Floating Point Error Modelling in Microprocessor Design

Final Year Project Report
School of Engineering
October 2016

Rivan
24324051

Supervised by
Dr. Kuang Ye Chow

Department of Electrical and Computer Systems Engineering
Monash University
Sunway Campus

Assessment cover sheet

Unit and student details					
Unit code	ECE 4095	**Unit title**	PROJECT B		
Student ID	**24324051**	**Surname**		**Given names**	**Rivan**

Assignment details			
Title of assignment	**Final Report**	**Authorised group assignment**	Yes ☐ No √
Lecturer/tutor	**Dr. Kuang Ye Chow**	**Tutorial day and time**	**WEDNESDAY (12-6)**
Due date	**20/10/2016**	**Date submitted**	**20/10/2016**

Submission date and extensions

All work must be submitted by the due date. If an extension of work is granted this must be authorised on this form with the signature of the lecturer or tutor.

Extension granted until			
Lecture/tutor		**Signature**	

Plagiarism and collusion

Intentional plagiarism amounts to cheating in terms of Monash Statute 4.1 – Discipline. For further information see the university's Plagiarism policy including details of penalties and information about the plagiarism register.

Plagiarism - Plagiarism means to take and use another person's ideas and/or manner of expressing themselves and to pass these off as one's own, failing to give appropriate acknowledgement. This includes material from any source, staff, students or the internet - published and unpublished works.

Collusion - Collusion is unauthorised collaboration with another person or persons.

Penalties - If there are reasonable grounds for believing that intentional plagiarism or collusion has occurred, this will be reported to the Chief Examiner, who may disallow the work concerned by prohibiting assessment or refer the matter to the Faculty Manager.

Student statement and signature

- I have read the university's statement on cheating and plagiarism, as described in the Student Resource Guide
- This assignment is original and has not previously been submitted as part of another unit/subject/course
- I have taken proper care of safeguarding this work and made all reasonable effort to ensure it could not be copied
- I acknowledge that the assessor of this assignment may for the purposes of assessment, reproduce the assignment and:
  - provide it to another member of faculty
- I understand the consequences for engaging in plagiarism as described in Statute 4.1. Part III – Academic Misconduct
- I certify that I have not plagiarised the work of others or participated in unauthorised collusion when preparing this assignment.

Student signature	*Rivan*	**Date**	20/10/2016

Please note that it is your responsibility to retain a copy of your assignment

Table of Contents

List of Figures

List of Tables

List of Abbreviations

IEEE	Institute of Electrical and Electronic Engineers
FPGA	Field Programmable Gate Array
FPU	Floating Point Unit
GPP	General Purpose Processor
GPU	Graphics Processing Unit
MC	Monte Carlo Simulation
IA	Interval Arithmetic
AA	Affine Arithmetic
SMT	Satisfiability-Modulo Theory
GHR	Generalised Handelman Representation
IID	Independent Identically Distributed
PDF	Probability Density Function
CDF	Cumulative Density Function
CLT	Central Limit Theorem
KS Test	Kolmogorov-Smirnov Test
FFT	Fast Fourier Transform
DCT	Discrete Cosine Transform
CARE	Continuous Time Algebraic Riccati Equation

Abstract

Implementing numerical computation rarely uses the full-precision of the floating point hardware. Reconfigurable microprocessor has the flexibility in the number representation which allows tuning on the precision used in the computation to meet the range of error specified. By fine-tuning the precision used in mathematical computation, it is possible to optimize the memory usage, processing speed, power budget, latency, and maximum frequency while using less silicon area in the design. Thus, ignoring this potential will significantly limit the achievable performance.

This thesis proposes an approach that allows the user to exploit the flexibility in customising the computation precision. The approach makes use of the fact that *Monte Carlo Simulation* (MC) is able to obtain the optimum bounds statistically provided sufficient searching space. The approach assumes that provided the information of the distribution are known, the distribution can be reconstructed, thus enabling *reverse engineering* of MC simulation. The research assumes floating point error model standard, which is defined in polynomial form. Then by using moment based probability distribution fitting method, the distribution of the error is reconstructed, which is used to estimate the error bounds. The probabilistic bounding manages to obtained tight bound with better accuracy.

Acknowledgement

I would like to express my sincere gratitude to my project supervisor, Dr. Kuang Ye Chow for offering this final year project and consistently providing guidance and advises throughout the span of this project. I would also like to thank Dr. David Boland of Monash University Clayton for providing necessary feedback and results to complete this project. Next, I would also like to thank Arvind Rajan for providing guidance and the necessary toolboxes required to complete this project. I would like to thank Chee Chyuan Ang, for providing the necessary results and assisting the completion of this project. Last but not least, I would like to thank the lab assistants and technical staffs at the lab for providing the assistance completion of this project.

Chapter 1

Introduction

1.1. Project Background

Most of the present day *Digital Signal Processing* (DSP) applications are prototyped using floating-point arithmetic, but the final form of the numerical representation rarely uses the full-precision floating point representation due to the restriction of silicon area or power budget [1]. Instead, these constants and variables are redefined to the standard *Institute of Electrical and Electronic Engineers* (IEEE) single or double precision. The process of translation from full floating-point to limited fixed-point precision must be optimized due to the trade-off precision for processing speed, power budget, silicon area, etc.

The task of determining the most appropriate bit-width has piqued the interest of many design tool developers. To illustrate the problem, suppose a simple computation of $y = x - x^2$ where x is the real values input variable in which the computational path is provided in the Figure 1 below.

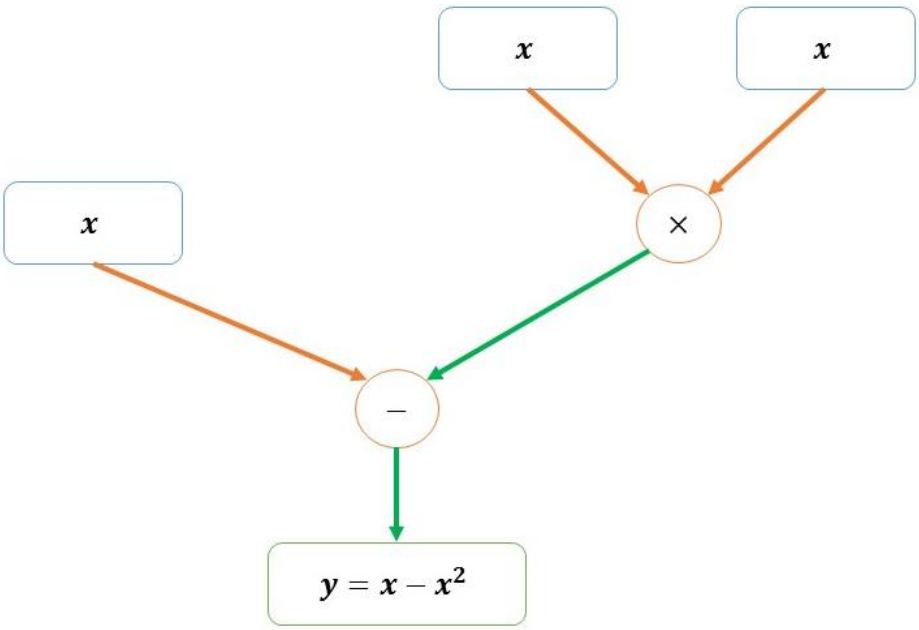


Figure 1. Computational Path of $y = x - x^2$

While 23 bit as given by IEEE-754 standard single precision floating-point is sufficient for most applications [1], it is redundant to assign all of the bit-width for a simple computation such as above. By assigning the appropriate bit-width for a computation, it is possible to maximize the performance on memory use, latency, clock speed, and data transfer of the hardware design while being restricted to some fixed silicon area or power budget. Therefore, A design tool that is capable of optimising bit-widths automatically will bring more freedom for the designer when designing a hardware with better performance subject to some fixed power budget or silicon area [2].

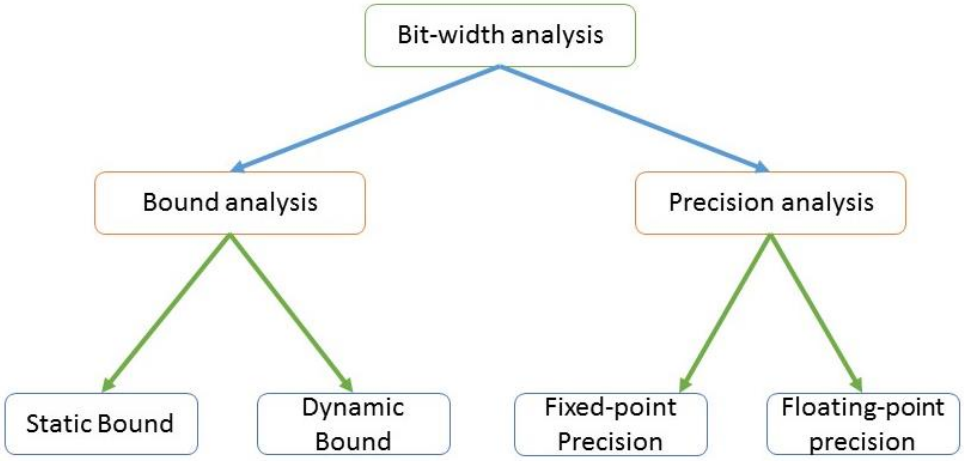


Figure 2. Bit-width analysis categorization [3]

There are two types of bit-width analysis in estimating error bound (see Figure 2.), which are static bound and dynamic bound. Static bound, while being simpler to implement compared to dynamic bound as only the input signal are needed, it is often subjected to overflows, which happen when the integer bit-width is too small, and precision loss or round-off error, which is due to the finite fractional bit-width [4]. On the other hand, dynamic bound relies on the user input to set the precision parameter by tuning the mantissa bit-width as per required, thus allowing much more efficient bit-widths analysis that provide bit-widths closer to the true bound [3].

Most of the mainstream existing methods that estimate the dynamic bound has mainly focused on either the scalability or tightness of bounds [2]. As such, there are methods that managed to obtain tight bounds, however is limited to simple computation [5-7], or methods that are able to work for larger problems, but resulted in loosen bounds and consequently limit the potential hardware optimizations [3, 8]. Due to the reasons above, there is a need for a new approach that is able to work on large scale computation without loosening the bounds or

limiting the potential hardware optimizations. This project proposed a new approach to estimate the error bounds of floating-point computation using moment-based distribution fitting technique. As the technique mainly consists of substitution to the input variables, it allows faster run time while preserving the tightness of the bounds.

1.2. Objectives

This project proposes a new approach to estimate the error bounds of floating-point computation using moment-based distribution fitting technique. This project will systematically compare the performance proposed method with existing mainstream methods.

1.3. Organisation of Thesis

Chapter 1 of the report introduces the project background and the objectives of this project. Chapter 2 provides a summary of the literature review on existing methods. The comparison of strength and limitation for each method are also presented in the same chapter. Chapter 3 provides complete overview on the moment-based distribution fitting technique. The results and discussions is presented in Chapter 4. This thesis is concluded in Chapter 5 which summarises the findings of the project and the possible of future research on the related topic.

Chapter 2

Literature Review

2.1. Fixed-Point Precision vs Floating Point Precision

Due to the limitation of the hardware, such as silicon area, processing speed, power budget, etc., most of the time the numerical computations are redefined to IEEE-754 standard single or double fixed-point precision, which format can be seen in Figure 3 below, instead of a full-precision floating point model. The fixed-point precision is mostly implemented for large scale designs due to its simple implementation and only the characteristic of the input signals are required. The fixed-point precision is also used in most of the low-cost embedded microprocessors due to the lack of *Floating Point Unit* (FPU). In addition to that, fixed-point precision is also believed to provide more conservative bit-width estimations. However, the process of translation from full to fixed-point will introduce quantization error or round-off error due to the finite fraction bit-width [4]. While the error introduced by the rounding of any single value may be insignificant, over the course of the computation, these errors may accumulate and hence cause a significant deviation from the nominal result [9].

