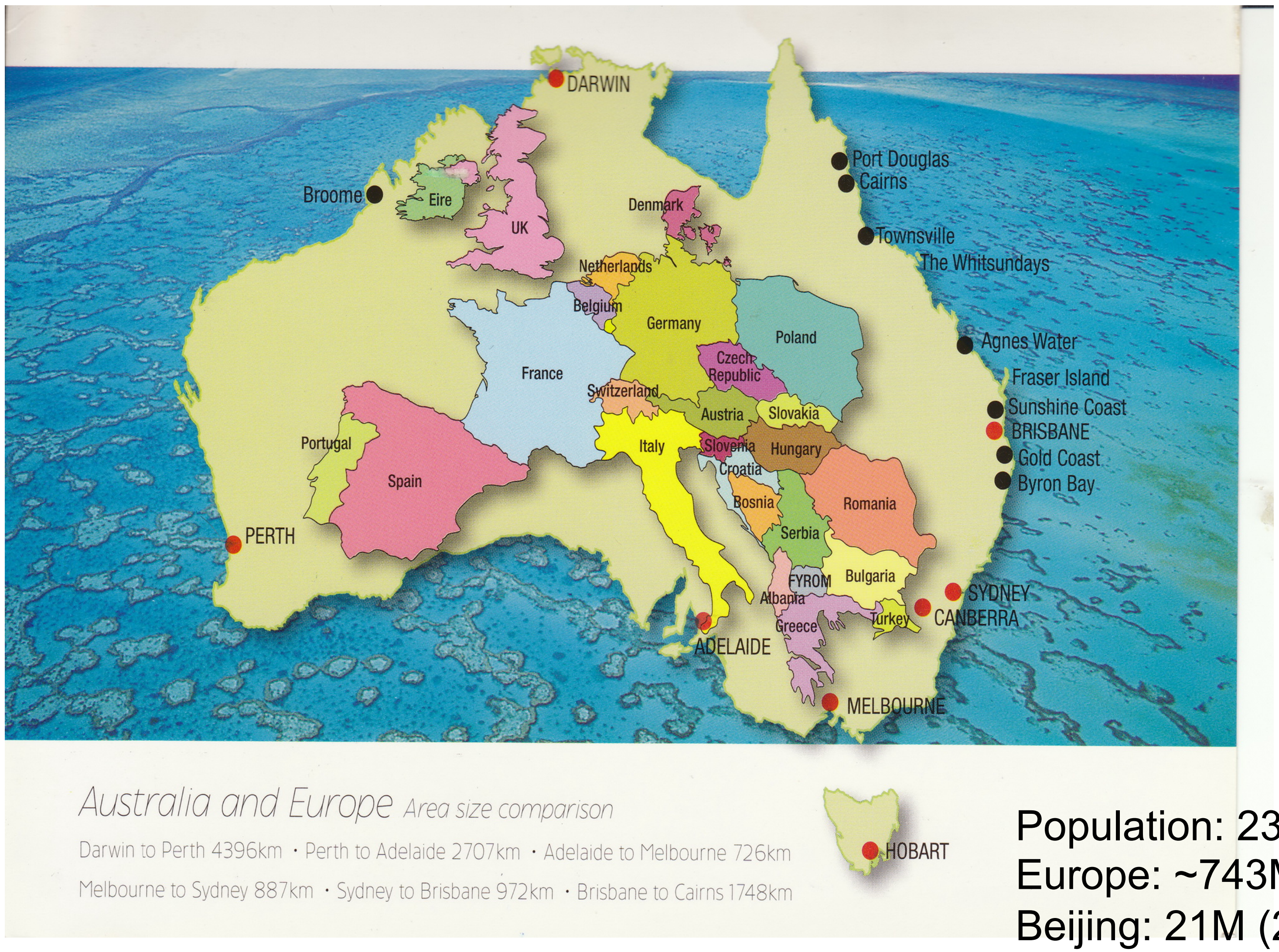
The Computer Engineering Lab

Philip Leong (梁恆惠)
Computer Engineering Laboratory
School of Electrical and Information Engineering
The University of Sydney
http://www.ee.usyd.edu.au/people/philip.leong/talks.html

THE UNIVERSITY OF SYDNEY

Australia



Australia and Europe *Area size comparison*

Darwin to Perth 4396km · Perth to Adelaide 2707km · Adelaide to Melbourne 726km
Melbourne to Sydney 887km · Sydney to Brisbane 972km · Brisbane to Cairns 1748km

Population: 23M (2013)
Europe: ~743M (2013)
Beijing: 21M (2013)

THE UNIVERSITY OF SYDNEY

Sydney

Population: 4.6M

THE UNIVERSITY OF SYDNEY

The University of Sydney

(Old part of campus – not our building)

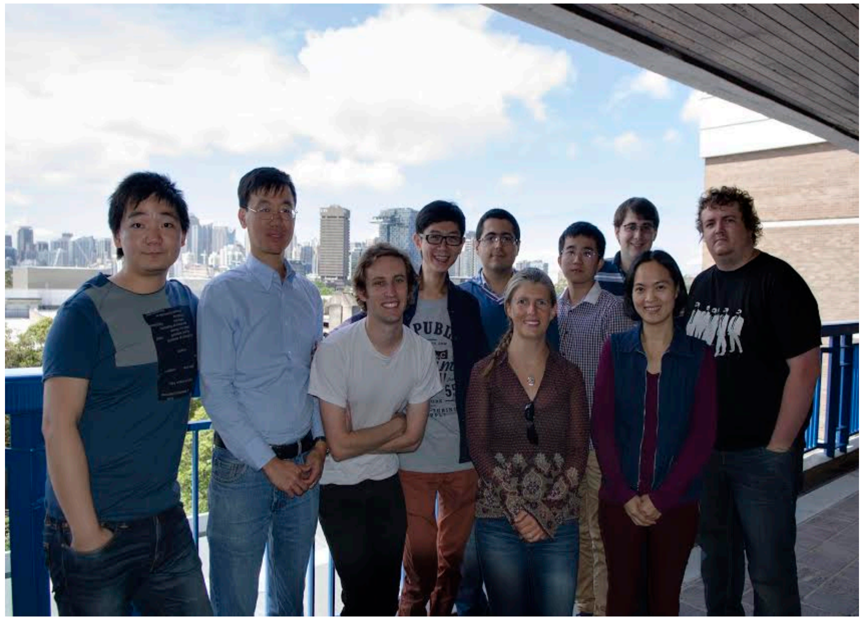
Population: 49,000
from 130 countries

THE UNIVERSITY OF SYDNEY

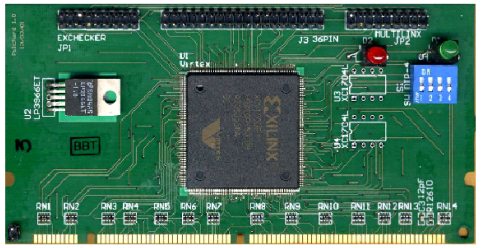
Computer Engineering Laboratory

› Work focuses on using parallelism to solve computationally demanding problems
  - Develop novel computer architectures and computing techniques.
  - Understand tradeoffs between ASIC, FPGA, GPU and microprocessor technologies
  - Improve designer productivity
  - 10 postgrads, 3 postdocs

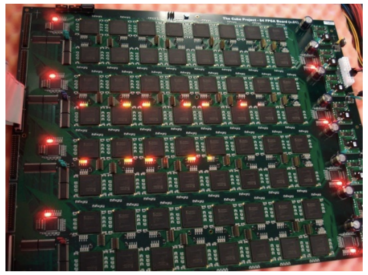
› Applications
  - Computational Finance
  - Signal Processing
  - Biomedical Engineering



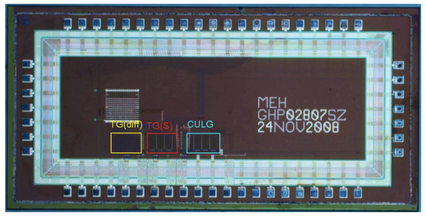
Some of our Past Work



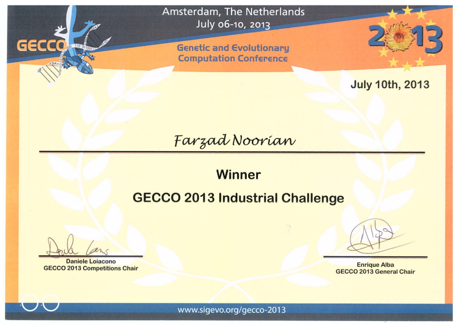
Pilchard DIMM FPGA board (2001)



Cube 64 FPGA board (2006)



Structured ASIC (2009)



Amsterdam, The Netherlands
July 06-10, 2013

GECCO

Genetic and Evolutionary Computation Conference

2013

July 10th, 2013

Farzad Noorian

Winner

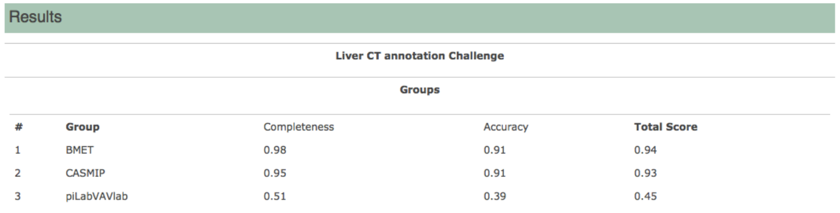
GECCO 2013 Industrial Challenge

[illegible]
Daniele Loiacono
GECCO 2013 Competitions Chair

[illegible]
Enrique Alba
GECCO 2013 General Chair

www.sigevo.org/gecco-2013

GECCO Industrial Challenge (2013)



Results

Liver CT annotation Challenge

Groups

#	Group	Completeness	Accuracy	Total Score
1	BMET	0.98	0.91	0.94
2	CASMIP	0.95	0.91	0.93
3	piLabVAVlab	0.51	0.39	0.45

ImageCLEF Liver CT Annotation (2014)

