Determine optimized precision:

$$\lceil p + K \rceil = 49$$

Example – Optimized Result



Chebyshev Polynomial			
Type	t	μ	Θ
Single	24	0.9985	1.2119e+00
Optimized	49	1.0000	3.4492e-08

TABLE 3
Comparison of Single and Optimized Precision Results for Chebyshev Polynomial (using $z = 1.0$)

Example – Error Detection & Algorithm Comparison

- Summation algorithm – widely used algorithm to sum a series of floating point values:

$$s = \sum_{i=1}^{n} x_i, \text{ for } n \geq 3$$

- Several algorithms available for implementation, including the Naïve, Pairwise and Kahan summation algorithms:

ALGORITHM 3: Pairwise Summation Algorithm

Input: Vector $X[1...n]$
Output: Sum s of vector X
n_{max} = 1;
if $n \leq n_{max}$ **then**
  s = $X[1]$;
  for $i = 2$ *to* n **do**
    s = s + $X[i]$;
  end
else
  m = floor(n / 2);
  s = pw($X[1...m]$) + pw($X[m+1...n]$);
end
return s

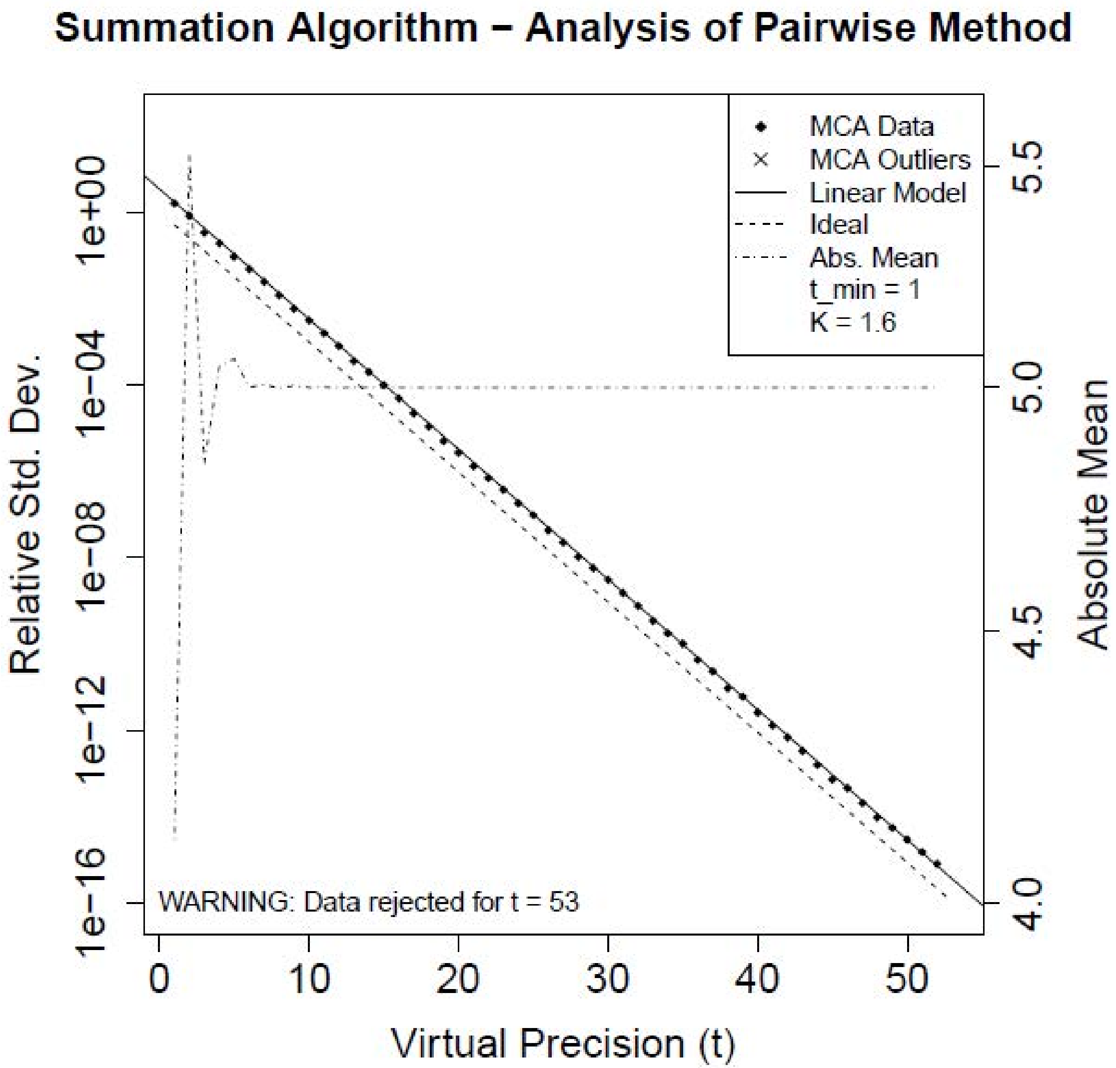
ALGORITHM 4: Kahan Summation Algorithm

Input: Vector $X[1...n]$
Output: Sum s of vector X
s = 0.0;
c = 0.0;
for i = *1 to* n **do**
  $y = X[i] - c$;
  $t = s + y$;
  $c = (t - s) - y$;
  $s = t$;
end
Return s

Example – Error Detection & Algorithm Comparison

- Can compare algorithm implementations using MCALIB
- Algorithm implementations automatically translated to use MCALIB
- Execute N = 100 trials for virtual precision values, (t), between 1 and 53
- Results analysis methods provide measure of sensitivity to rounding error for each algorithm
- Can perform quantitative comparison of algorithm implementations
- MCA plots provide fast visual comparison of algorithm implementations