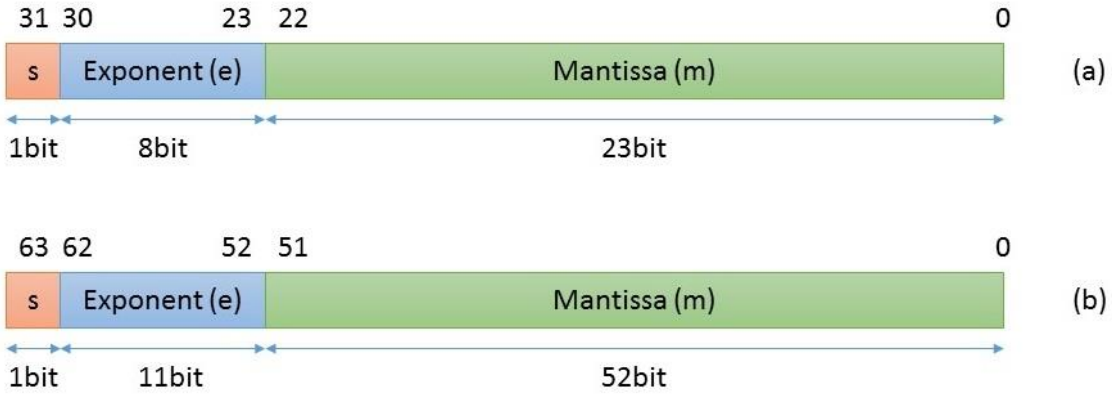


Figure 3. (a) IEEE-754 Standard Single Precision format[1] (b) IEEE-754 Standard Double Precision format.

Another commonly used precision model in precision analysis is floating-point precision, which allows more flexible custom formats. General floating point format as specified by IEEE-754 standard single or double precision consists of three fields, which are sign, exponent and mantissa (see Figure 4.). The precision of the floating point model are determined by the mantissa bit-width, and 23 bit (IEEE standard single precision) is sufficient

for most applications [1]. The floating point model gives the flexibility to the user to tune the precision used in an algorithm. By fine-tuning the precision used in an algorithm, it is possible to maximize the performance on the memory use, latency, clock speed and data transfer while using less silicon area. The freedom in tuning precision used throughout an algorithm has received a considerable amount of interest, especially from reconfigurable hardware community. Furthermore, numerous research have been done on the potential benefits of fine-tuning the precision used in an algorithm on a *Field Programmable Gate Array* (FPGA), mainly in DSP domain [9, 10].

In this project, we would be focusing more on the floating point model, as it shows more potential benefits compared to the fixed-point model as mentioned above. Same multivariate polynomial model is used in order to explain the existing mainstream methods.

2.2. Simulation Method

Simulation method, especially *Monte Carlo Style simulation* (MC), is one of the most commonly used method to evaluate the error propagation in the implementation of custom floating-point model. This method is conceptually simple to understand and operationally straight-forward. Kum. et. al. [11] have applied Monte Carlo statistical simulation in order to optimize word-length iteratively. The IEEE standard double precision is normally assumed to have virtually infinite precision in the simulation study [1]. While the simulation method manages to obtain satisfactory bounds, it is very inefficient for multivariate model due to the large searching space [11, 12]. Suppose a polynomial $f(x, y) = x^2y - xy^2$ where $x \in [-2,6]$ and $y \in [-3,5]$ are uniformly distributed random variables to be evaluated over 10 million iteratively through Monte Carlo Simulation. The result obtained, which can be seen in Figure 4, shows the histogram of the polynomial model and the red circles show the bound of the polynomial. While the results obtained manages to converge the output error towards zero after certain iterations and obtain the optimum results, however, it is quite inefficient and the estimation does not form a bound as there is a possibility that the corner cases are missed [9].

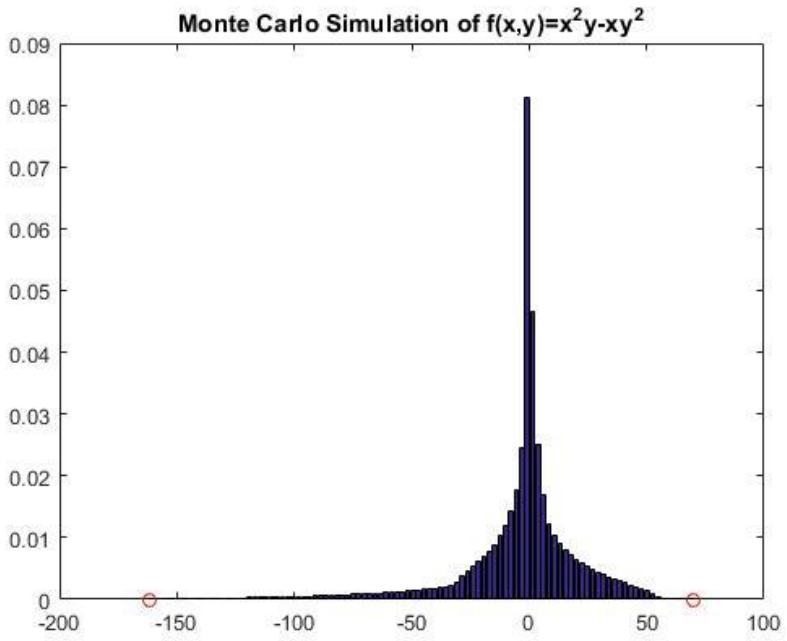


Figure 4. Monte Carlo Simulation of $\boldsymbol{f(x,y) = x^2y - xy^2}$

With the mentioned shortcoming, Monte Carlo Simulation is only applicable to small problems or low error bound reliably. Common MC efficiency improvement techniques such as importance sampling are designed to achieve variance reduction. These techniques are not suitable in the work related to error bound determination.

2.3. Bound Estimation

Another technique is based on static bit-width analysis. Currently there are two methods based static bit-width analysis which are widely used, which are *Interval Arithmetic* (IA) and *Affine Arithmetic* (AA). Interval Arithmetic has the tendency to overestimate bit-widths, which lead to pessimism as the overestimation accumulates exponentially along the computation path [12]. A. B. Kinsman and N. Nicolici [5, 13] have developed a range refinement method using *SAT-Modulo Theory* (SMT), but their approach has limitation in which the run time grows significantly with the problem complexity. Fang et. al. [1, 4] proposed static bit-width analysis by using Affine Arithmetic model in DSP applications. Although their static error analysis is computational efficient and have good accuracy, it was restricted by the assumption that all the uncertainty in the model are independent, which is not necessarily true for all inputs as strong correlations between inputs may lead to over-estimations [4]. Whilst the approach of using fixed-point precision is a suitable paradigm for *Graphics Processing Units* (GPUs), as they

mostly consist of many parallel floating point units, for reconfigurable hardware such as FPGAs, this significantly limit the achievable performance, as reconfigurable hardware has the ability to implement any precision required to meet a given specification.

In order to remove the limitation of the use of fixed-point precision, D. Boland and G. A. Constantinides [9] introduced a general analytical approach to deduce the worst case bounds through algebraic simplification of polynomial error model under a given floating point precision representation, which is Handelman Representation. The bounds obtained through this method are shown to perform better compared to other existing methods in general, enabling the possibility to design a hardware that still achieves the same error specification as existing methods, while using less silicon area. Although their approach can provide provable bounds which can achieve tighter bounds compared to both interval and affine arithmetic and running significantly faster and has better scalability than SMT, it is still time consuming for large scale algorithms. By make use of general error model for floating point computation, a new method to calculate the bounds on any distributions of inputs variables by using moment based probability distribution fitting approach has been proposed in this project.

2.3.1. Interval Arithmetic

Interval arithmetic (IA) was introduced by Ramon E. Moore in the 1960s [14-16]. IA was "rediscovered" by researchers from many applied fields after being neglected for decades, due to the flexibility and effectiveness for range analysis. In interval arithmetic, a real quantity of x is represented by an interval $\bar{x} = [x_{min}\_x_{max}]$ of floating point numbers. These intervals are then propagated through the computation, which calculate the new worst case bound at every stages of computation paths based on the basic rules [6] as given below.

$$[x_1, x_2] + [y_1, y_2] = [x_1 + y_1, x_2 + y_2]$$
$$[x_1, x_2] - [y_1, y_2] = [x_1 - y_2, x_2 - y_1]$$
$$[x_1, x_2] \times [y_1, y_2] = [\min(x_1y_1, x_2y_1, x_1y_2, x_2y_2), \max(x_1y_1, x_2y_1, x_1y_2, x_2y_2)] \quad (1)$$
$$\frac{[x_1, x_2]}{[y_1, y_2]} = \begin{cases} undefined, if\ 0 \in [y_1, y_2] \\ \left[min\left(\frac{x_1}{y_1}, \frac{x_2}{y_1}, \frac{x_1}{y_2}, \frac{x_2}{y_2}\right), \max\left(\frac{x_1}{y_1}, \frac{x_2}{y_1}, \frac{x_1}{y_2}, \frac{x_2}{y_2}\right)\right] \ otherwise \end{cases}$$

In which the rules above can be expressed as

$$[x_1, x_2] \odot [y_1, y_2] = \quad (2)$$

$= [\min(x_1 \odot y_1, x_2 \odot y_1, x_1 \odot y_2, x_2 \odot y_2), \max(x_1 \odot y_1, x_2 \odot y_1, x_1 \odot y_2, x_2 \odot y_2)]$
Where $(\odot \in \{+, -, *, /\})$ [2, 17]

IA often results in an interval with much wider range compared to the exact range in computation. This is caused by the loss of dependency information between arithmetic operations called the dependency problem. When a variable is being used multiple times in a chain of arithmetic operations, the interaction of the same variable across different operations are not taken into account in IA. As an example to this, the IA evaluation of $f(x, y) = x^2y - xy^2$, given $x \in \bar{x} = [-2, 6]$ and $y \in \bar{y} = [-3, 5]$ respectively, will produce result of interval $[-258, 270]$ (detailed working see Appendix 1.). This problem of IA often leads to overestimation of the bit-width, in which the overestimation will accumulate exponentially along the computation path [12]. In a chained computation, the results of one step are used as the inputs for next step, the overestimation factors of the individual steps tend to get multiplied, which cause the final intervals to be too wide to be useful [14].

2.3.2. Affine Arithmetic

Fang et. Al. [1, 4, 12] have introduced the model based on affine arithmetic into the verification of floating point precision effects in DSP applications as the solution to reduce the dependency problem in IA. Affine arithmetic avoids the dependency problem by restricting the polynomials to first order to ensure the polynomial does not contain any dependencies [2]. *Affine Arithmetic* (AA), as defined by J. Stolfi and L. H. de Figueiredo [14] is a model for self-validated computation which produces guaranteed bounds for computed quantities while preserving the correlation between interval. By assuming floating point error as a range, the representation and computation of floating-point number can be modelled by using AA.

A conventional error model of floating-point can be expressed as $x_f = x + x2\Delta\delta$, where $\delta \in [-1,1]$ is error term and error bound $\Delta = 2^{-m}$. It is important to note that the floating-point error model above is in an affine form, where m is the mantissa bit-width [1, 18], x_f is the floating-point representation of a real number x, and the floating-point approximation or rounding is represented by the uncertainty term $2x\Delta\delta$.

In order to apply IA to error analysis by using AA model, the floating-point model is to be represented in ranges. Assume a variable x is given in the range $\hat{x} \in [x_0 - x_1, x_0 + x_1]$ in

which the affine form can be expressed as following: $\hat{x} = x_0 + x_1\delta_r$. By applying the conventional floating point model into the expression, the floating point representation is

$$\hat{x}_f = x_0 + x_1\delta_r + (x_0 + x_1\delta_r) \cdot 2\Delta \cdot \delta_f \quad (3)$$

In order to reduce the expression above into an affine form, the bounding operator B is the introduced as following: $B(x_0 + \sum_{i=1}^{N} x_i\delta_i) = \sum_{i=0}^{N}|x_i|$ which computes a hard upper bound of its argument [1, 4]. When the bounding operator B is applied on the floating point representation, an upper bound of x_f with associated error $E(\hat{x}_f)$ can be obtained.

$$\hat{x}_f \leq x_0 + x_1\varepsilon_r + B(x_0 + x_1\delta_r) \cdot 2\Delta \cdot \delta_f \quad (4)$$

$$E(\hat{x}_f) = \hat{x}_f - \hat{x} \leq (|x_0| + |x_1|) \cdot 2\Delta \cdot \delta_f \quad (5)$$

Note that the sign '$\leq$' implies that the range on the left is included on the right. In the associated error equation above, the error is related to the insufficient information given on the floating-point error and the exact magnitude of x [1]. By using similar approach, AA models for floating-point range computations can be derived for standard operations, which can be expressed as following:

$$\hat{z}_f \leq \left(\hat{x}_f \odot \hat{y}_f\right) + B(\hat{x}_f \odot \hat{y}_f)) \cdot 2\Delta \cdot \delta_f \quad (6)$$

$$E(\hat{z}_f) \leq \left(\hat{x}_f \odot \hat{y}_f\right) - (\hat{x} \odot \hat{y})$$
$$where\ (\odot \in \{+, -, *, /\})\ and\ \delta_f \in [-1,1] \quad (7)$$

The benefit of using AA model over IA model is shown in [1], in which AA-based error ranges carry information about the error sources, thus enabling cancellation of error without leading to overestimation. Furthermore, the error introduced after each operation in AA model is linear [16] compared to IA where the error growth exponentially due to the accumulated overestimation in computation paths. This will benefit the AA model over IA model, especially for large scale problems, such as chained computations.