Machine Learning

THE UNIVERSITY OF SYDNEY

Hedging of Foreign Exchange Risk

› Three year ARC Linkage project started 2012 sponsored by Westpac

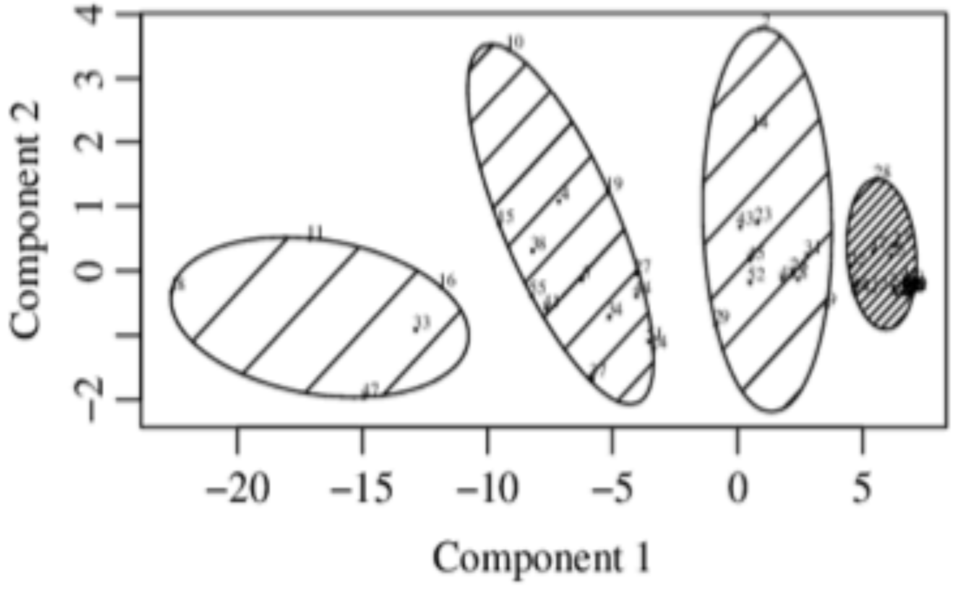
› Apply parallel computing and machine learning techniques to better understand and manage exposure to FX risk

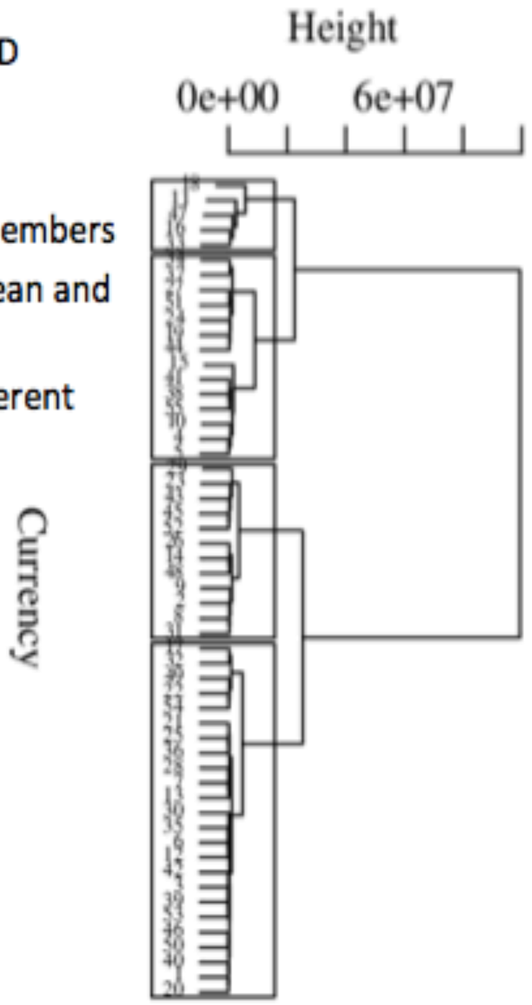
  - Interface to scalable cloud computing resources
  - Predict customer flow and exchange rates
  - Develop hedging strategies and market models

› Enable Australian banks to better quantify and manage risk, making them more competitive in global FX markets

Clustering of Irregular Time Series

- Tick history (25 Sept 2012) of 55 major currencies against USD
- On average 18000 recordings per currency at 1ms resolution
- Cosine basis representation with 200 elements
- # clusters= 4 via SVD and K-means assigns 5, 14, 12 and 24 members
- I: Western European, II: Central European, III: Eastern European and South American, IV: Middle Eastern and African
- Recursive K-means groups Middle Eastern and African in different clusters





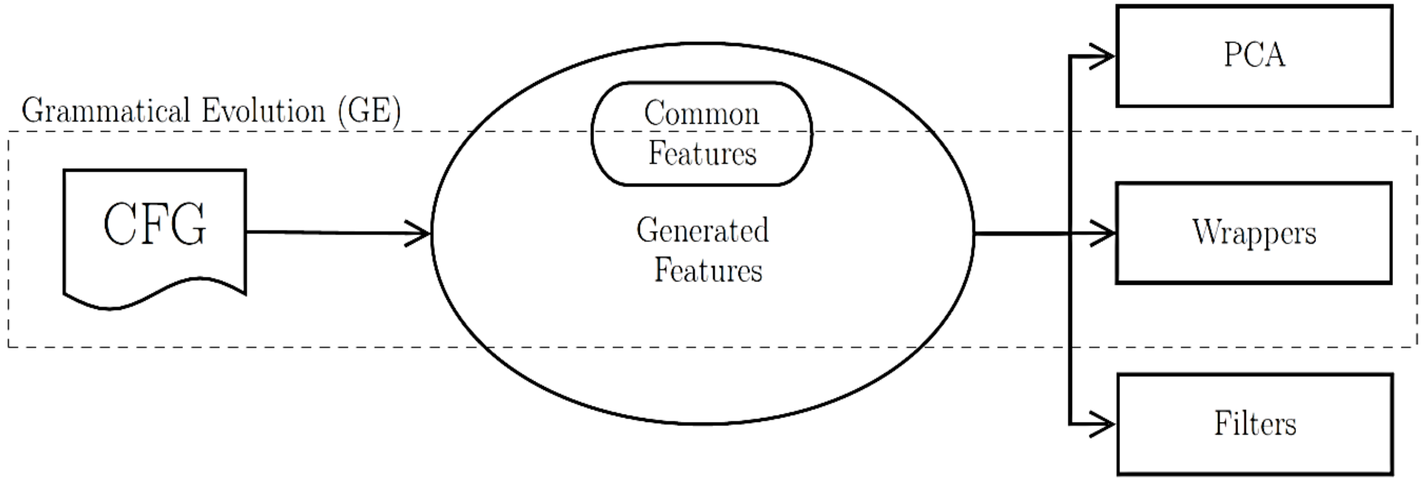
Grammar Based Feature Generation

› Technical indicators have been widely used as input features to ML algorithms.

  - Technical indicators are a type of “Feature extractors”.

› The number of technical indicators selected is not the same with some overlap between different works. However, the choice is generallv ad-hoc.

› Cal
  frar



Application of Feature Generation to Index

Feature Selection using mRMR + Integer GA

Method	FTSE	N225	NDX	HSI	SSMI
ARIMA	34.33	207.07	14.35	138.95	47.96
ETS	33.54	207.13	14.05	139.95	47.14
AR(1)	33.71	207.21	**13.98**	140.24	47.60
EMA	42.63	275.20	14.05	183.51	61.42
SVM (TIs)	40.27	203.99	15.15	136.70	46.26
SVM (Grammar)	**32.64**	**203.30**	14.37	**135.11**	**46.17**

Method	SSEC	TWII	AORD	GDAXI	GSPC
ARIMA	19.78	69.52	29.97	44.19	7.58
ETS	19.78	69.46	29.80	44.08	7.60
AR(1)	19.71	69.41	29.97	44.18	7.61
EMA	28.16	96.55	38.81	58.26	9.39
SVM (TIs)	18.03	**66.85**	29.23	**43.70**	7.54
SVM (Grammar)	**17.96**	67.13	**28.61**	44.49	**7.49**

Table: RMSE for test data for major stock indices using the ARIMA, ETS, AR(1), EMA($p = 5$) and SVM using technical indicators (TIs) and grammar features

THE UNIVERSITY OF SYDNEY

Hedging of FX Risk

› Using stochastic model predictive control (quadratic optimisation)

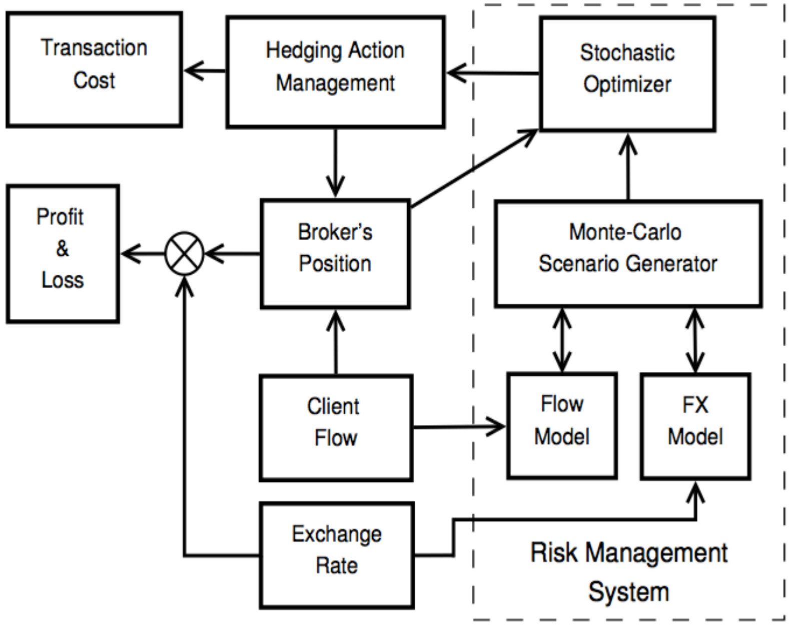


Fig. 1. The proposed FX risk management system.

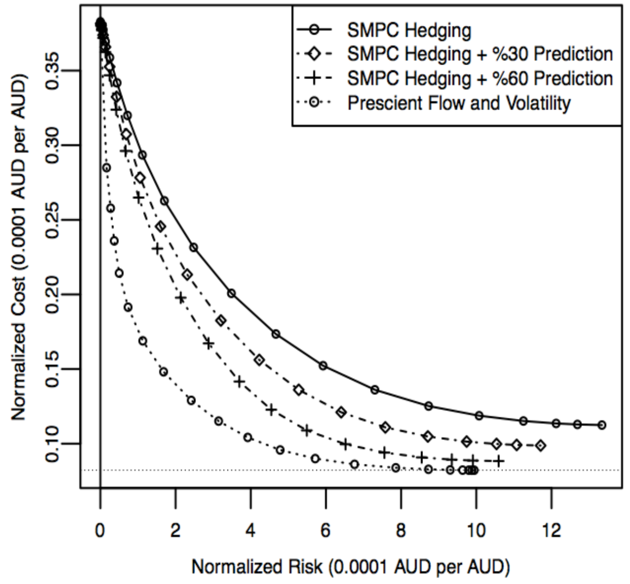
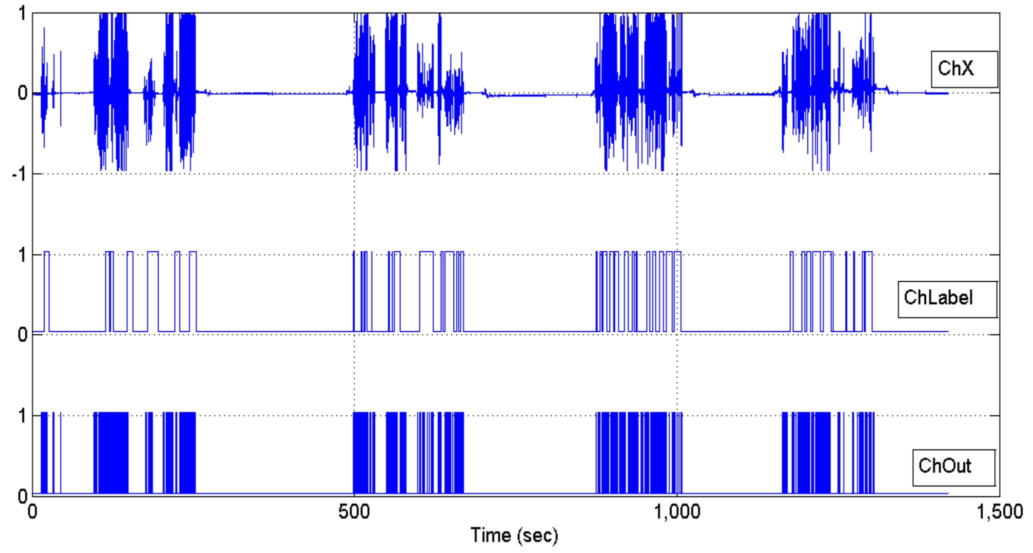