Results for Individual Algorithm Implementation



Comparison of Algorithm Implementations

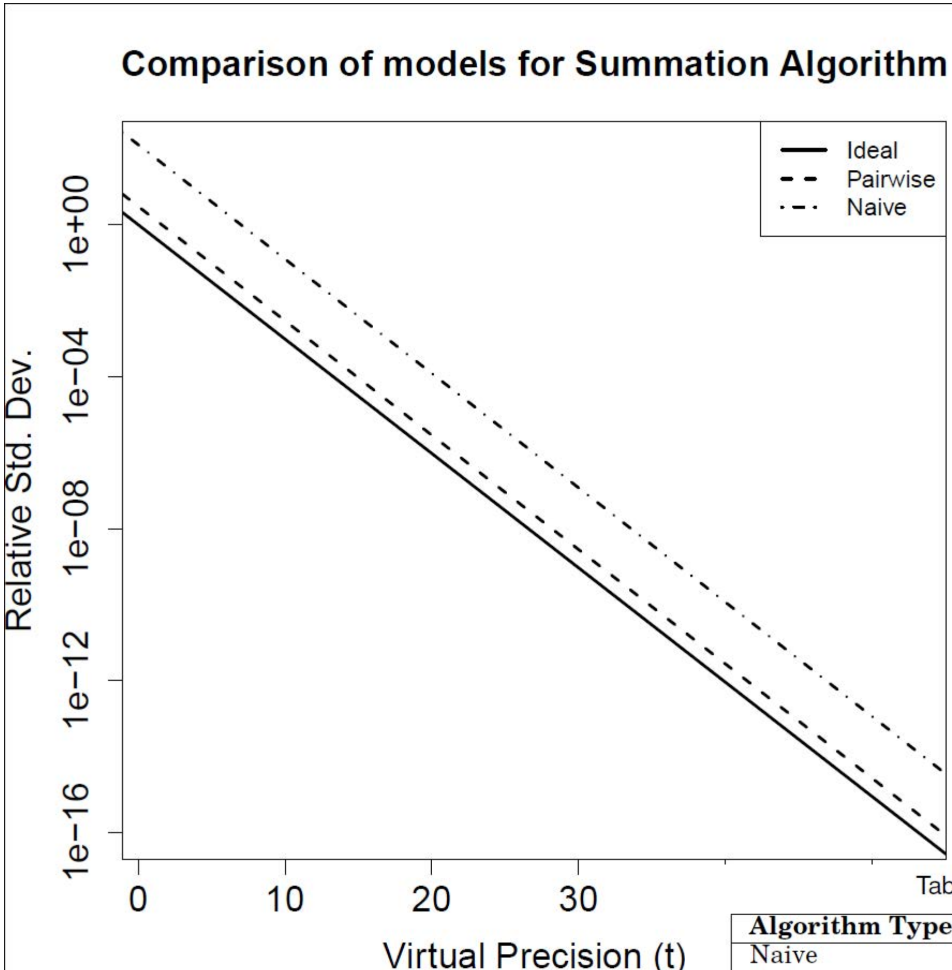


Table VI. Analysis of Summation Algorithms