However, as many functions, including general multiplication and division, are not affine, therefore approximations must be made [9]. Difference that presents between the approximation made and true bound of the higher order terms may result in overestimation of the bounds, while the information between lower and higher order terms is not preserved [2]. This will sometimes cause the affine arithmetic to perform worse than IA.

Suppose a polynomial $f(x, y) = x^2y - xy^2$ and the range of x and y are given as $x \in \bar{x} = [-2, 6]$ and $y \in \bar{y} = [-3, 5]$ respectively. In order to perform precision analysis on the

polynomial, the interval bound of both x and y must be converted from IA interval into affine form of: $\bar{x} = x_0 + x_k\delta_k$ where δ_k is the newly introduced noise symbol that does not exist in other affine form, x_k is the half-width of the interval $\bar{x}$, and x_0 is the midpoint of interval $\bar{x}$. From the interval of x and y provided, the affine form of both x and y can be expressed as: $\hat{x} = 2 + 4\delta_k$ and $\hat{y} = 1 + 4\delta_k$. Then any scalar operation can be performed on the affine model based on the expression above. After the operation, the affine form of the results is obtained as: $\hat{d} \leq 2 + (12 + 18 \times 2\Delta)\delta_x - (8 + 10 \times 2\Delta)\delta_y + (14 + (44)2\Delta)2\Delta\delta_a + (112 + 210 \times 2\Delta)2\Delta\delta_b - (80 + 130 \times 2\Delta)2\Delta\delta_c + (26 + (66)2\Delta)2\Delta\delta_d$ (for detailed derivation see Appendix 2.) Then, by converting the affine form back into interval set, by setting the noise symbol ε_k in the range of $[-2^{-8}, 2^{-8}]$, and the bit-width mantissa m to tune the precision 2Δ in order to obtain the range interval, which is $[-244.8470, 232.7059]$.

2.3.3. Bound Deduction through Handelman Representation

D. Boland and G. A. Constantinides [6, 9] proposed an approach to deduce the worst case bounds through algebraic simplification of polynomial error model. The proposed method consists of several stages. Firstly, the potential range is to be represented in a polynomial form. According to N. J. Higham [6, 9, 19], for a real value of x, the closest approximation $\hat{x}$ of radix-2 floating point x can be expressed as

$$\hat{x} = x(1 + \delta), (|\delta| \leq \Delta, where\ \Delta = 2^{-m}) \quad (8)$$

Where m represent the mantissa bits used or the precision. By using similar approximation, the radix-2 floating-point as a result of any scalar operations complying with IEEE standard arithmetic can be bounded in the following expression.

$$\widehat{x \odot y} = (x \odot y)(1 + \delta_i), where\ (\odot \in \{+, -, *, /\}) \quad (9)$$

For every stages of the computational path, a new single polynomial δ_i is introduced to represent the error variables due to the round-off or truncation of the results. After the polynomial representation is obtained, the next stage involves cancelling all the monomials in order to obtain the bounds. D. Boland and G. A. Constantinides [9] proposed a new heuristic approach to find an upper bound and lower bound such that $\hat{\gamma}_{upper} \geq \gamma_{upper}$ and $\gamma_{lower} \geq \hat{\gamma}_{lower}$ to be as small as possible.

By applying Handelman discovery [9, 20] on a result emanating from real algebra, a *Generalised Handelman Representation* (GHR) polynomial which is positive over the compact set S can be expressed as below.

$$S = \{\delta \in \mathbb{R}^n | \forall(\Delta - \delta_i \geq 0) \wedge (\Delta + \delta_i \geq 0)\}$$

$$p_{ghr} = \sum_{j \in \mathbb{N}} c_j \prod_{i=1}^{n} \left(\Delta^{|\mu_{i,j}|} - \delta^{\mu_{i,j}}\right)^{\alpha_{i,j}} \left(\Delta^{|\mu_{i,j}|} + \delta^{\mu_{i,j}}\right)^{\beta_{i,j}} \quad (10)$$

Where $\mu_{i,j}$ are arbitrary integer vectors.

Then, in order to deduct the lower and upper bounds that satisfy $\hat{\gamma}_{lower} \leq f(\delta) \leq \hat{\gamma}_{upper}$, due to the fact that $\hat{\gamma}_{upper} - f(\delta)$ and $f(\delta) - \hat{\gamma}_{lower}$ are positive over the compact set of inequalities S, the upper bound and lower bound is equivalent to the form below.

$$f(\delta) - \hat{\gamma}_{lower} = p_{lower\_ghr} \quad (11)$$

$$\hat{\gamma}_{upper} - f(\delta) = p_{upper\_ghr} \quad (12)$$

The complexity of this method lies in the fact that there are numerous monomials in to cancel and several ways to cancel any given higher order monomials $f(\delta)$ using the Generalised Handelman Representation. Based on this idea, D. Boland and G. A. Constantinides proposed a heuristic in which the monomials are cancelled starting from highest order monomials, while at the same time the absolute value of the coefficients of lower order monomials is also reduced. By cancelling the monomials this way, the algorithm termination is guaranteed [9].

The approach is proven to have more control over the trade-off between the quality of the bounds and the runtime compared to any existing methods. In addition to that, the approach is able to obtain much tighter bounds, with less precision [9]. This will enable a new hardware design to obtain similar performance while using less silicon area. However, the approach still have scalability problem, in which the approach is still time consuming for large scale computations, especially when more complex functions are involved [6].

In order to demonstrate the use of the method, an example of multivariate polynomial as following is consider: $f(x, y) = x^2y - xy^2$. Then, the floating point precision for each computational stages is introduced into the polynomial equation, hence polynomial representation of the function can be expressed as $f(\delta) = x^2y - xy^2 + (x^2y - xy^2)\delta_1 + x^2y\delta_2 + x^2y\delta_1\delta_2 - xy^2\delta_3 - xy^2\delta_1\delta_3$. The next steps involve finding the appropriate Handelman representations that cancel the monomials starting from the highest order

monomials, which in this case is either $x^2y\delta_1\delta_2$ or $-xy^2\delta_1\delta_3$. After all monomials are cancelled out, the lower bound obtained will be $\hat{y}_{lower} = x^2y - xy^2 - 2x^2y\Delta + 2xy^2\Delta + x^2y\Delta^2 - xy^2\Delta^2$ and the upper bound will be $\hat{y}_{upper} = -x^2y + xy^2 + 2x^2y\Delta - 2xy^2\Delta - x^2y\Delta^2 + xy^2\Delta^2$ (detailed working see Appendix C.), which is consist of both real values of x and y and the precision or error bound defined by user Δ. By substituting the all the variables and the error bound to be 2^{-8}, the bound is estimated to be $[-160.7183, 69.4252]$

2.4. Post-Processing Improvement

2.4.1. Satisfiability-Modulo Theory (SMT)

While most of the time AA performed better than IA, AA can still result in overestimation, predominantly when strong non-affine operations involved in the computation, a general example to be division. While this case is rare to happen in DSP design, it occurs in scientific calculations frequently [4, 5, 13]. Because of this limitation of AA and the inability of simulation based methods such as Monte Carlo-Style Statistic Simulation in providing efficient yet robust variable bounds, A. B. Kinsman and N. Nicolici proposed a range refinement method based on Satisfiability-Modulo Theory [5, 13].

It is important to note that *Satisfiability-Modulo Theory* (SMT) is not another variant of method in performing bit-width analysis, but is a post processing improvement method in order to refine the ranges obtained from interval arithmetic process. SMT method perform range analysis by using binary search algorithm iteratively on the range obtained through interval analysis. For each SMT instance evaluated contains specified Boolean constraints, which is used to successively remove over-estimation from the original bounds, thus result in tighter bounds.

Although the SMT approach manage to obtain tighter bounds than original bounds, it has a limitation in which the processing time grows significantly as the problem become more complex [9]. This is the trade-off between the SMT timeout and the accuracy of the bounds. The SMT assumes satisfiability when terminated for short timeouts, resulting in more bit-width to be allocated. As the timeout increases, the range of the bound will decrease slowly until the specified bounds are obtained [5, 13].

2.5. Contribution Strength/ Limitation

This section summarizes the strength and limitation of all existing methods discussed in this chapter to provide some insight on the existing methods mentioned in the previous sections.

Table 1. Comparison of existing methods

Method	Advantage	Disadvantage
Monte Carlo Simulation	• Straight forward and easy to implement • Able to obtain satisfactory bounds for small problems • Application independent	• Inefficient, especially for multivariate model due to the large searching space. • There is a possibility that the corner cases are missed [9], resulting in underestimation of the bounds. • Only applicable for small scale problems
Interval Arithmetic (IA)	• Flexible and easy to implement • Able to obtain bounds within very short execution time	• The correlation between operands are not taken into account, which lead to unacceptable overestimation [2, 6]. • Only applicable for small scale problems.
Affine Arithmetic (AA)	• Able to keep track of the correlations between operands • Able to provide a fairly tight bound estimation • Applicable for large scale problems, such as chained computations.	• General function, such as division and multiplication are not affine, thus, approximations of the bounds must be made. • The difference between the exact bound and the approximations made will cause the

		dependency information between the lower and higher order terms to be lost; the overestimation may be worse than IA in some cases.
Bound Deduction through Handelman Representation	• Provide more control over the trade-off between quality of estimated bounds and execution time. • Able to obtain much tighter bounds, with less precision.	• Only applicable for small scale problems • Time consuming for large scale problems with complex functions such as square roots and exponential.
Satisfiability-Modulo Theory (SMT)	• Simple to implement, as it is taking the bounds estimated through IA and perform range refinement by removing the over-estimation through Boolean constraint. • Able to obtain robust bit-width even for general functions, such as multiplication and division. • The bounds obtained is tighter than AA model.	• Processing time grows significantly as the complexity of the problems increases. • The tightness of the bounds obtained is dependent on the Boolean constraints used to refine the bounds

2.6. Summary

The existing methods in estimating bounds have been fully studied in this chapter. IA method provides a simple yet efficient method. However, it often results in overestimation of the bounds by large margin, rendering the bound to be useless for application. AA method is introduced to mitigate the dependency problem of IA, however, most of general operations, such as multiplication and division are not affine. This causes some approximations have to be made. For large chained computation, the final estimate of the bound may lead to over pessimistic. SMT method is introduced as bound refinement method to refine the overestimated bound obtained from IA through Boolean constraint. While it is able to provide better estimate of the bound, it requires large resources as the complexity of the problem increases. Handelman representation is introduced as a method to deduce the bound through algebraic manipulation. While it is able to deduce the bound with less precision, is only applicable for small scale problems.

MC simulation estimates the bound by simulates all possible input vectors iteratively over large searching space. While it is able to obtain optimum results, it is too inefficient due to the high computational efforts. A different approach is proposed in the following chapter. The approach assumes that provided moment of the distribution is known, the numerical distribution of the MC can be reconstructed through known distribution fitting technique.

Chapter 3

Proposed Method

This project investigates the feasibility of using statistical moment to perform bound estimation. By using the polynomial representation of floating point in [9], the method is to estimate the bound through propagation of moments has its root in the works of Y. C. Kuang et. al. [21]. The method takes in the floating point error model polynomial of $f(\delta_1, \delta_2, \delta_3, \dots, \delta_N)$, where the δ_i is the error introduced through every computational stages, and input distributions for each variables then computes the moment of the polynomial. There is no need to keep track of the bound and interactions between variables every step of the computation. The net result of interactions between correlated variables are encoded in the moments of the output variable. Then by making use of the moment obtained, the distribution of the output variable can be used to estimate the bounds of the model. The general framework of the proposed method is given in Figure 5.