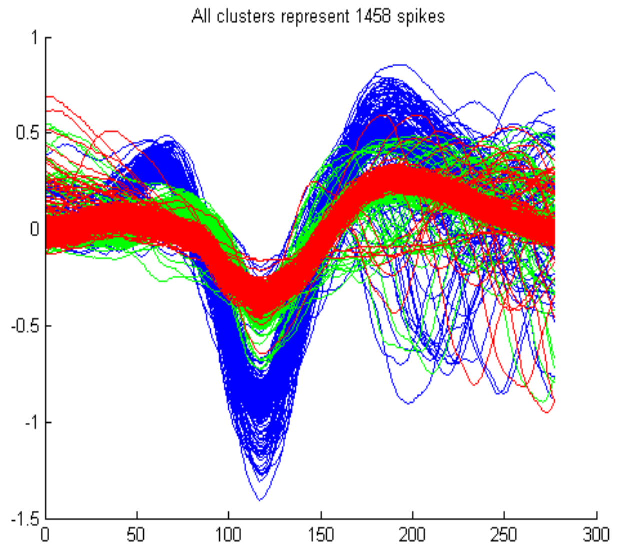
Fig. 10. Risk-cost frontiers for historical example, with and without prediction.

Classification in Biological Data

› Automatic subject-independent anomaly detection for freezing of gait detection in Parkinson's Disease

› Online sorting of muscle action potentials from a needle EMG recording into active motor units





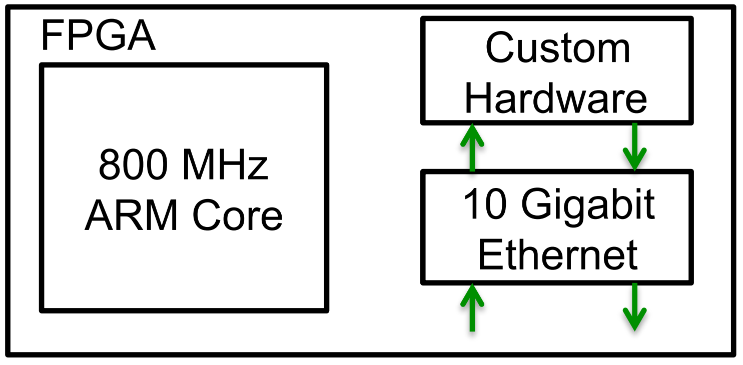
Hardware Design

THE UNIVERSITY OF SYDNEY

Online FPGA-based Anomaly Detection

- Three year ARC Linkage project announced 2013 sponsored by Zomojo
- Online hardware-assisted machine learning systems which reduce latency and energy consumption by 10-1000x
  - FPGAs which integrate network and decision logic
- Improved classifiers, regression and outlier detection algorithms with emphasis on latency with applications in network monitoring, high speed signal processing, and machine prognosis

Platform	Power (mW)	Latency (uS)	Energy (10^-5 J)
Our processor	26880	**28**	**75**
NIOS II	15120	58428	88344
DSP	**2025**	54926	111123
CPU (Intel)	36818	238	876

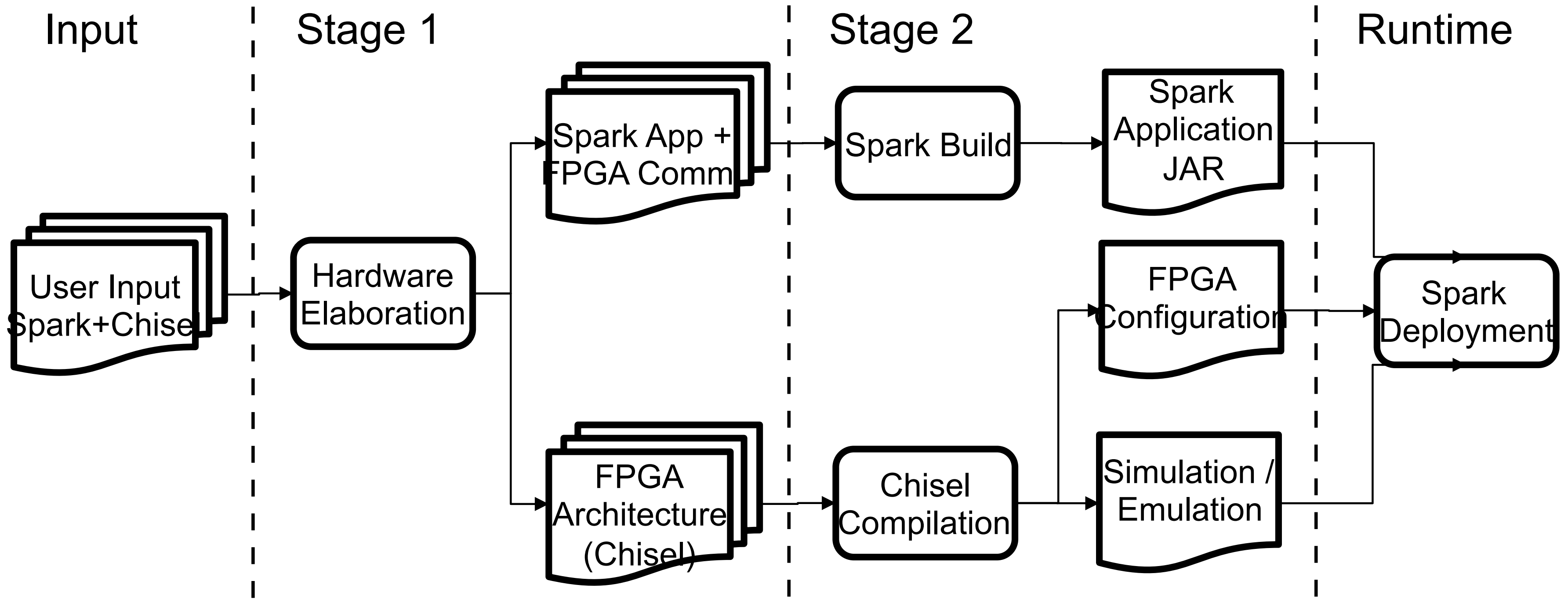


Signals from network

MapReduce FPGA Environment (MrFe)

› Combine Chisel and Apache Spark to create a heterogeneous MapReduce platform for programming FPGAs