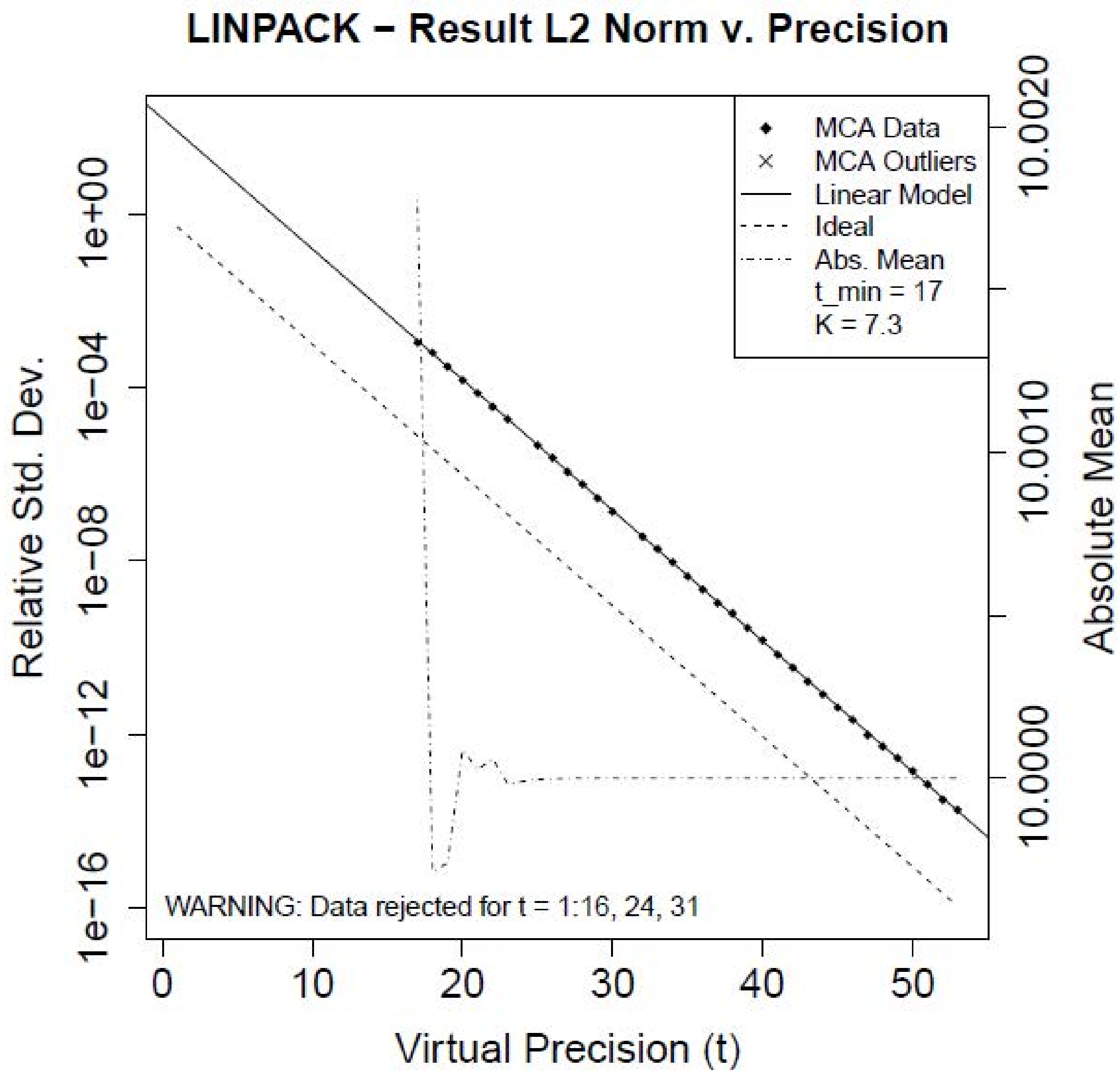
Algorithm Type	**Min. Req. Precision -** t_{min}	**Sig. Fig. Lost -** K
Naive	7	7
Kahan	7	7
Pairwise	1	1.6