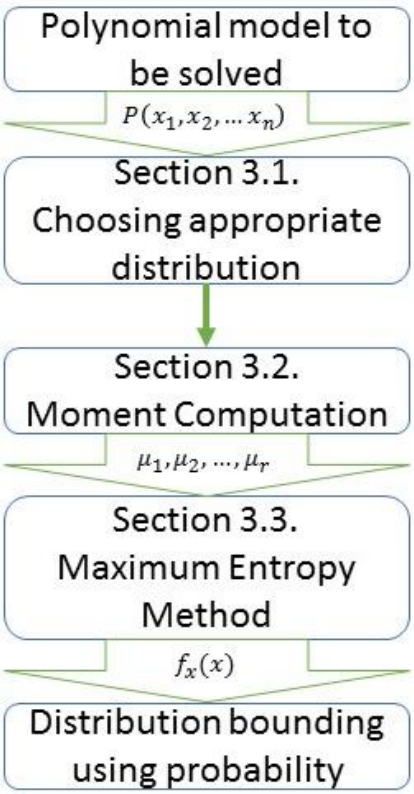


Figure 5. Framework of proposed method

3.1. Input Distribution

Distribution is used to propagate the moments. There are two type of distributions used in this project, which are standard uniform distribution and non-standard power distribution. Different distribution will affect the probabilistic outcomes, which may provide better estimation of the bounds.

3.1.1 Uniform Distribution

Translated uniform distribution is the most generally used distribution in uncertainty evaluation. It is due to the maximum entropy distribution among all finitely supported continuous distributions [21]. For a given interval of $[a, b]$, a standard uniform distribution is given in (13).

$$f(x) = \frac{1}{2L}$$
$$where\ L = \frac{b-a}{2} \tag{13}$$

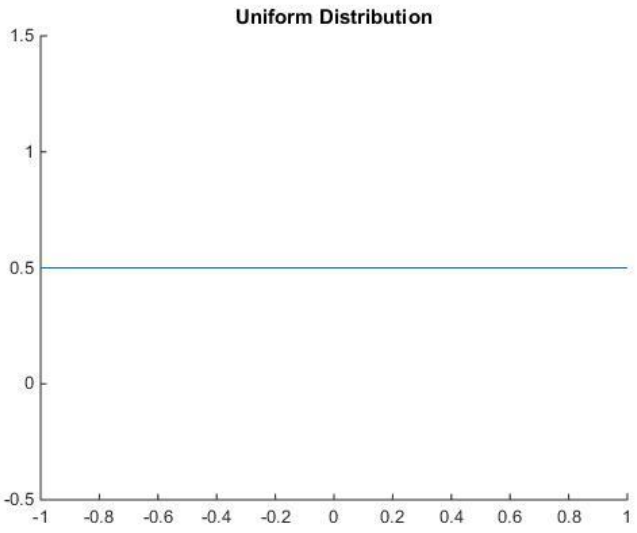


Figure 6. Uniform Distribution

Uniform distribution often leads to underestimation of the bounds in most nonlinear cases. This is because uniform distribution has constant probability within a finite interval $X \in [\mu - \sigma, \mu + \sigma]$, which is fairly redundant as distribution at the tails are more likely to affect the bounds rather than the distribution at the mean. A non-standard distribution is introduced in the next section.

3.1.2. Power Distribution

Power distribution is introduced to address the underestimation problem of uniform distribution and to improve the efficiency of the proposed method. Unlike the standard uniform distribution, the power distribution is more concentrated at the tail, in which the shape of distribution can be varied by the user. A general power distribution can be expressed as

$$f(x) = \frac{q+1}{2L}\left|\frac{x}{L}\right|^{q} \qquad (14)$$
$$where\ q \geq 0\ and\ L = \frac{b-a}{2}$$

The shape of the distribution can be varied the by fine-tuning the q parameter whereby $q = 0$ provides a standard uniform distribution. Higher q parameter will have higher concentration at the tail, while the information at around the mean is lost.

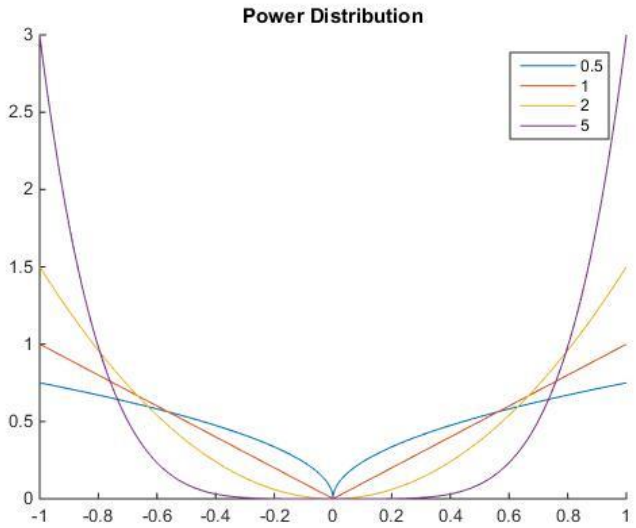


Figure 7. Power Distribution for different q parameter

For benchmarking purposes, a MATLAB function $powrnd()$ is used to generate random number of power distribution within finite interval of $X \in [\mu - \sigma, \mu + \sigma]$. Then by simulating the model at its extreme points through MC, the estimated true bounds are obtained. It is important to note that the estimated true bounds obtained is not the actual true bounds, but it is only an estimation which is close to the actual true bounds. The function makes use of MATLAB built-in function $rand()$ to generate uniformly distributed random number of CDF within interval of [0,1]. The CDF of a general power distribution is given by expression in (15).

$$F(x) = \begin{cases} 0 & x \le -L \\ \frac{1}{2}\left(1 - \frac{x^{q+1}}{L^{q+1}}\right) & -L \le x \le 0 \\ \frac{1}{2}\left(1 + \frac{x^{q+1}}{L^{q+1}}\right) & 0 \le x \le L \\ 1 & otherwise \end{cases} \quad (15)$$

Then the distribution can be reconstructed by determining the inverse of the CDF. The inverse of the CDF is given in (16) (for detailed derivation see Appendix).

$$F^{-1}(x) \equiv f(x) = \begin{cases} (1-2p)^{\frac{1}{q+1}}L & -L \le x \le 0 \\ (2p-1)^{\frac{1}{q+1}}L & 0 \le x \le L \\ 0 & otherwise \end{cases}, where\ F(x) = p \in [0,1] \quad (16)$$

Assuming very large q parameter, the bounds obtained will eventually converge towards a value, in which is assumed to be the estimated true bounds. However, this is not always true; there are cases in which the true bounds are not caused by the extreme points. To show this, consider a simple polynomial of $f(x) = x - x^2, where\ x \in [-1,5]$. By simulating $f(x)$ at the extreme points of x, the bound obtained will be $[-20, -2]$ while the actual bound will be $[-20, 0.25]$. This is because the upper bound is not caused by the extreme points, instead, caused by the $x = 0.5$, which is located around the mean. Thus, in order to obtain the estimated true bounds, the model is simulated for different q parameters through MC to obtain the largest possible bounds for the model.

3.2. Moment Computation

Given the definition of moment of order r[th] can computed through $E[y^r] = \int_{-L}^{L} x^r f_y(x)dx$, whereby L is finite range of the model which is given by $\frac{b-a}{2}$ and $f(x)$ represents the PDF of y [22], the raw moment m_r can be defined as follow:

$$m_r = L^r \psi(r) \quad (17)$$

where m_r is the raw moment, L^r is the scale contributor, and $\psi(r)$ is the shape contributor. The scale contributor L^r component scales the range geometrically with respect to the order of the moment. The scale contributor L^r is independent of the shape contributor $\psi(r)$. $\psi(r)$ can be decomposed into $\psi(r) = \psi^+(r) + (-1)^r\psi^-(r)$ to observed its contribution on positive axis and negative components. Since this project assumes symmetrical distribution, $\psi^+(r) = \psi^-(r)$, thus the shape contributor can be written as

$$\psi(r) = \begin{cases} 2\psi^{+}(r) & r \in 2\mathbb{N} \\ 0 & r \in 2\mathbb{N}+1 \end{cases} \quad (18)$$

In this project, with the assumption that X are uniformly distributed random variables in the finite interval $X \in [\mu - \sigma, \mu + \sigma]$, the shape contributor $\psi(r)$, can be expressed as

$$\psi(r) = \begin{cases} \dfrac{1}{r+1} & r \in 2\mathbb{N} \\ 0 & r \in 2\mathbb{N}+1 \end{cases} \quad (19)$$

Similarly, the shape contributor of random variables X in power distribution with finite interval $X \in [\mu - \sigma, \mu + \sigma]$ can be expressed as

$$\psi(r) = \begin{cases} \dfrac{q+1}{r+q+1} & r \in 2\mathbb{N} \\ 0 & r \in 2\mathbb{N}+1 \end{cases} \quad (20)$$

3.3. Maximum Entropy Distribution Fitting

Maximum entropy principle is a versatile stochastic tool for characterizing *probability density function* (pdf) efficiently with least-biased estimation compared to other distribution fitting methods. In general, maximum entropy principle defines that the pdf that maximizes the Shannon information entropy provides the most information among all other possible pdfs that satisfy the known constraints, thus enabling least-biased estimation to be made [23]. The Shannon entropy is the expected information quantity, which is defined as follow:

$$H(f) = -\int_{x_{min}}^{x_{max}} \ln\big(f(x)\big) f(x) dx \quad (21)$$

Where $ln(f(x))$ is the quantity of the information and f is the pdf. Maximum entropy fully utilizes the known constraints while carefully avoids the unknown parameters. Given the statistical moments of arbitrary basis functions $(h_i(x); i = 0, \dots m)$, the maximum entropy can be expressed as an optimization problem in (22).

$$\int_{x_{min}}^{x_{max}} h_i(x) f(x) dx = \mu_i, i = 0, \dots, m \quad (22)$$

The expression in (22) can be solved by using the method of Lagrange multipliers to obtain the maximum point for the known constraints.

$$L(f, \lambda) = H(f) - \sum_{i=0}^{m} \lambda_i \left(\int_{x_{min}}^{x_{max}} h_i(x) f(x) dx - \mu_i \right)$$

$$\frac{\partial(f,\lambda)}{\partial f} = \int_{x_{min}}^{x_{max}} \left(-1 - \ln\big(f(x)\big) - \sum_{i=0}^{m} \lambda_i h_i(x) \right) dx = 0 \qquad (23)$$

Finally, the analytical form of the maximum entropy pdf can be written as follow:

$$f(x) = \exp\left(-1 - \sum_{j=0}^{m} \lambda_i h_i(x) \right) \qquad (24)$$

3.4. Summary

The probabilistic bounding through moment technique provides another alternative to other existing methods in estimating bound such as Monte Carlo simulation. While MC method is able the estimate optimal bounds, it requires large searching space. This makes it nearly impractical to estimate the bounds especially on hardware with limited performance. The proposed method makes use of the idea of reconstructing the distribution through known finite number of moments and estimate the bound through probability. Maximum entropy method is used as it is able to reconstruct the distribution with least-biased approximation. To demonstrate on how the proposed method performed in comparison with other existing method, a comparative study is performed in the following chapter.

Chapter 4

Results and Discussions

This chapter reports the methodology used to estimate the error bound, case studies on different type of problems, and the summary of the performance of the proposed methods compared with existing methods. Summary of the open problems are also provided at the end of the chapter.

4.1. Testing Methodology

The initial aim of the result is to understand how the proposed method performed compared with existing methods. This done by comparing the error bound obtained through the proposed method with the estimated true bounds (difference in error obtained by distance (a) in Figure 8). The estimated true bound is obtained by simulating the case at the extreme points to obtain the largest bound possible. Then the error bound obtained through GHR, IA and AA are also compared with the estimated true bound where the relative error of GHR, IA and AA is represented by distance (c) and (d) respectively. Lastly, the MC simulation is also compared with the estimated true bounds to see how accurate is MC simulation in determining the bound close to the estimated true bound, which is given by distance (b).