- Functional programming, Fault tolerance, HDFS, existing MR libraries,

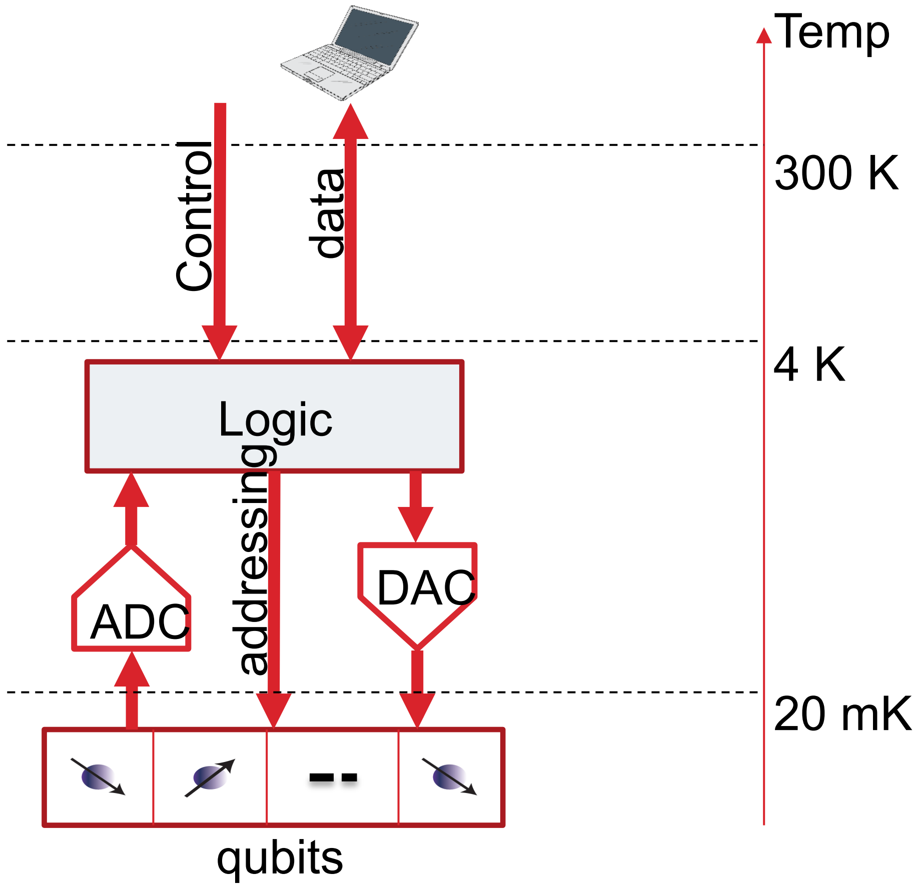


THE UNIVERSITY OF SYDNEY

Interfaces for Quantum Computing

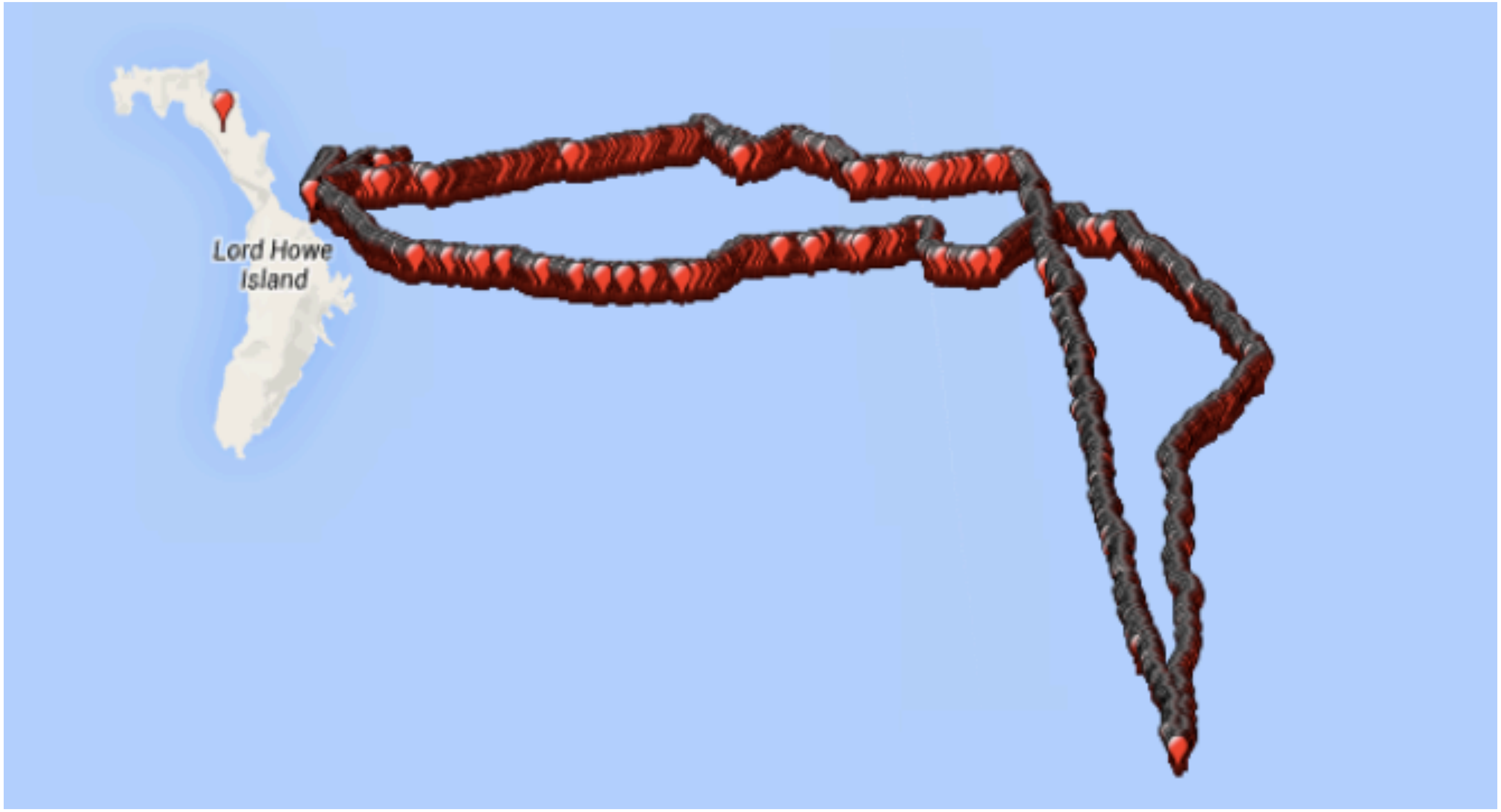
› Converting data/signals interfacing to quantum computing (qubits).

› Major challenge is to deal with CMOS integrated circuit design at very low temperature, 4K



GPS Tracking

Nobody had recorded entire flight path of masked booby (nutritional data)



Wildlife Tracking

- We developed first device capable of recording 20 hours of continuous video and used it to record masked boobies (alas, no GPS)
- Develop improved low-power video+GPS using microcontroller
- Understand nutrition of animals in wild

Masked Booby Diving

Cow Nutrition

MCALIB - Measuring Sensitivity to Rounding Error with Monte Carlo Programming

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

THE UNIVERSITY OF SYDNEY

Overview

Overview

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

Overview

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

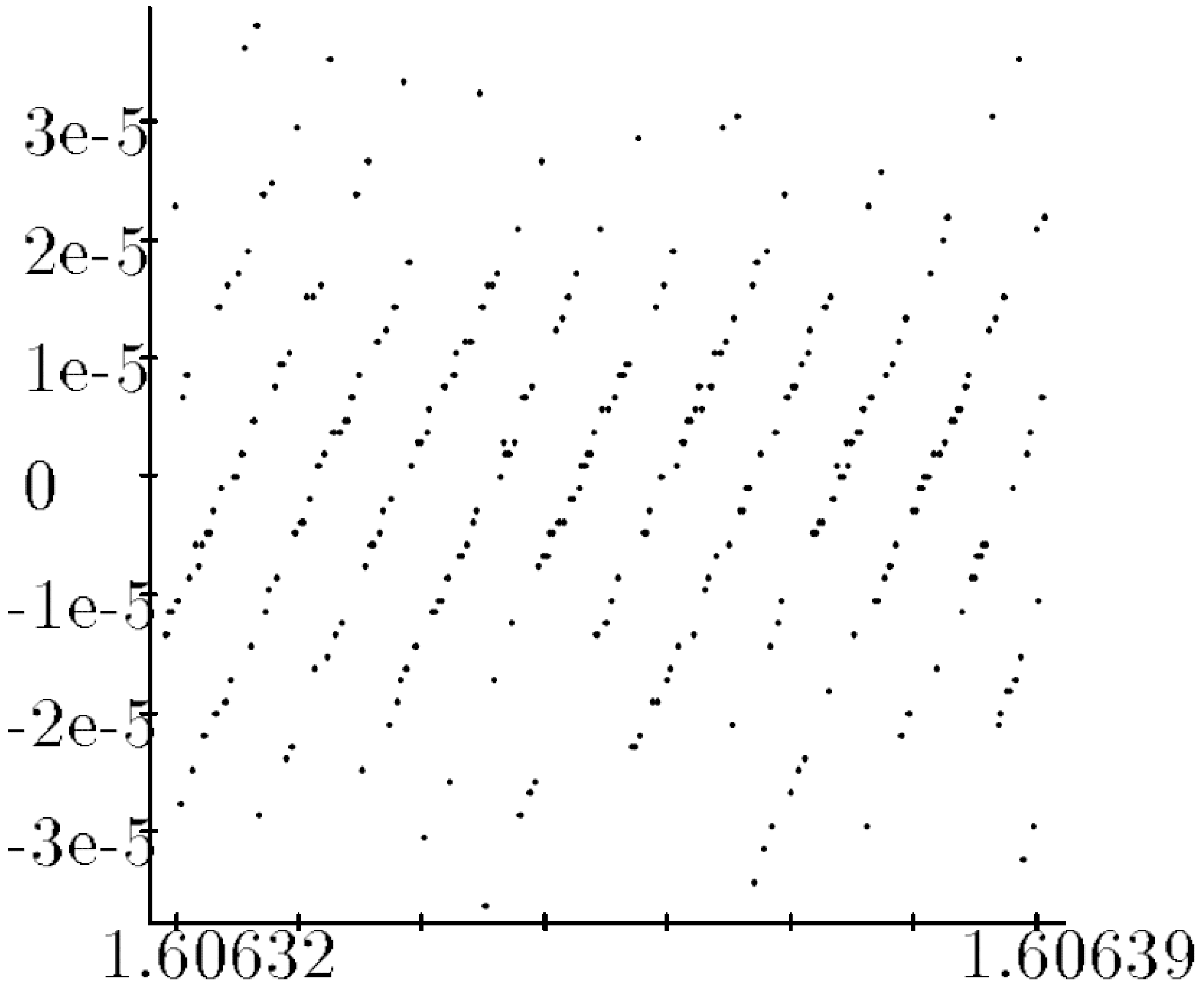


Figure: D.S. Parker UCLA

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
- $\xi \in U(-\frac{1}{2}, \frac{1}{2})$

THE UNIVERSITY OF SYDNEY

Monte Carlo Arithmetic

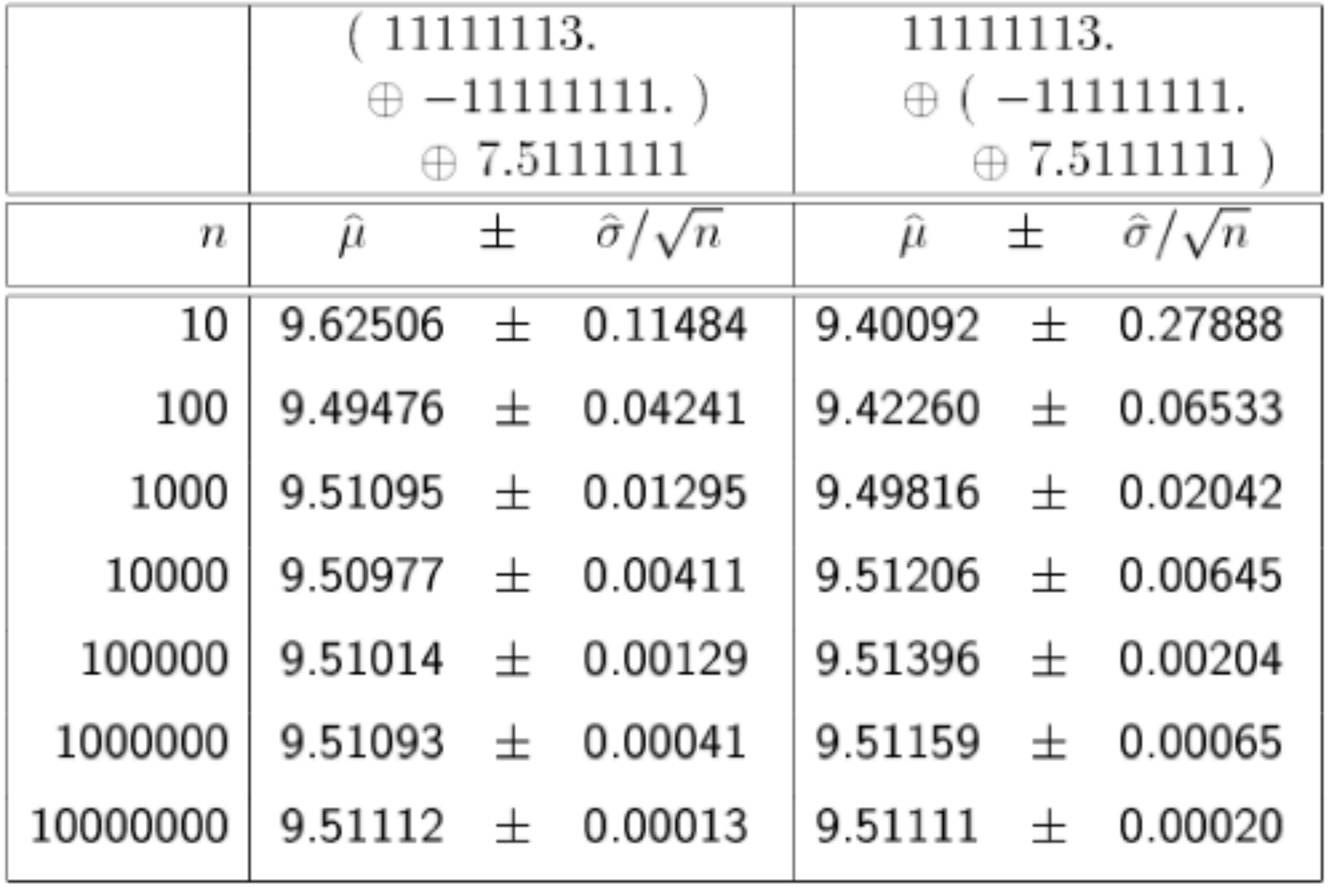
- Define floating point operation $\circ \in \{+, -, \times, \div\}$ in terms of the inexact function:

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

- 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	$\pm$	0.11484	9.40092	$\pm$	0.27888
100	9.49476	$\pm$	0.04241	9.42260	$\pm$	0.06533
1000	9.51095	$\pm$	0.01295	9.49816	$\pm$	0.02042
10000	9.50977	$\pm$	0.00411	9.51206	$\pm$	0.00645
100000	9.51014	$\pm$	0.00129	9.51396	$\pm$	0.00204
1000000	9.51093	$\pm$	0.00041	9.51159	$\pm$	0.00065
10000000	9.51112	$\pm$	0.00013	9.51111	$\pm$	0.00020

Standard error σ/√n gives a measure of the error in the mean.

Notice convergence to the exact sum value 9.5111111.

Figure: D.S. Parker UCLA

THE UNIVERSITY OF SYDNEY

MCA Rounding errors

› Zero expected rounding error:

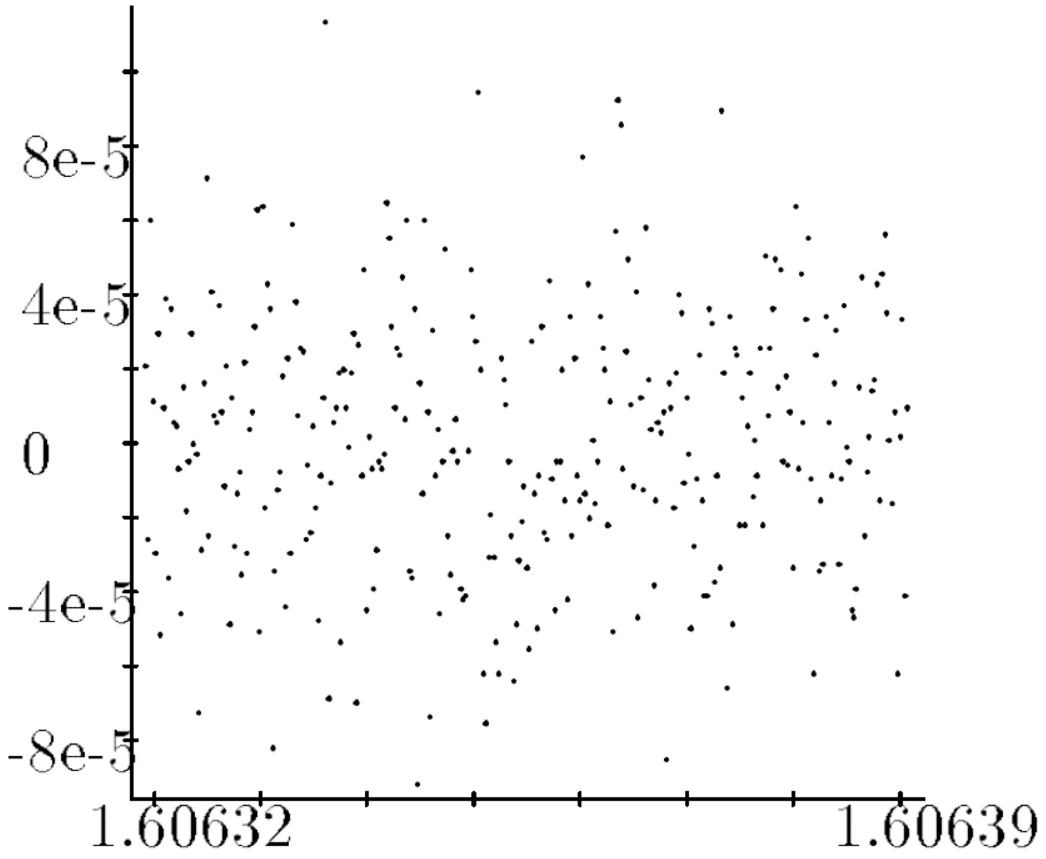
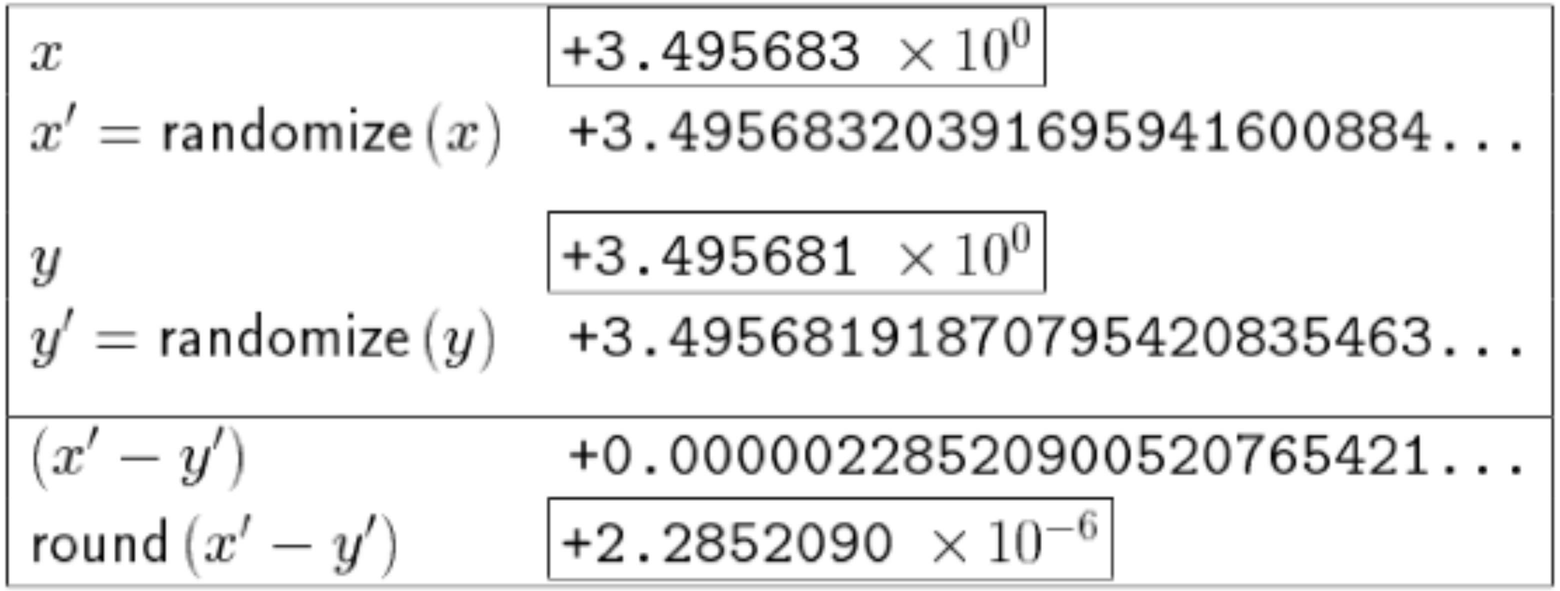


Figure: D.S. Parker UCLA

MCA Catastrophic Cancellation



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

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

- For large values most of the digits of the result will be different

Figure: D.S. Parker UCLA

Overview

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 – Source to Source Compilation

› MCALIB implementation of binary FP operation:
  - Extend input operators to working precision using MPFR
  - Apply inexact operation
  - Round results to original precision

```
ALGORITHM 1: MCA Binary Operation
Input: Precision $p$ FP operands $x_f$ and $y_f$
Output: Precision $p$ FP result $r_f$
$x$ = extend($x_f$, $p + t$);
$y$ = extend($y_f$, $p + t$);
$r$ = extend(0.0, $p + t$);
$x$ = inexact($x$);
$y$ = inexact($y$);
$r$ = mpfr_op($x$, $y$);
$r$ = inexact($r$);
$r_f$ = round($r$, $p$);
return $r_f$
```

MCALIB – Library Implementation using MPFR

- Inexact operation:
  - Implements simplified random sampling
  - Using gcc RNG – Uniform absolute random values used