Note: Summation Algorithm Results - Naive, Kahan & Pairwise

Comparison of Algorithm Implementations

- Comparison of more complex implementations (linear solvers):
  - LINPACK benchmark
  - LU Decomposition w. Back Substitution implementation from Numerical Recipes in C
- Results used to compare sensitivity to rounding error and Single vs. Double precision performance

Comparison of Algorithm Implementations



Comparison of Algorithm Implementations

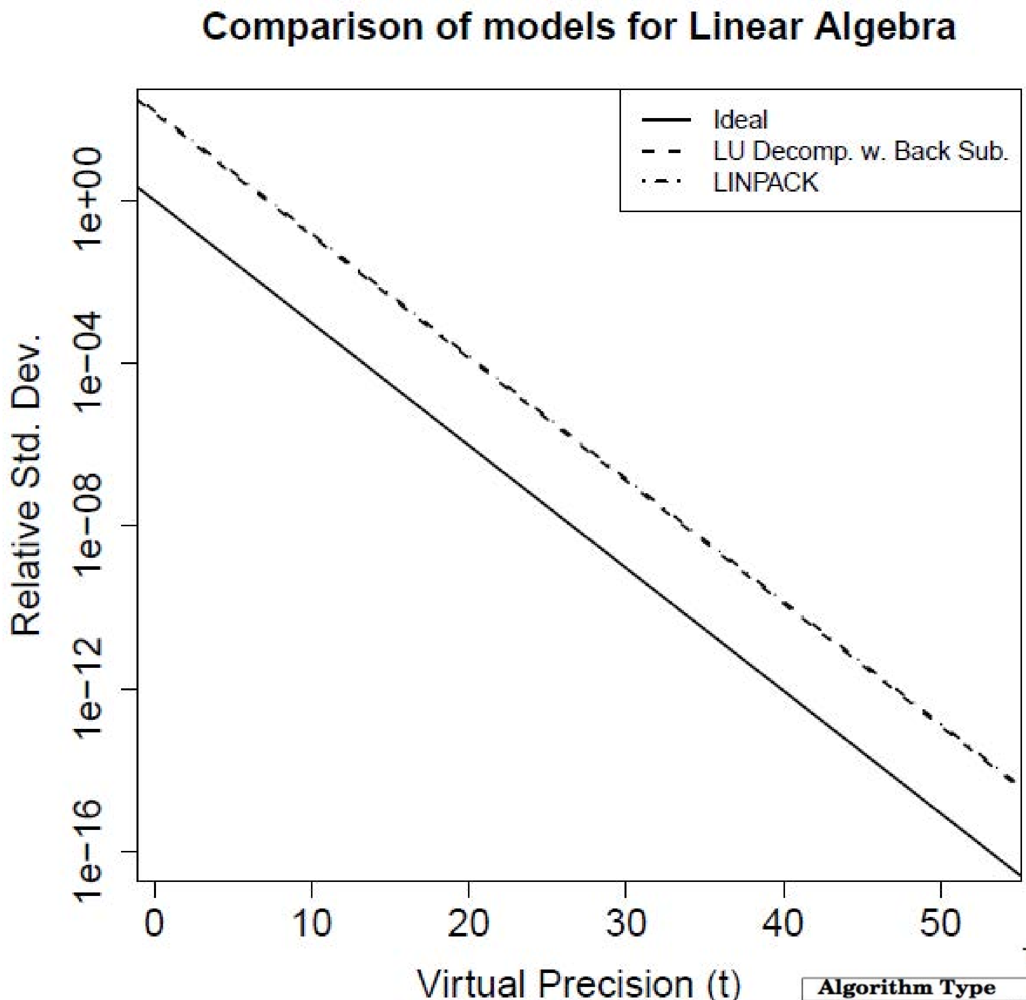


Table VII. Comparison of Linear Solvers

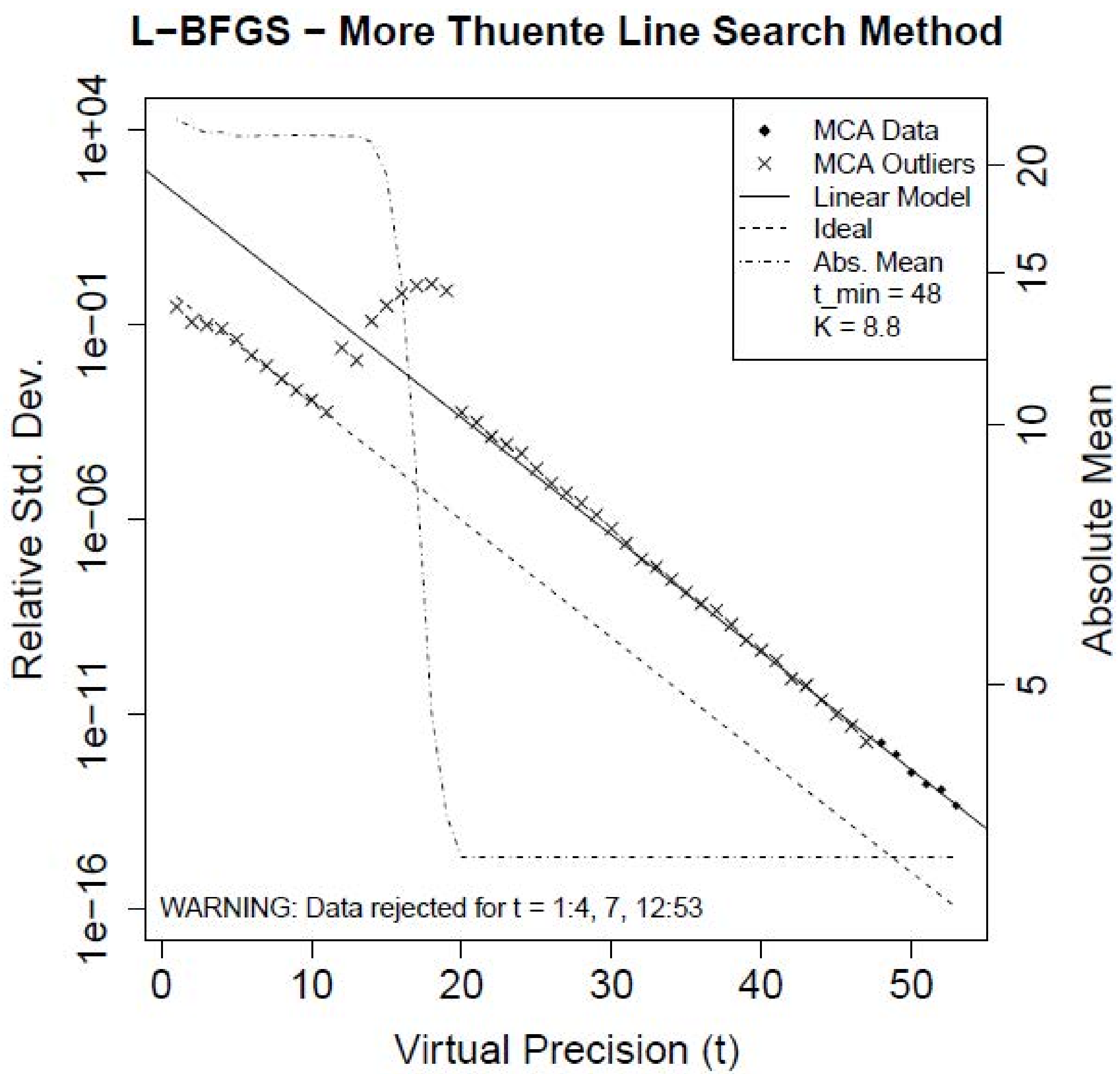
Algorithm Type	**Min. Req. Precision** - t_{min}	**Sig. Fig. Lost** - K
LU Decomp. w. Back Sub.	17	7.1
LINPACK	17	7.3

Note: Linear Solvers - Comparison of LINPACK and LU Decomposition with Back Substitution

Non-Normal Result Data

- All result data tested for normal distribution before results analysis is performed.
  - Data grouped by virtual precision (t) for testing
  - Anderson Darling test used
  - Non-normal data removed and not used in analysis
- L-BFGS Optimization – Iterative optimization algorithm
  - Precision analysis (MCALIB) tampers with convergence of results
  - Example of non-normal data
  - Anderson Darling test flags 47 out of 53 data sets as non-normal
  - Non-normal data sets have been included in example result to demonstrate the effect on analysis