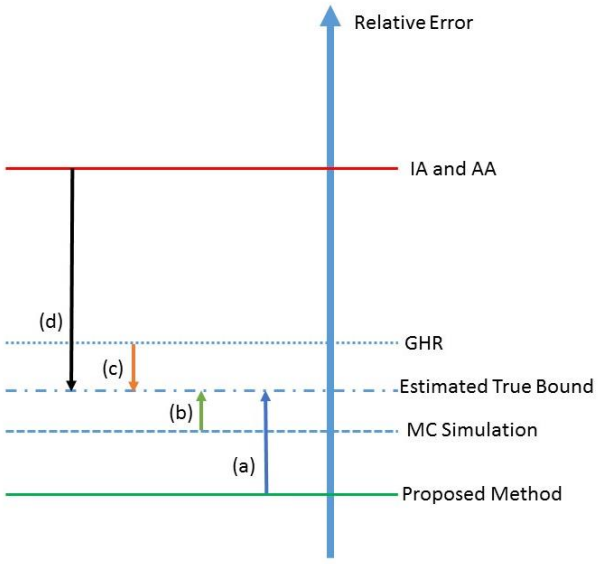


Figure 8. Hierarchy of the error bound

4.2. Comparative Studies

In order to characterise the proposed method in this project, there are four different model to be tested for the comparative studies, which are simple polynomial of $f(x,y) = x^2y - xy^2$, Newton's Method, Toeplitz matrix determinant, and 2-dimensional discrete cosine transform matrix multiplications. The study on the visibility of the method is done to understand the strength and weakness of every method, which is highlighted in Chapter 2. By performing this comparative study, it is hoped to find the examples that are able to highlight the strength and weaknesses of the proposed method. The studies are performed on Windows 7 Enterprise 64-bit PC with Intel® Xeon® CPU E5-1603 v3 @ 2.80GHz (4 cores), ~ 2.7GHz and 16384MB RAM. The comparative studies use MATLAB platform and Wolfram Mathematica platform for moment computation. INTLAB [24] has been used to determine the bounds for IA and AA methods.

4.2.1. Polynomial of $f(x,y) = x^2y - xy^2$

The proposed method is tested on a simple polynomial of $f(x,y) = x^2y - xy^2$. This model is chosen as it exhibits strong correlation in each variables. An error term δ due to the floating point error is introduced in every computational paths. Both x and y are the input distributions, in which the distribution is varied accordingly, with the parameter as follow:

$$x \in [-1,5] \qquad y \in [-2,8]$$
$$\delta \in \left[-2^{-mantissa}, 2^{-mantissa}\right] \qquad , where\ mantissa = 8\ bits$$

The computation can be divided into four computational stages as shown in Table 2.

Table 2. Computational stage of polynomial $f(x,y) = x^2y - xy^2$

Pseudo-code	Floating Point Model
$a = xy$	$a = xy(1+\delta_1)$
$b = ax$	$b = x^2y(1+\delta_1)(1+\delta_2)$
$b = ay$	$c = xy^2(1+\delta_1)(1+\delta_3)$
$d = b - c$	$d = (b-c)(1+\delta_4)$

From Table 2, the floating point model can be expressed as follow:

$$f(\delta) = (x^2y - xy^2 + (x^2y - xy^2)\delta_1 + x^2y\delta_2 + x^2y\delta_1\delta_2 - xy^2\delta_3 - xy^2\delta_1\delta_3)(1+\delta_4) \quad \textbf{(25)}$$

In order to obtain the floating point error model, the mean is subtracted from the floating point model obtained, eliminating any variables without error terms.

$$f(\delta) = \left((x^2y - xy^2)\delta_1 + x^2y\delta_2 + x^2y\delta_1\delta_2 - xy^2\delta_3 - xy^2\delta_1\delta_3\right)(1+\delta_4) + (x^2y - xy^2)\,\delta_4 \quad (26)$$

The result obtained is summarized in Table 3 below.

Table 3. Bound Comparison for $\boldsymbol{f(x,y) = x^2y - xy^2}$

Method	**Lower Bound**	**Upper Bound**	**Computation Time**
IA	-638.3529	637.1059	0.409374s
AA	-502.1051	502.1051	0.553463s
MC with 10^7 iterations	-2.5944	2.5801	25.506904s
MC with 10^9 iterations	-2.8147	2.7910	2509.275287s
GHR	-2.9864	2.9511	NA
Moment	-2.9871	2.9682	106.173211s
SMT	-2.9865	2.9511	NA
*Estimated True Bounds**	-2.9865	2.9511	NA

The proposed method managed to obtain bounds which are close to the estimated actual bounds. The proposed method performed better than IA and AA in term of accuracy and better than MC in term of efficiency. The IA method resulted in overestimation by large margin despite being the most efficient method due to the correlation between operands which are lost for each intervals computation. Similarly, AA method also resulted in overestimation in the derived bounds. This is because multiplication operations are not affine, thus further assumption is required to be made and these approximations may cause overestimation of the derived bounds.

MC method is simulated by using standard normal distribution to model the floating point error. The result obtained (see Table 3) shows that the bounds obtained is always resulted in underestimation due to the insufficient searching space. By increasing the number of iterations, the bound through MC will converge toward the actual true bounds, however this requires high performance computer which is a trade-off between accuracy and efficiency. On the other hand, *Handelman Representation* (GHR) and SMT managed to obtain the tightest bounds compared with all existing methods for simple computation.

The proposed method estimates tight bounds by varying the input distribution, probability level and the order of moments. Large q parameter allows the estimated shape contributor $\psi(r)$to converge. Higher order moments provide more information regarding the "tail" of the distribution, allowing the least-biased approximation of the distribution to be

reconstructed through distribution fitting. This can be seen in Figure 9 where the bound for higher order moment with large q parameter is converging towards the estimated true bounds. While bound estimation by using higher order moments can be applied for small scale problem, it is impractical for large scale problem as the complexity of the moment computations grows exponentially. Thus, in this model, the proposed method considers estimating the bound obtained by choosing appropriate probability level. From Figure 9, the proposed method obtained tighter bound given probability level of $1e-5$ with q parameter of 24. This shows that by choosing appropriate q parameter, it is possible to estimate the appropriate shape contributor $\psi(r)$, thus mitigating the need of higher order moments. Accurate estimation $\psi(r)$ is the key to tight bound estimation.

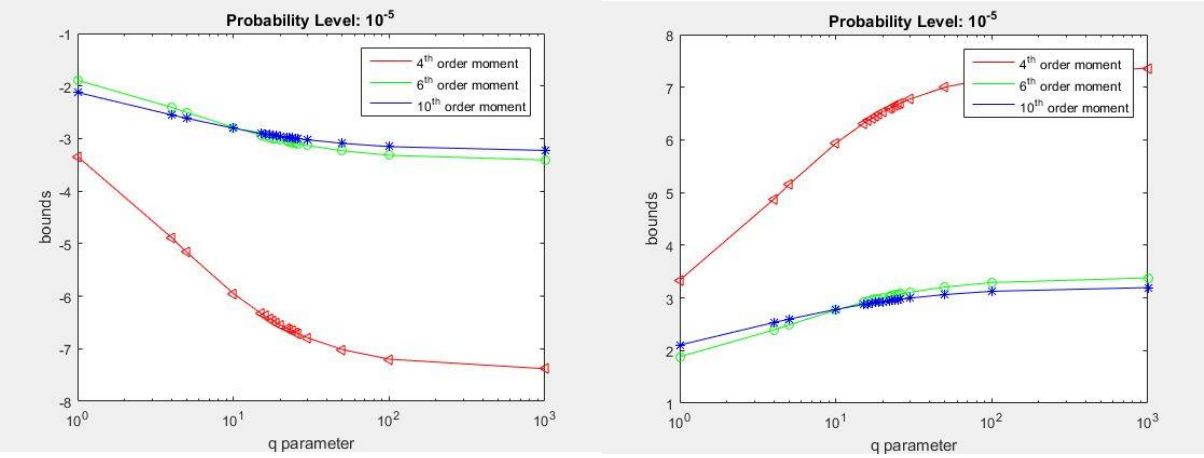


Figure 9. Convergence of the estimated bounds towards the actual true bounds for (a) Lower bound (b) Upper bound

In practice, the error bound of $[-2.9865, 2.9511]$ is too large to be useful. It shows that 8 bits precision is not enough for actual computation of this model as the actual value has an error which is propagating within wide interval. In order to have better results for actual computation, more precision bits may be considered. By using more precision in the computation, the error bound will decreases exponentially (see Figure 10).

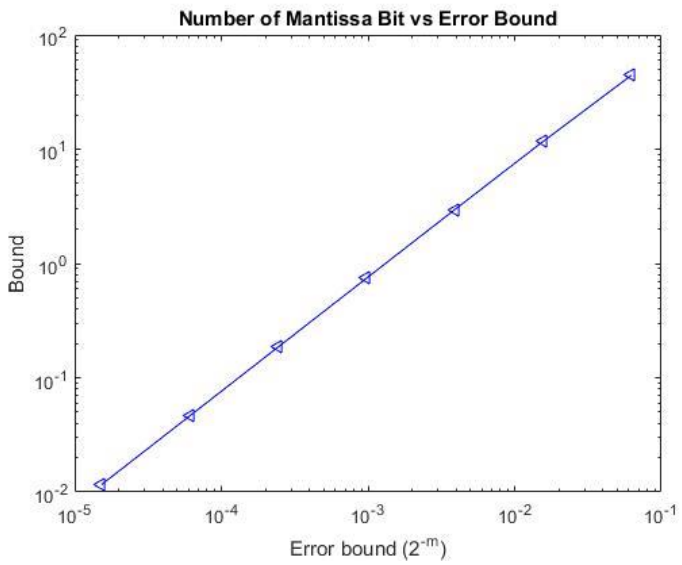


Figure 10. Effect of the mantissa bit used on the error bound

4.2.2. Division

In order to deal with rational function such as $\frac{f(x)}{g(x)}$, the numerator and denominator are assumed to be independent of each other. Then, the bound of numerator and denominator are to be obtained separately, which is then to be combined by inversing the bound of the denominator, which is similar to how IA handle rational function.

$$[n_l, n_u] \times [d_l, d_u]^{-1} \equiv [n_l, n_u] \times \left[\frac{1}{d_l}, \frac{1}{d_u}\right] = \left[\min\left(\frac{n_l}{d_u}, \frac{n_u}{d_l}, \frac{n_u}{d_u}, \frac{n_l}{d_l}\right), \max\left(\frac{n_l}{d_u}, \frac{n_u}{d_l}, \frac{n_u}{d_u}, \frac{n_l}{d_l}\right)\right] \quad (27)$$

In this section, the comparative studies will be performed on Newton's Method. Newton's Method, also known as Newton-Raphson method, is one of the mostly used numerical analysis method to find better approximations to the roots. A general Newton's method is given as

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \equiv \frac{x_n f'(x_n) - f(x_n)}{f'(x_n)} \quad (28)$$

The Newton's method is chosen as one of the case studies due to the strong dependencies for each terms in the numerator and denominator. Consider a simple polynomial model of $f(x) = x^3 + x^2 - 2x$ where the derivative is given as $f'(x) = 3x^2 + 2x - 2$ and $x \in [0.7, 1.5]$. Unlike polynomial model in previous section, the bound estimation of rational function is to be performed separately, given as follows:

$numerator$:

$$z1 = ((x^2(1+\delta_1) + x^3(1+\delta_1)(1+\delta_2))(1+\delta_3) - 2x(1+\delta_4))(1+\delta_5) \quad (29)$$

$denominator$:

$$z2 = (-2 + (3x^2(1+\delta_1)(1+\delta_2) + 2x(1+\delta_3))(1+\delta_4))(1+\delta_5) \quad (30)$$

$division$:

$$z3 = \frac{z1}{z2}(1+\delta) \quad (31)$$

$floating\ point\ model$:

$$z4 = (x - z3)(1+\delta) \quad (32)$$

In order to obtain the floating point error model, the mean is subtracted from the overall floating point model, results in the eliminations of the components which is unaffected by the error terms.

The results obtained is summarized in Table 4 below. Since the computation is performed separately, the detailed results for each computational stage is given in Appendix 5.

Table 4. Bound comparison for Newton's Method model

Method	**Lower Bound**	**Upper Bound**	**Computation Time**
IA	-8.2234	8.1844	0.559673s
AA	-4.4026	4.3902	0.882249s
MC with 10^7 *iterations*	-0.0147	0.0155	40.624649s
MC with 10^9 *iterations*	-0.0148	0.0159	372.238360s
GHR	-3.6904	1.0221	NA
Moment	-3.9637	1.3205	179.337622s
SMT	-0.0491	0.0527	NA
*Estimated True Bounds**	-0.0155	0.0165	NA

The result obtained showed that IA heavily overestimates the bounds despite perform faster compared with other methods. This is because in a model such as Newton's method, there are strong correlations between numerator and denominator. Since IA method lost the correlations between operands for every computational stage performed, it results in bounds which is too large to be useful. AA method managed to mitigate the dependencies problem of IA and obtain better bounds, however it is still overestimate the bounds by large margin. This is because operations, such as division and multiplications are not affine, and approximation to certain extent has to be made in order to obtain better bounds than IA.