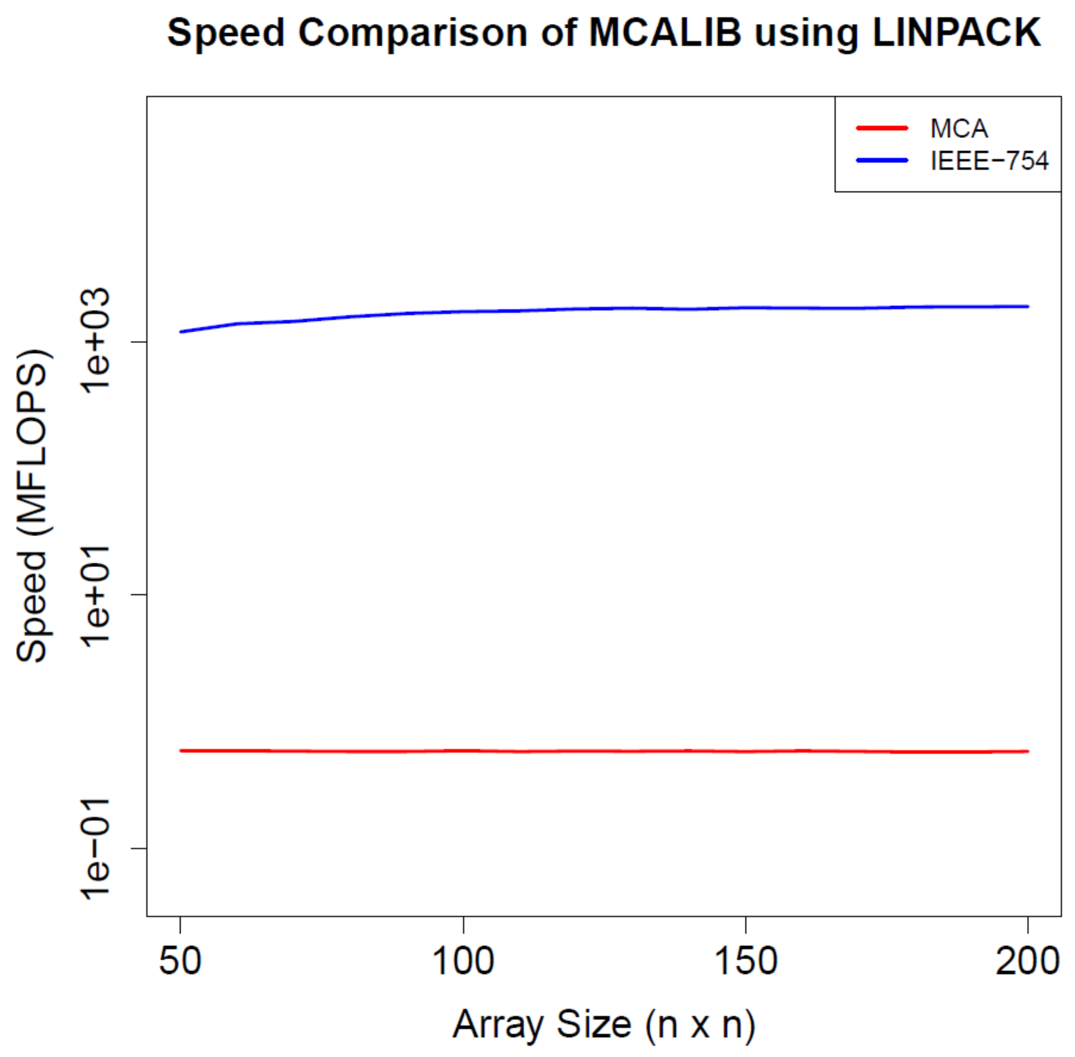
ALGORITHM 2: MCA Inexact Operation

```
Input: Precision $p + t$ MPFR_T variable $x$
Output: Precision $p + t$ MPFR_T variable $x$
if $x == 0$ then
    return $x$;
else
    $\xi_f$ = (rand()/RAND_MAX) - 0.5;
    $\xi$ = extend($\xi_f$, $p + t$);
    $\xi$ = mpfr_mul(pow(2, $e_x$ - ($t$ - 1)), $\xi$);
    $x$ = mpfr_add($x$, $\xi$);
    return $x$;
end
```

MCALIB – Performance Decrease

- Using MCALIB all FP operations are implemented using MPFR – SW based FP implementation
  - Results in significant decrease in FP performance
- Performance testing conducted using LINPACK
  - System used:
    - Intel Core 2 Duo Processor (2 GHz)
    - 3.7GB Ram
    - Array sizes between 50x50 and 200x200
  - Average IEEE-754 performance – 1718 MFLOPS
  - Average MCALIB performance – 0.585 MFLOPS
- Approximately 3000x decrease in FP performance but trivially parallelisable

MCALIB – Performance Decrease



Overview

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

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

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).

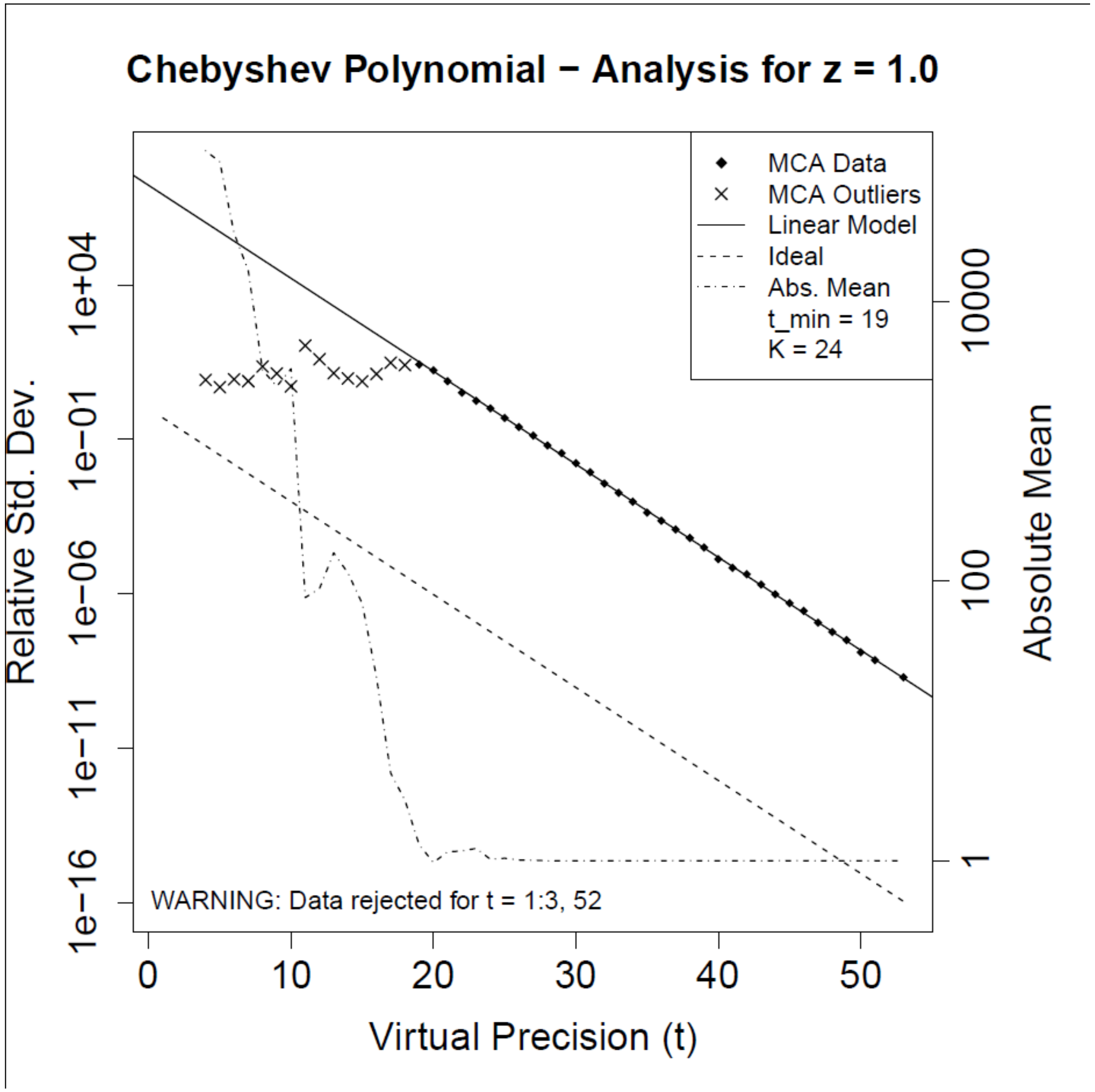
› For each t value, results are summarized by calculating relative standard deviation