Non-Normal Result Data

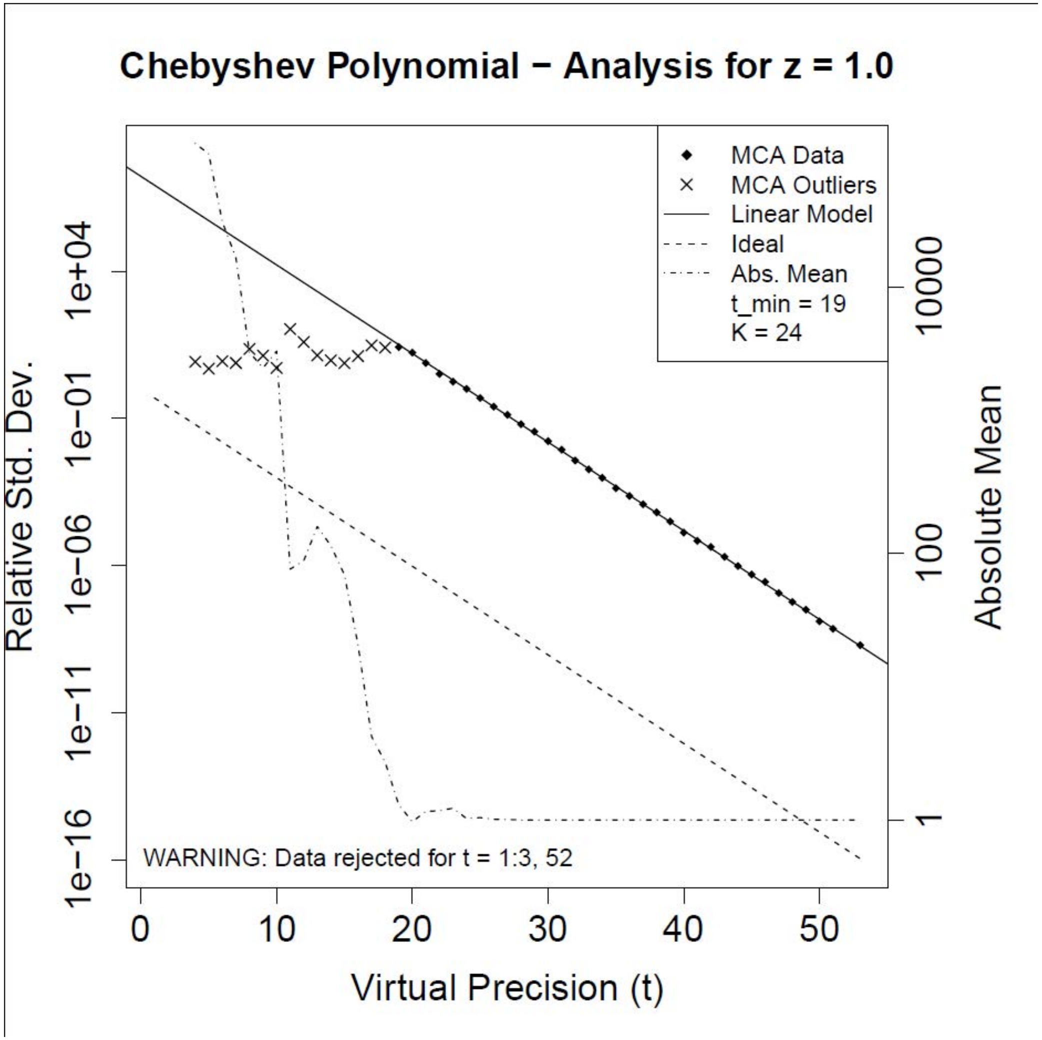


Overview

- Introduction
- Theory
- Implementation
- Results
- Conclusion

Conclusion

- MCALIB gives quantitative measurements of sensitivity to rounding error
  - Takes arbitrary C source and generates summary graph
- Applications in data analysis:
  - Dirty data
  - Missing data
  - Inexact Data
  - Sensitivity analysis



Future Work

- Family of automated rounding error analysis tools
  - Floating to fixed point conversion
  - Range analysis
  - Mixed precision analysis
  - Interval Arithmetic
- MCA operator analysis
  - Proof of correctness of implementation
- Speed improvements
  - Use quasi-Monte Carlo methods to increase the rate of convergence

References

- Michael Frechtling and Philip H. W. Leong. MCALIB - a tool for automated rounding error analysis. ACM Transactions on Programming Languages and Systems, 37:5:1–5:25, April 2015 (preprint available from http://www.ee.usyd.edu.au/people/philip.leong/publications.html)
- Code available from github (see paper)