MC simulation managed to preserve the dependencies between the numerator and denominator. As a result, it managed to obtain very tight bounds which is close to the actual true bounds. However, it requires large resources in estimating bounds that close to the actual bounds, which is a fair trade-off between computation time and accuracy. GHR method, while it is able to estimate tight bound for the numerator and denominator, it lost the dependencies during division operation, resulting in overestimation of the actual bounds. Similarly, SMT also managed to preserve the dependencies between numerator and denominator. However, the error bounds obtained is overestimated due to the Boolean constraint set. The proposed method also managed to estimate better bounds compared to IA and AA for the numerator and denominator, but the correlations between numerator and denominator is lost in division operation, resulting in overestimation of the actual bounds.

In order to resolve the dependency problem, one may consider of Taylor series approximation to represent the polynomial model. A general truncated series can be written as follow:

$$T(x) = f(\mu) + f'(\mu)(x-\mu) + \frac{f''(\mu)}{2!}(x-\mu)^2 + \cdots + \frac{f^n(\mu)}{n!}(x-\mu)^n \qquad (33)$$

Where the number of truncation depends on the required accuracy and the mean value of the x. The proposed method managed to estimate better bound through Taylor series approximation. Through Taylor series, the proposed method estimates a bound of $[-0.0164, 0.0162]$, which is close to the estimated true bounds. Whilst Taylor series is able to resolve the dependency problem, it is limited to small scale problem. In problem concerning large multivariate polynomial, the numbers of monomials grow too large to compute as the polynomial is raised to higher power.

Numerous researches performed on numerical methods assume independencies for division operation. In real-world applications, however, there bounds to be some dependencies between numerator and denominator. Overlooking this dependency problem will resulted in overestimation of the bounds as happen most of the time for IA method. While division operation rarely occurs in signal processing, it does appear in scientific computation frequently.

4.2.3. Determinant of a Toeplitz Matrix

The determinant of a Toeplitz matrix is chosen to highlight on how the proposed method perform under situation where the same variable is used multiple times due to

symmetry. In this model, floating point model for determinant of Toeplitz matrix is to be constructed recursively by using Laplace theorem for the determinant.

Theorem (Laplace's Formula for the Determinant). *Suppose matrix* $M \in R^{n\times n}$. *For every* $i, j \in \{1,2,\dots,n\}$, *the matrix* $M[i,j]$ *is defined to be* $(n-1)\times(n-1)$ *submatrix of* M *obtained by deleting the* i*th row and the* j*th column. Then for each* $i_0, j_0 \in \{1,2,\dots,n\}$,

$$\mathrm{Det}(M) = \sum_{i=1}^{n} a_{i,j_0}(-1)^{i+j_0}|M[i,j_0]| = \sum_{j=1}^{n} a_{i_0,j}(-1)^{j+i_0}|M[i_0,j]| \qquad (34)$$

Error term δ is introduced at every computational path as *independent identically distributed* (i.i.d.) random variables and is highly correlated.

The model considers a symmetrical 3×3 Toeplitz matrix M for case study between the proposed method and existing methods as given in (35) (for detailed floating point model see Appendix). Each element of the matrix is a constant represented in floating point format, which is bounded by 2^{-m}, where in this case m is assumed to be 8 bits.

$$M = \begin{bmatrix} a & b & c \\ b & a & b \\ c & b & a \end{bmatrix}, where\ a = 2, b = 3, and\ c = 4$$

$$\begin{aligned} &\det(M) \\ &= \begin{vmatrix} [2+2^{-m}2, 2-2^{-m}2] & [3+2^{-m}3, 3-2^{-m}3] & [4+2^{-m}4, 4-2^{-m}4] \\ [3+2^{-m}3, 3-2^{-m}3] & [2+2^{-m}2, 2-2^{-m}2] & [3+2^{-m}3, 3-2^{-m}3] \\ [4+2^{-m}4, 4-2^{-m}4] & [3+2^{-m}3, 3-2^{-m}3] & [2+2^{-m}2, 2-2^{-m}2] \end{vmatrix} \end{aligned} \qquad (35)$$

The floating point error model can be obtained by subtracting the components without error term from the floating point model in Appendix. The bound comparison with other existing methods is summarized in Table 5 below.

Table 5. Bound comparison for determinant of Toeplitz matrix model

Method	**Lower Bound**	**Upper Bound**	**Computation Time**
IA	-3.8108	3.8145	2.844758s
AA	-1.2024	1.2024	3.223793s
MC with 10^7 *iterations*	-0.7158	0.7624	62.329142s
MC with 10^9 *iterations*	-0.7628	0.7969	6304.400102s
GHR	-0.7629	0.7800	NA
Moment	-0.9060	0.9168	958.158997s

SMT[1]	-	-	NA
*Estimated True Bounds**	-0.9020	0.9191	NA

From the result obtained in Table 5, IA method results in overestimation of the bounds despite being the most efficient method. This is caused by the correlations between operands which is lost during each computation. Similarly, AA method also results in overestimation of the bounds. While AA methods is able to preserve the correlations which is lost in IA, general operation, such as multiplication, is not affine. Thus some assumption requires to be made in order to obtain better bounds. MC simulations, underestimate the bounds even after 10^9 iterations. This shows that larger searching space is required in order to be able to estimates better bounds using MC, which is a fair trade-off between efficiency and accuracy. Similarly, GHR method assumes some simplifications in order to reduce the work scale of the problems. SMT is not performed in this model due to the scalability of the problem, which requires more complex Boolean constraint in order to estimate the bounds within the given timeout.

The moment method obtained tighter bounds compared to other methods. The tight bound is estimated by choosing appropriate probability level and order of moment. In this model, 6^{th} order moment and probability level of $1e-5$ are chosen as it provides tighter bounds. Unlike previous models, the resulted distribution does not vary much for higher order moments. Instead, the bounds obtained is depend on the sampled $x-axis$ points used in distribution fitting and the probability level. Figure 11 shows that different order of moment will have different probability level in which the bound will reach asymptotic state despite the resulted distribution to have insignificant change for larger order moment. Another factor which affect the bounds obtained is the q parameter of the power distribution, which is used to fine tune the estimated bounds.

[1] The floating point error model of the determinant of Toeplitz matrix is a large scale problem which may be too complex to perform in SMT

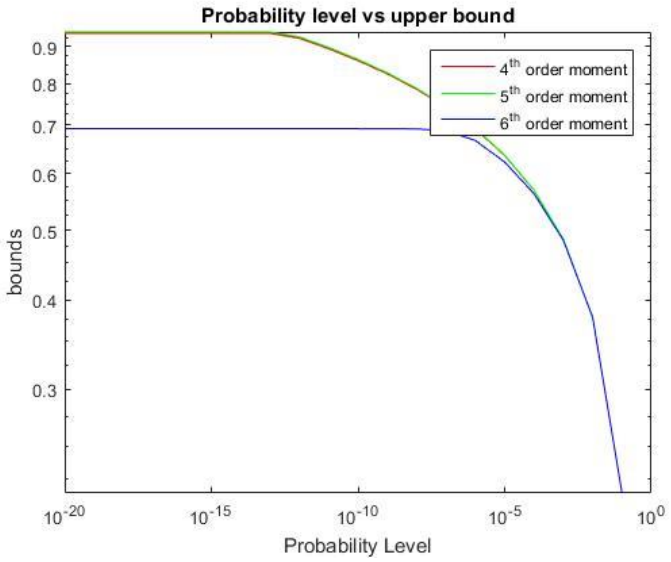


Figure 11. Probability Level used vs the order of moment

Another finding is that the resulted distributed is converging towards standard Gaussian distribution. This finding also applies for different q parameter of the power distribution. This shows that for large chained computation, the *Central Limit Theorem* (CLT) can be applied.

Theorem (Central Limit Theorem). Let $X_1, X_2, \dots, X_n$ be a sequence of independent random variables and each X_i have an arbitrary pdf $f_x(x_1, x_2, \dots, x_n)$ with finite mean μ_i and variance σ_i, under additional conditions on the distribution of large addend $(n \to \infty)$, the distribution will converge to normal distribution with mean μ_N and variance σ_N^2 can be defined such that

$$\mu_N = \sum_{i=1}^{n} \mu_i \ and \ \sigma_N^2 = \sum_{i=1}^{n} \sigma_i^2 \qquad (36)$$

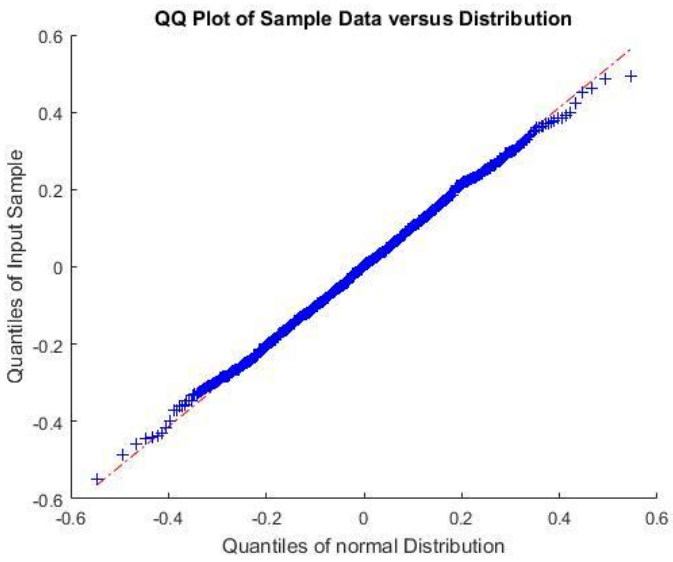


Figure 12. QQ plot of sample data through MC with distribution obtain from proposed method

From Figure 12, it can be observed that the sample data is scattered randomly along the distribution line. Thus, it can be concluded that the distribution obtained satisfies the CLT. A *Kolmogorov-Smirnov Test* (KS test) was performed to the sample data obtained, which shows that the distribution is a Gaussian given for any probability level larger than $2.4273e-26$. This shows that the assumption of CLT holds true as the Gaussian model computed through CLT fits the set of observations from MC simulation.

4.2.4. Matrix Multiplication

In this section, studies are performed for a simple matrix multiplication with dependency assumption between operands and model with independent elements. Proposed method will assume higher order moment for matrix multiplication with dependency assumption while for model with independent elements, the proposed method showcases the application of CLT on the matrix model.

4.2.4.1. Matrix Multiplication Model with Dependency

This model demonstrates matrix multiplication with dependency assumption between each element of the matrix. The model consists of a 3×3 Sobel filter which is commonly used for edge detection in image processing. The matrix A is multiplied with its transpose which is given as follow:

$$A = \begin{bmatrix} a & b & -a \\ c & c & c \\ a & b & -a \end{bmatrix}, where\ B = AA^T \qquad (37)$$

Where $a \in [-1,2], b \in [-2,3], and\ c \in [-2^{-m}, 2^{-m}]$. The bound comparison with other existing methods is summarized in the Table 6 below (for detailed bound comparison see Appendix 7.1.).

Table 6. Bound Comparison for matrix multiplication with dependency assumption

Method[2]	**Lower Bound**	**Upper Bound**	**Computation Time**
IA	-10.0391	17.0664	0.467993s
AA	-6.0352	6.0352	0.838266s
MC with 10^8 *iterations*	0	17.0009	1197.093721s
GHR	-	-	NA
Moment	-0.3822	16.8575	156.032218s
SMT	-	-	NA
*Estimated True Bounds**	0	17.0294	NA

The results obtained in Table 6 shows that the dependency in the model has caused IA to overestimate the lower bound obtained by large margin. This is because IA assumes independency between each interval, thus the bound is assumed to be caused by worst case scenario, which may not be the case. Similarly, AA also resulted in overestimation of the bound as multiplication operation is not affine. In order to resolve this, approximation to some degree has to be made. One may consider Chebyshev approximation to estimate better bound through AA. MC managed to obtain tighter bound, however, the amount of computation time used is inefficient.

The moment method obtained better estimation of the bounds compared to IA and AA. In this model, the bound is estimated by using 10^{th} order moments with probability level of $1e-4$ and q parameter of 1. The q parameter of 1 was chosen as it provided the appropriate shape contributor $\psi(r)$ in order to estimate tight bound. Whilst higher order moment is applicable for small scale problem, it is impractical for large scale problem. In large scale problem, the dependency between random variables will weaken, thus allowing the possibility of CLT to be applied. The application of CLT is provided in the next section.

[2] GHR and SMT are not evaluated due to the complexity of the problem which require specific toolboxes.

4.2.4.2. Matrix Multiplication with Independent Element

This model demonstrates on how the Central Limit Theorem can be applied to the precision analysis for large matrix multiplication because the moments can be found. Large matrix multiplication is the foundation of many highly efficient numerical methods. One of the particular large matrix multiplication is *Discrete Cosine Transform* (DCT), which is used in numerous applications in engineering and science. This section will study on the *Discrete Cosine Transform* (DCT) of a vector of sampled triangle wave and 2-Dimensional *Discrete Cosine Transform* (DCT) of a matrix of input signal.