› 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

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

Example – Error Detection & Optimization

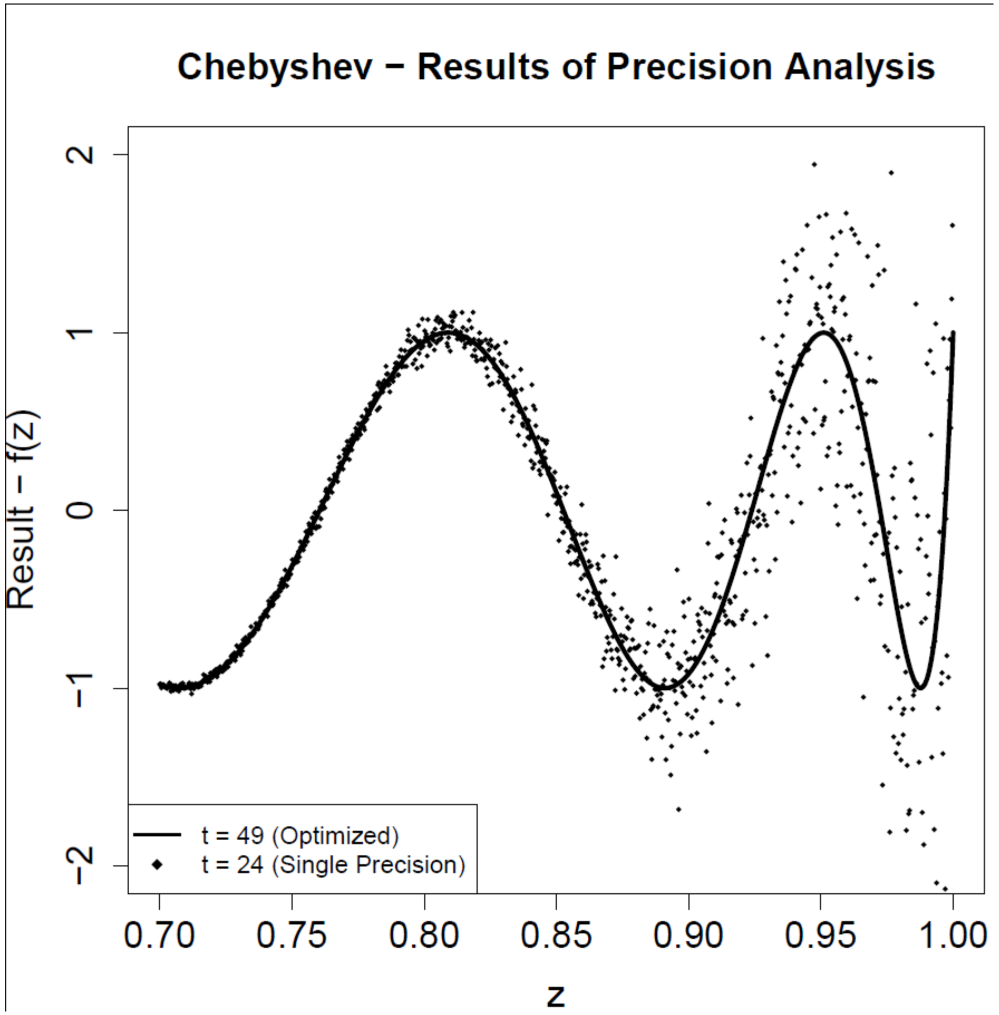


Example – Error Detection & Optimization

- 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

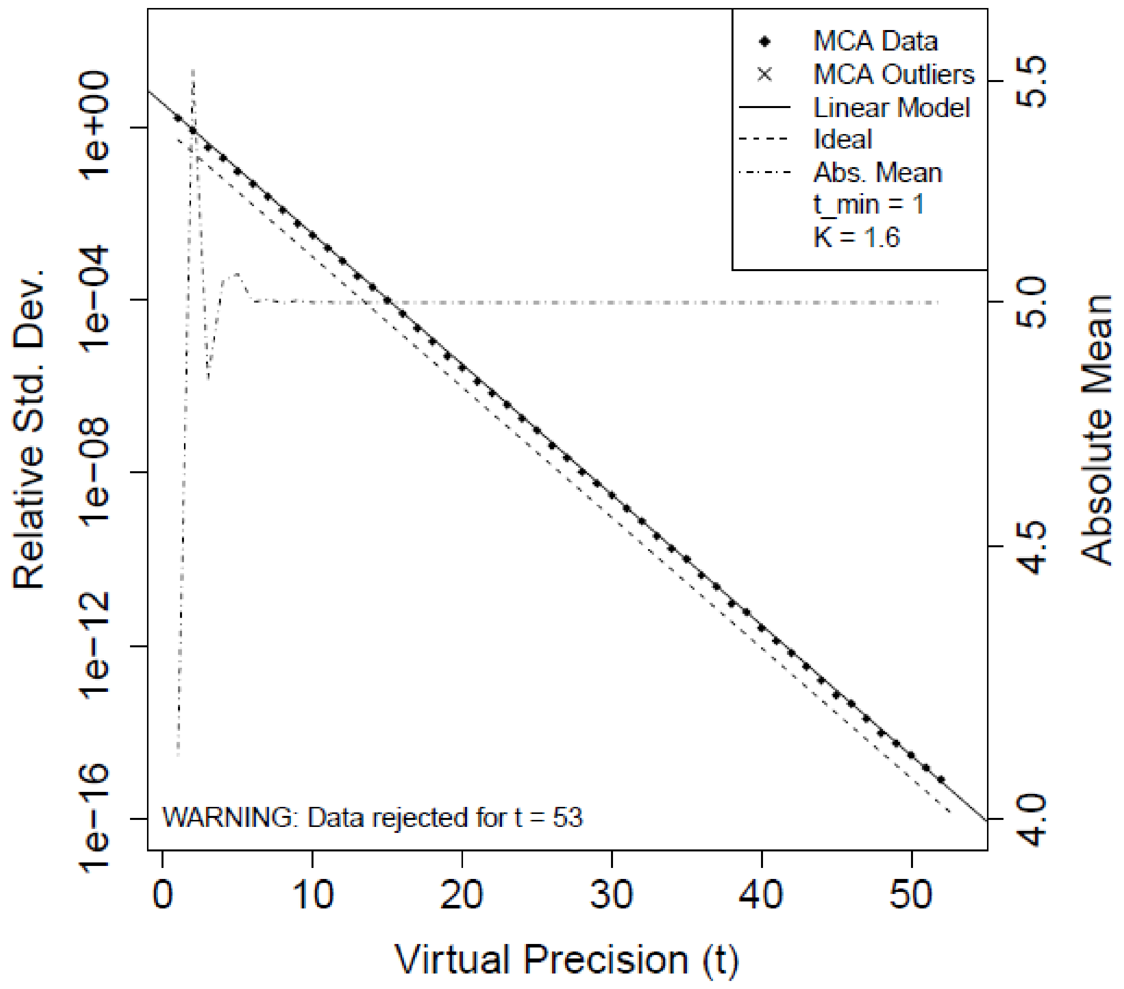
```
Input: Vector $X[1...n]$
Output: Sum $s$ of vector $X$
$s$ = 0.0;
$c$ = 0.0;
for $i$ = 1 to $n$ do
    $y = X[i] - c$;
    $t = s + y$;
    $c = (t - s) - y$;
    $s = t$;
end
Return $s$
```

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

Summation Algorithm – Analysis of Pairwise Method



Comparison of Algorithm Implementations

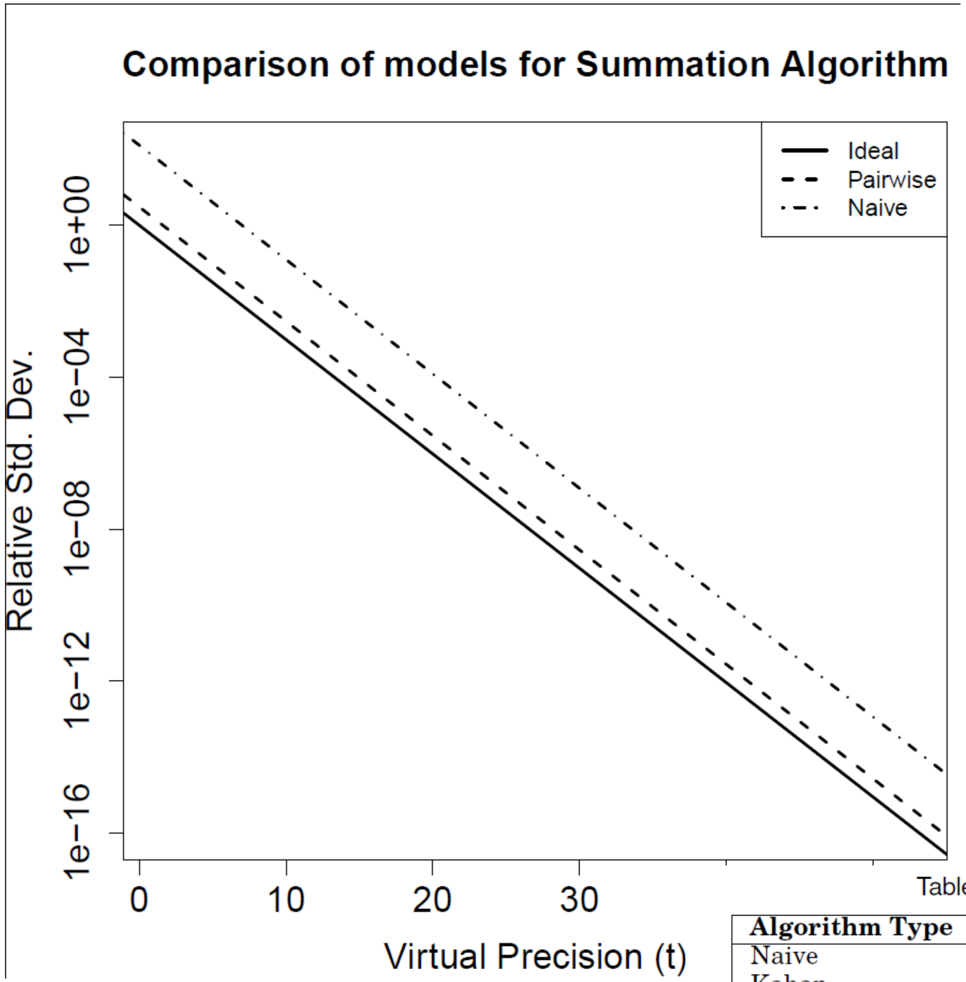


Table VI. Analysis of Summation Algorithms

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

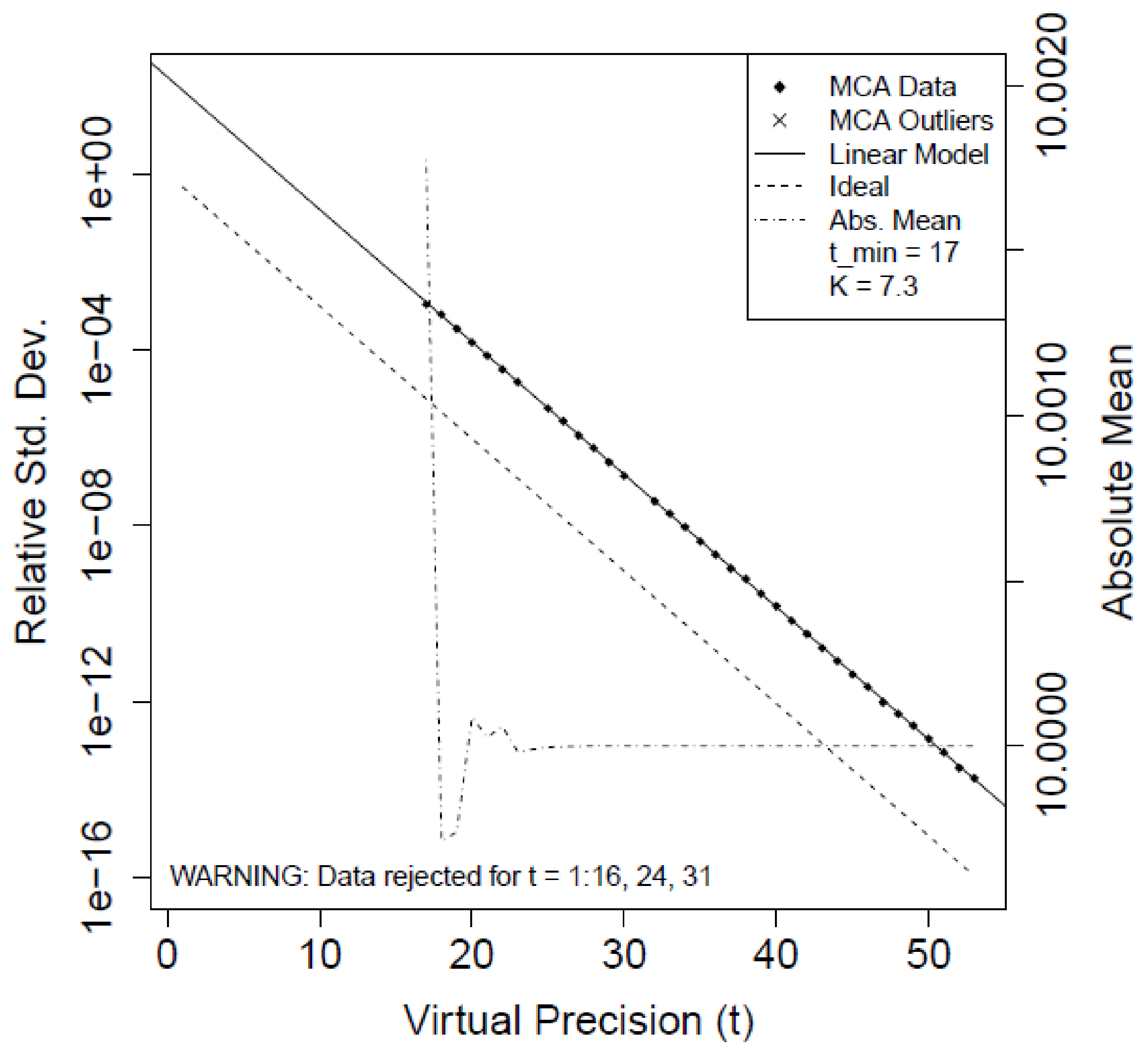
Note: Summation Algorithm Results - Naive, Kahan & Pairwise