4.2.4.2.1. Discrete Cosine Transform of a Vector

A discrete cosine transform expresses a finite sequence of input data points in terms of cosine function which oscillates at different frequencies. A general DCT of a vector of input signal can be written as

$$B = Dx, where\ B_k = \sum_{n=0}^{N-1} x_n \cos\left[\frac{\pi}{N}\left(n+\frac{1}{2}\right)k\right], k = 0, \dots, N-1 \qquad (38)$$

Where x is the vector of input signal of size $n \times 1$, B is the DCT vector of x, and D is a DCT matrix of size $n \times n$. Consider an example of a 4×4 DCT matrix where each elements of the matrix are a constant which is expressed in floating point format, which is bounded by 2^{-m}, where m is assumed to be 8 bits precision. An input signal is given as $x = \Lambda(\omega t)$ where the frequency is $5Hz$ and sampling time of $0.1s$.

The bound comparison between each methods is summarized in Table 7 (for detailed result see Appendix 7.2.)

Table 7. Bound Comparison for discrete cosine transform of a vector

Method	**Lower Bound**	**Upper Bound**	**Computation Time**
IA	-7.8125e-03	7.8125e-03	0.485071s
AA	-3.0557e-05	3.05573e-05	0.816236s
MC with 10^8 *iterations*	-7.4674e-03	7.4095e-03	1856.884827s
GHR	-	-	NA
Moment	-7.8535e-03	7.8535e-03	5.257635s
SMT	-2.6550e-3	2.6550e-3	NA
*Estimated True Bounds**	-7.8125e-03	7.8125e-03	NA

From the results obtained, IA method is able to obtain the bounds, without overestimating the bounds. This is due to the initial assumption in which all the parameters are introduced as i.i.d. random variables. As all random variables are independent of each other, IA method is able to estimate the bounds without overestimation. AA method, however, heavily underestimates the bounds. This is because while each computation is independent of each other, AA method assumes dependencies between each computation. This is worsened by the fact that the DCT matrix is cosine wave in which for every cosine constant in the matrix elements, there will be its negative counterparts, which results in cancellation of the terms. This causes AA to fail to form a bound as the bound converges towards the actual values. Another reason is that multiplication operation is not affine. For a large chained computation of non-affine operations, AA method may lead to pessimistic results. MC method results in underestimation of the bounds due to insufficient searching space. This results in MC method to fail to address outlier scenario which gives the estimated true bounds.

The proposed method overestimates the bounds compared to MC simulation and AA method. The distribution obtained is also satisfy the Central Limit Theorem as defined in previous section, which can be seen in Figure 13. where the sample data are scattered randomly along the distribution line. This is also backed by the results of the KS-test where the distribution will remain Gaussian for probability level larger than $1.2404e-80$. The small probability level ensured that the set of observations from MC simulation has an identical distribution with the Gaussian approximation of CLT.

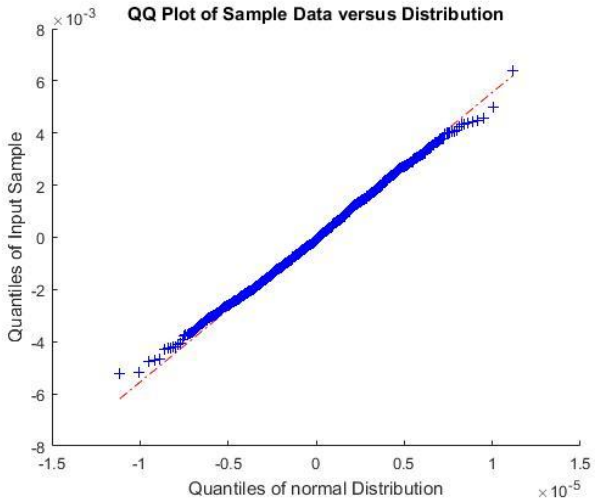


Figure 13. QQ plot of the sample data obtain through MC and distribution obtain through moment method

In terms of efficiency, the IA methods remains the most efficient method in this type of problem as it is able to estimate tightest bounds within very short time period even for a very

large scale matrix multiplication (see Figure 14). Similarly, AA methods also performed efficiently, but fails to estimate the bounds in some cases. MC method is the least efficient method as it requires large resources in order to be able to estimates the actual bounds. The proposed method managed to obtains bounds for smaller computations efficiently. However, as the computation complexity grows, the computation time required is also growing exponentially.

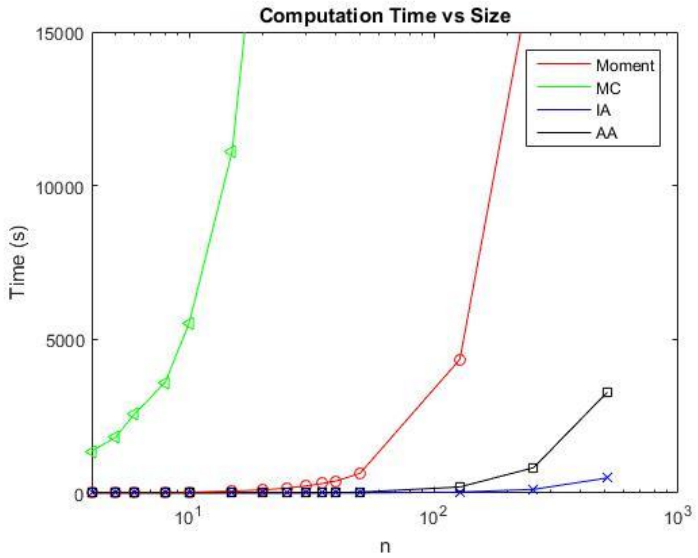


Figure 14. Performance comparison of Moment vs MC simulation

4.2.4.2.2. 2-Dimensional Discrete Cosine Transform

Computation of a 2-dimensional DCT has higher complexity compared to a DCT of a vector. In a 2-dimensional DCT, it is necessary to make uses of similarity transform in order to obtain the resulted DCT matrix. A general similarity transform can be written as

$$B = DAD^{-1} \tag{39}$$

Where A is the 2-dimensional input matrix, D is the DCT matrix, and B is the resulted DCT matrix. As the DCT matrix is orthogonal, the inverse of the matrix is equal to its transpose. Thus, a general 2-dimensional DCT can be expressed as follows:

$$B = DAD^{T} \tag{40}$$

Where each element of the DCT matrix B can be written as

$$B_{pq} = \alpha_p \alpha_q \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} A_{mn} \cos\frac{\pi(2m+1)p}{2M} \cos\frac{\pi(2n+1)q}{2N}, \quad 0 \le p \le M-1, \; 0 \le q \le N-1 \tag{41}$$

$$where\ \alpha_p = \begin{cases} \frac{1}{\sqrt{M}}, p = 0 \\ \sqrt{\frac{2}{M}}, 1 \le p \le M-1 \end{cases} \quad and\ \ \alpha_q = \begin{cases} \frac{1}{\sqrt{N}}, q = 0 \\ \sqrt{\frac{2}{N}}, 1 \le p \le N-1 \end{cases}$$

This section demonstrates on how the proposed method perform in real-world application of image processing (i.e. lossy compression of image (JPEG)). Given A is a normalized image matrix of size 5×5, in which each elements of the matrix are bounded by interval $A_{ij} \in [0,1]$ and D is an DCT matrix of size 5×5 where each elements of the matrix are given as a constant represented in floating point format, which is bounded by 2^{-m}, where m is assumed to be 8 bits precision.

The bound comparison is summarized in Table 8 (for detailed bound comparison for every element of the resulted DCT matrix see Appendix 7.3.)

Table 8. Bound Comparison for 2-dimensional DCT model

Method[3]	**Lower Bound**	**Upper Bound**	**Computation Time**
IA	-1.9093	1.9093	0.736920s
AA	-0.0074	0.0074	2.243810s
MC with 10^8 *iterations*	-1.4019	1.4702	∞
GHR	-	-	NA
Moment	-1.5009	1.5009	664.965196s
SMT	-	-	NA
*Estimated True Bounds**	-1.9093	1.9093	NA

From the result obtained, IA method is able to provide good estimation on the bound. This is due to the assumptions that all parameters are introduced as independent random variables. In independent case, the bound is caused by the worst case scenario, resulting in IA to be able to provide tight bound compared to other method. AA method underestimates the bound heavily. This is because AA assumes dependency between operands, resulting in cancellation of the terms and the estimated bound to actually converge towards the actual values. MC methods manages to estimate optimum bound while still underestimates the bound due to insufficient searching spaces. Similarly, the proposed method managed to estimate optimum bound even for larger size of matrix i.e. $n = 8$ despite underestimating the bound. The distribution obtained also satisfies the CLT as defined in the previous section. This can be

[3] GHR and SMT are not evaluated in this model due to the complexity which requires specific toolboxes.

seen in Figure 15 where the distribution obtain is randomly scattered on the distribution line. KS-test performed on the sample data also showed that the distribution is Gaussian for probability level larger than $2.2816e-40$.

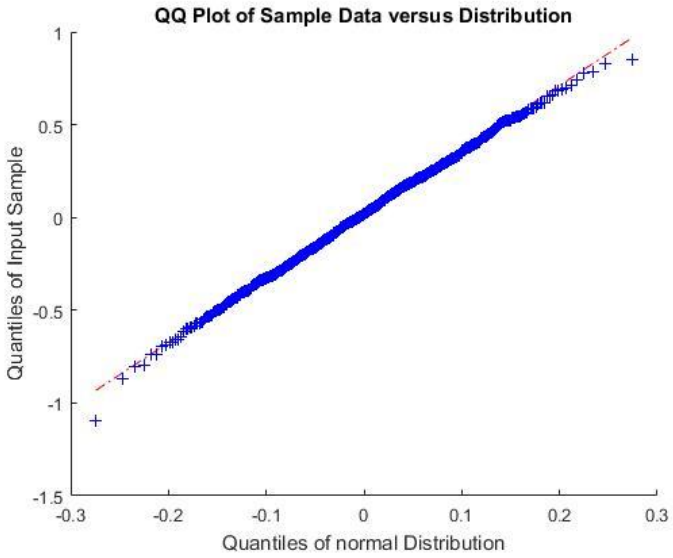


Figure 15.QQ-plot of the sample data through MC with distribution obtained through proposed method

IA method remains the most efficient method in estimating bound where the system considers independent random variables such as matrix multiplication. In such a system, the bound is known to be caused by worst case scenario, enabling IA method to provided best estimation of bound with less cost. However, for large matrix multiplication where the system is a nonlinear, this may not hold. One may consider an example of algebraic Riccati equation. In general, a *continuous time algebraic Riccati equation* (CARE) can be defined as

$$A^T X + XA - XBR^{-1}B^T X + Q = 0 \qquad (42)$$

In algebraic Riccati equation, there are dependency strong dependency among the operands. This will cause IA method to overestimate the bound as the correlations is lost in between the intervals as shown in Section 4.2.4.1.

4.3. Open Problems

The proposed method showed some potentials compared with current existing methods. However, there are still some open problems which may be the focus for future research. Firstly, due to the nature, moment method often leads to underestimation of the bounds, especially for large scale computation. While this can be resolved by using higher order

moment to obtain better estimation of the shape contributor $\psi(r)$, for large scale chained computation, obtaining higher order moments is nearly impractical. Thus the proposed method introduced the power distribute, in practice, bound deduction is a more complex process than bound refinement. Secondly, when addressing to operation such as division, the proposed methods assumes independencies between the numerator and denominator. In real-world application, this rarely the case as most of the rational function have some correlations between the numerator and denominator. Ignoring these correlations will lead to overestimation of the bound. Lastly, the assumption of CLT is only applicable in the case where there are sufficiently large arithmetic means of independent random variables. For similar case of large arithmetic means of independent random variables, IA method still remains the most efficient method to estimate bound.

4.4. Summary

This chapter provides clear insight on how the proposed method performed compared with other existing methods. The proposed method provides better bound estimation than IA and AA, especially in the cases where strong correlations between variables are considered. When strong correlations are present, IA method will lead to over pessimistic bound. Similarly, for computation of non-affine with strong correlations, AA method will tend to overestimate the bound. The proposed method also performs more efficiently than MC simulation. CLT introduced a unique way in addressing problem for large scale computation for the proposed method. The project is concluded in the following chapter.