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

LINPACK – Result L2 Norm v. Precision



Comparison of Algorithm Implementations

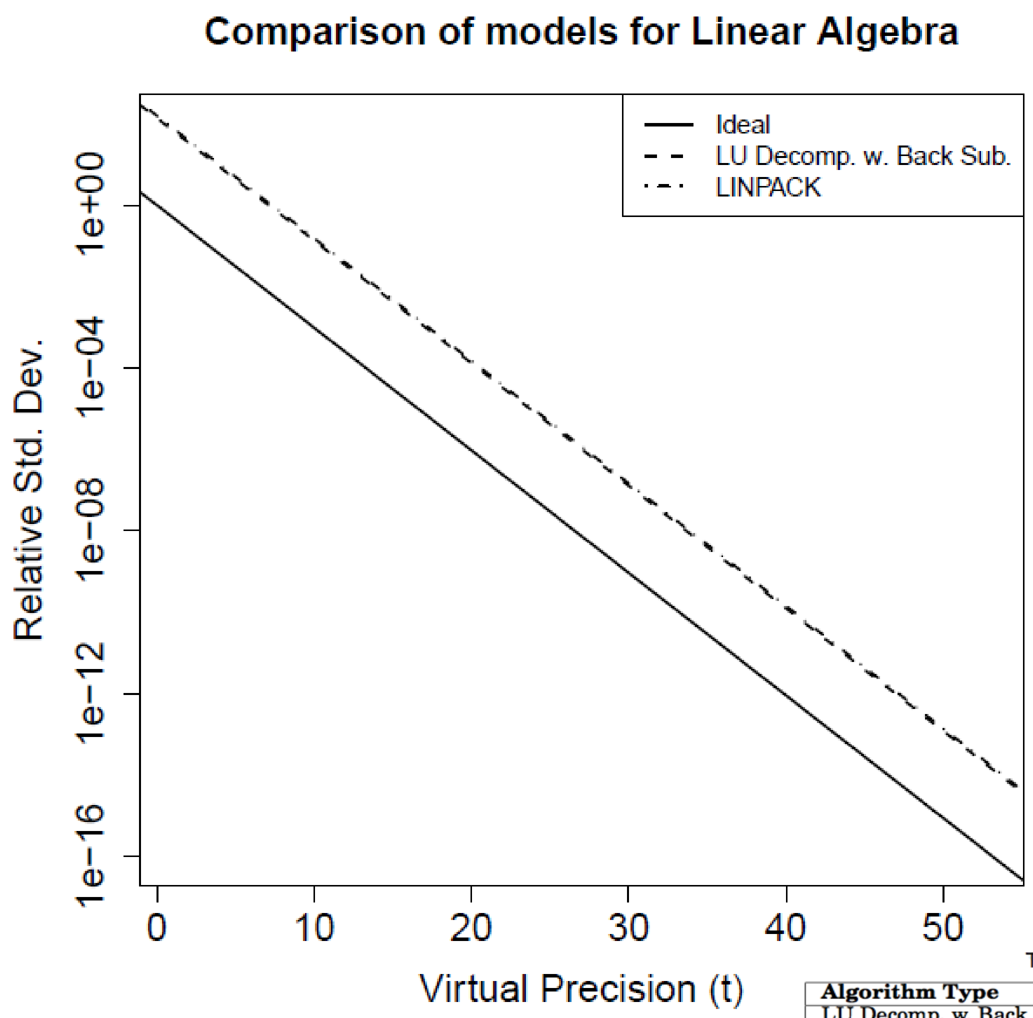


Table VII. Comparison of Linear Solvers

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

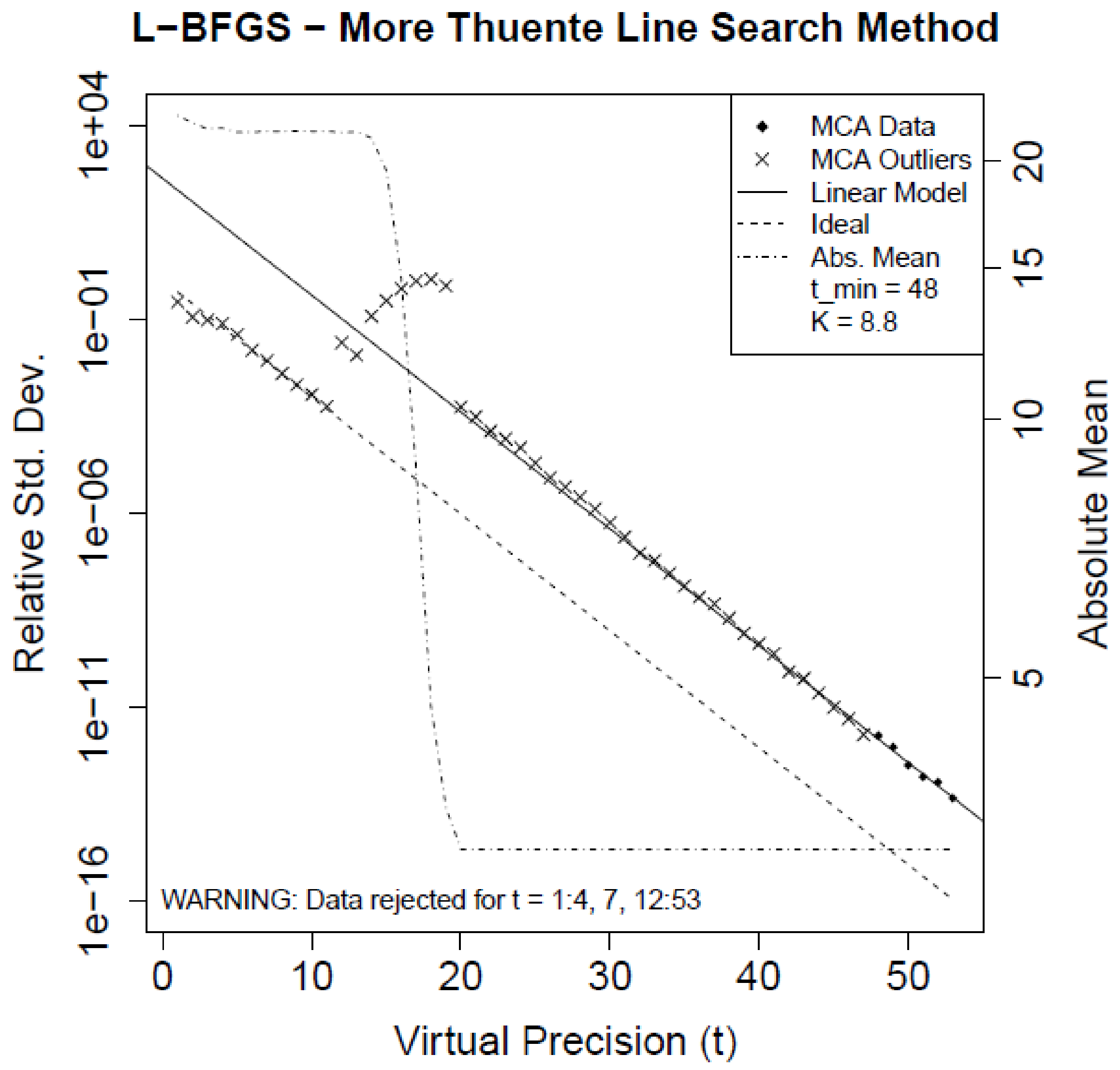
Note: Linear Solvers - Comparison of LINPACK and LU Decomposition with Back Substitution

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

THE UNIVERSITY OF SYDNEY

Non-Normal Result Data

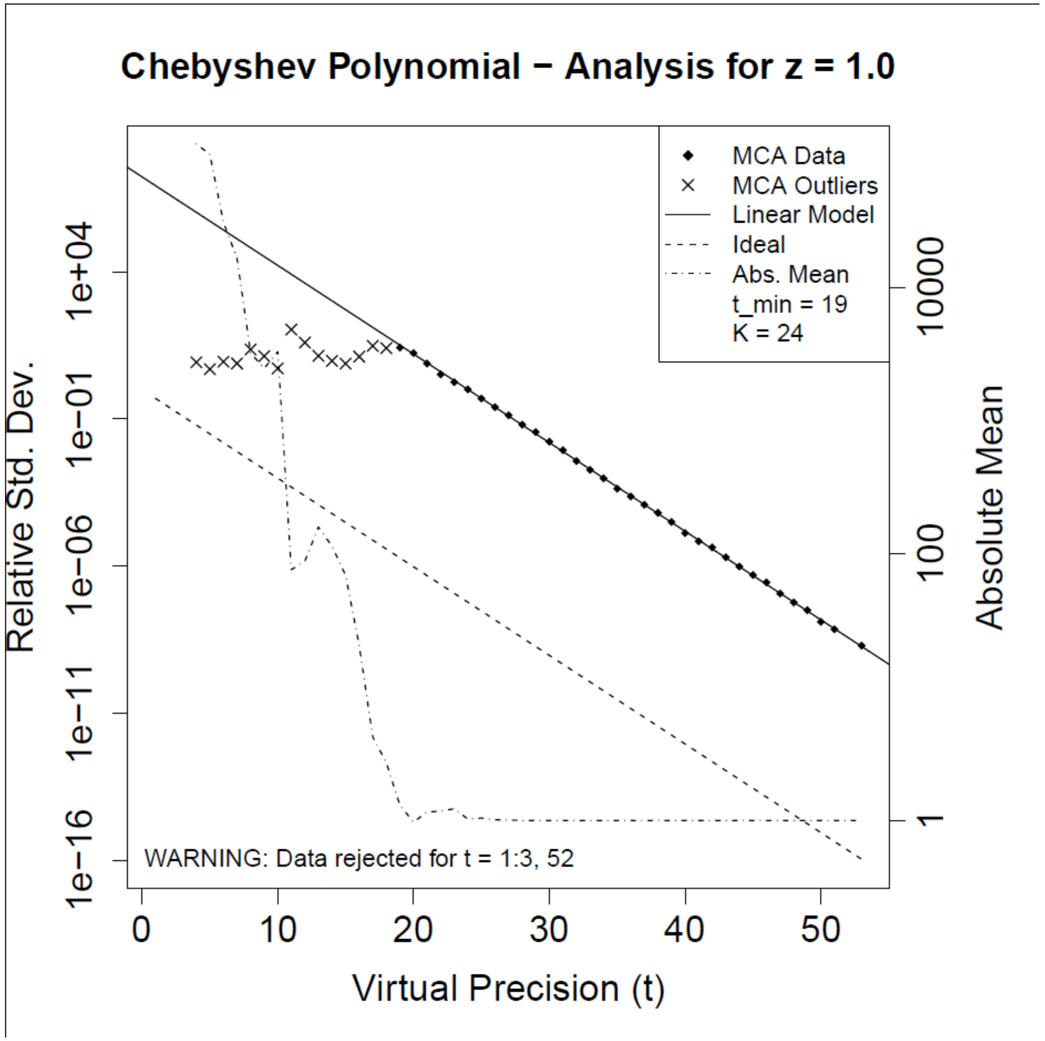


THE UNIVERSITY OF SYDNEY

Overview

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

Future Work

› Variance Reduction Techniques:
  - Reduce the required number of samples using variance reduction techniques as used in Monte Carlo Methods
  - Quasi-Monte Carlo Methods – Use low discrepancy sequences to increase the rate of convergence.