Chapter 5

Conclusion and Future Work

5.1. Conclusion

This project has demonstrated that probabilistic bounding through moment technique provides another alternative to any present existing methods in estimating bounds for any values. The proposed method is able to address the dependency problem that lingers in existing method such as IA and AA, especially for large scale computations. Alternatively, the proposed method provides *less-expensive* solution through statistical analysis than MC simulation. MC simulation requires large searching space in order to be able to simulate the outlier case in order to obtain the optimum bound. The proposed method makes use of statistical analysis and algebraic substitution to estimate the finite number of moments that define the overall distribution.

5.2. Future Work

Whilst this research has shown some potentials compared with current existing methods, there are still some open problems which may benefit from further research. These include the open problems as highlighted in Section 4.3, a better algorithm to deduce the probability level that bound the distribution without underestimation, a better method to estimate the q parameter accurately, improving the stability of the distribution when computation involving very small floating point is involved, as well as extend the application of the method to handle more complex computations such as exponential, square roots and trigonometric functions.

Appendix

Appendix 1. Interval Arithmetic

$f(x,y) = x^2y - xy^2$ given $x \in \bar{x} = [-2,6]$ and $y \in \bar{y} = [-3,5]$

In order to calculate the interval of the function $f(x,y)$, the computation in the function can be separated into several parts as shown below.

$$a = x * y = [-2,6] \times [-3,5]$$

$$= [\min(-2 \times -3, 6 \times -3, -2 \times 5, 6 \times 5), \max(-2 \times -3, 6 \times -3, -2 \times 5, 6 \times 5)]$$

$$= [\min(6, -18, -10, 30), \max(6, -18, -10, 30)]$$

$$= [-18, 30]$$

$$b = a * x = [-18,30] \times [-2,6]$$

$$= [\min(-18 \times -2, 30 \times -2, -18 \times 6, 30 \times 6), \max(-18 \times -2, 30 \times -2, -18 \times 6, 30 \times 6)]$$

$$= [\min(36, -60, -108, 180), \max(36, -60, -108, 180)]$$

$$= [-108, 180]$$

$$c = a * y = [-18,30] \times [-3,5]$$

$$= [\min(-18 \times -3, 30 \times -3, -18 \times 5, 30 \times 5), \max(-18 \times -3, 30 \times -3, -18 \times 5, 30 \times 5)]$$

$$= [\min(54, -90, -90, 150), \max(54, -90, -90, 150)]$$

$$= [-90, 150]$$

$$d = b - c = [-108,180] - [-90,150] = [-108 - 150, 180 - (-90)] = [\mathbf{-258}, \mathbf{270}]$$

Appendix 2. Affine Arithmetic

$f(x,y) = x^2y - xy^2$ given $x \in \bar{x} = [-2,6]$ and $y \in \bar{y} = [-3,5]$

As both x and y are given in the interval form, it must be converted into affine form before affine analysis can be performed.

$$\hat{x} = 2 + 4\delta_x$$

$$\hat{y} = 1 + 4\delta_y$$

Where $\delta \in [-1,1]$

Similarly, to compute the bound of the $f(x, y)$, the computation in the function can be separated into several part as shown below.

• $\hat{a} = \hat{x}\hat{y}$

$$\hat{a} \leq (2 + 4\delta_x)(1 + 4\delta_y) + B\big((2 + 4\delta_x)(1 + 4\delta_y)\big)2\Delta\delta_a$$

$$\hat{a} \leq (2 + 4\delta_x)(1 + 4\delta_y) + (1 + 4\delta_y)B(2 + 4\delta_x)2\Delta\varepsilon_x + (2 + 4\delta_x)B(1 + 4\delta_y)2\Delta\delta_y$$

$$+B(2 + 4\delta_x + 8\delta_y)2\Delta\delta_a$$

$$\hat{a} \leq (2 + 4\delta_x)(1 + 4\delta_y) + 6(1 + 4\delta_y)2\Delta\delta_x + 5(2 + 4\delta_x)2\Delta\delta_y + (14)2\Delta\delta_a$$

$$\hat{a} \leq 2 + (4 + 6 \times 2\Delta)\delta_x + (8 + 10 \times 2\Delta)\delta_y + (14)2\Delta\delta_a$$

$$E(\hat{a}) = \hat{a} - \hat{x}\hat{y} \leq 6(1 + 4\delta_y)2\Delta\varepsilon_x + 5(2 + 4\delta_x)2\Delta\delta_y + (14)2\Delta\delta_a$$

note: the second order term $\delta_x\delta_y$ will result in very small number, thus can be neglected

• $\hat{b} = \hat{a}\hat{x}$

$$\hat{b} \leq \big(2 + (4 + 6 \times 2\Delta)\delta_x + (8 + 10 \times 2\Delta)\delta_y + (14)2\Delta\delta_a\big)(2 + 4\delta_x)$$

$$+B\left(\big(2 + (4 + 6 \times 2\Delta)\delta_x + (8 + 10 \times 2\Delta)\delta_y + (14)2\Delta\delta_a\big)(2 + 4\delta_x)\right)2\Delta\delta_b$$

$$\hat{b} \leq \big(2 + (4 + 6 \times 2\Delta)\delta_x + (8 + 10 \times 2\Delta)\delta_y + (14)2\Delta\delta_a\big)(2 + 4\delta_x)$$

$$+\big(2 + (4 + 6 \times 2\Delta)\delta_x + (8 + 10 \times 2\Delta)\delta_y + (14)2\Delta\delta_a\big)B(2 + 4\delta_x)2\Delta\delta_x$$

$$+(2 + 4\delta_x)B\big(2 + (4 + 6 \times 2\Delta)\delta_x + (8 + 10 \times 2\Delta)\delta_y + (14)2\Delta\delta_a\big)\Delta\delta_a$$

$$+B\left(\big(2 + (4 + 6 \times 2\Delta)\delta_x + (8 + 10 \times 2\Delta)\delta_y + (14)2\Delta\delta_a\big)(2 + 4\delta_x)\right)2\Delta\delta_b$$

$$\hat{b} \le \big(2 + (4 + 6 \times 2\Delta)\delta_x + (8 + 10 \times 2\Delta)\delta_y + (14)2\Delta\delta_a\big)(2 + 4\delta_x)$$

$$+6\big(2 + (4 + 6 \times 2\Delta)\delta_x + (8 + 10 \times 2\Delta)\delta_y + (14)2\Delta\delta_a\big)2\Delta\delta_x + (2 + 4\delta_x)B(14 + (30)2\Delta)2\Delta\delta_a$$

$$+B\big(4 + 8\delta_x + 2(4 + 6 \times 2\Delta)\delta_x + (8 + 10 \times 2\Delta)\delta_y + 2(14)2\Delta\delta_a\big)2\Delta\delta_b$$

$$\hat{b} \le 4 + (16 + 24 \times 2\Delta)\delta_x + (8 + 10 \times 2\Delta)\delta_y + (28 + (88)2\Delta)2\Delta\delta_a + (28 + 44 \times 2\Delta)2\Delta\delta_b$$

$$E\big(\hat{b}\big) = \hat{b} - \hat{a}\hat{x} \le (12)2\Delta\delta_x + (28 + (60)2\Delta)2\Delta\delta_a + (28 + 44 \times 2\Delta) \times 2\Delta\delta_b$$

note: the second order terms will result in very small number, thus can be neglected

- $\boldsymbol{\hat{c} = \hat{a}\hat{y}}$

$$\hat{c} \le (2 + (4 + 6 \times 2\Delta)\delta_x + (8 + 10 \times 2\Delta)\delta_y + (14)2\Delta\delta_a)\big(1 + 4\delta_y\big)$$

$$+B(\big(2 + (4 + 6 \times 2\Delta)\delta_x + (8 + 10 \times 2\Delta)\delta_y + (14)2\Delta\delta_a\big)\big(1 + 4\delta_y\big))2\Delta\delta_c$$

$$\hat{c} \le (2 + (4 + 6 \times 2\Delta)\delta_x + (8 + 10 \times 2\Delta)\delta_y + (14)2\Delta\delta_a)\big(1 + 4\delta_y\big)$$

$$+(2 + (4 + 6 \times 2\Delta)\delta_x + (8 + 10 \times 2\Delta)\delta_y + (14)2\Delta\delta_a)B\big(1 + 4\delta_y\big)\Delta\delta_y$$

$$+\big(1 + 4\delta_y\big)B(2 + (4 + 6 \times 2\Delta)\delta_x + (8 + 10 \times 2\Delta)\delta_y + (14)2\Delta\delta_a)\Delta\delta_a$$

$$+B(\big(2 + (4 + 6 \times 2\Delta)\delta_x + (8 + 10 \times 2\Delta)\delta_y + (14)2\Delta\delta_a\big)\big(1 + 4\delta_y\big))2\Delta\delta_c$$

$$\hat{c} \le (2 + (4 + 6 \times 2\Delta)\delta_x + (8 + 10 \times 2\Delta)\delta_y + (14)2\Delta\delta_a)\big(1 + 4\delta_y\big)$$

$$+5(2 + (4 + 6 \times 2\Delta)\delta_x + (8 + 10 \times 2\Delta)\delta_y + (14)2\Delta\varepsilon_a)2\Delta\delta_y + \big(1 + 4\varepsilon_y\big)B(14 + (30)2\Delta)2\Delta\delta_a$$

$$+B(2 + 8\delta_y + (4 + 6 \times 2\Delta)\delta_x + (8 + 10 \times 2\Delta)\delta_y + (14)2\Delta\varepsilon_a)2\Delta\delta_c$$

$$\hat{c} \le 2 + 8\delta_y + (4 + 6 \times 2\Delta)\delta_x + (8 + 10 \times 2\Delta)\delta_y + (14)2\Delta\delta_a + (10)2\Delta\delta_y + (14 + (30)2\Delta)2\Delta\delta_a$$

$$+(22 + (30)2\Delta)2\Delta\delta_c$$

$$\hat{c} \le 2 + (4 + 6 \times 2\Delta)\delta_x + (16 + 20 \times 2\Delta)\delta_y + (14 + (44)2\Delta)2\Delta\delta_a + (22 + (30)2\Delta)2\Delta\delta_c$$

$$E(\hat{c}) = \hat{c} - \hat{a}\hat{y} \leq (10)2\Delta\delta_y + (14 + (30)2\Delta)2\Delta\delta_a + (22 + (30)2\Delta)2\Delta\delta_c$$

note: the second order and higher order term terms will result in very small number, thus can be neglected

- $\hat{\boldsymbol{d}} = \hat{\boldsymbol{b}} - \hat{\boldsymbol{c}}$

$$\hat{b} = 4 + (16 + 24 \times 2\Delta)\delta_x + (8 + 10 \times 2\Delta)\delta_y + (28 + (88)2\Delta)2\Delta\delta_a + (28 + 44 \times 2\Delta)2\Delta\delta_b$$

$$-\hat{c} = -(2 + (4 + 6 \times 2\Delta)\delta_x + (16 + 20 \times 2\Delta)\delta_y + (14 + (44)2\Delta)2\Delta\delta_a + (22 + (30)2\Delta)2\Delta\delta_c)$$

$$\begin{aligned}\hat{d} \leq\ & 2 + (12 + 18 \times 2\Delta)\delta_x - (8 + 10 \times 2\Delta)\delta_y + (14 + (44)2\Delta)2\Delta\delta_a \\ & + (28 + 44 \times 2\Delta)2\Delta\delta_b - (22 + (30)2\Delta)2\Delta\delta_c + (84 + 166 \times 2\Delta)2\Delta\delta_b \\ & - (58 + 100 \times 2\Delta)2\Delta\delta_c + B(2 + (12 + 18 \times 2\Delta)\delta_x - (8 + 10 \times 2\Delta)\delta_y \\ & + (14 + (44)2\Delta)2\Delta\delta_a + (28 + 44 \times 2\Delta)2\Delta\delta_b - (22 + (30)2\Delta)2\Delta\delta_c)2\Delta\delta_d\end{aligned}$$

$$\begin{aligned}\hat{d} \leq\ & 2 + (12 + 18 \times 2\Delta)\delta_x - (8 + 10 \times 2\Delta)\delta_y + (14 + (44)2\Delta)2\Delta\delta_a \\ & + (112 + 210 \times 2\Delta)2\Delta\delta_b \\ & -(80 + 130 \times 2\Delta)2\Delta\delta_c + (26 + (66)2\Delta)2\Delta\delta_d\end{aligned}$$

$$\begin{aligned}E(\hat{d}) &= \hat{d} - (\hat{b} - \hat{c}) \\ &\leq (84 + 166 \times 2\Delta)2\Delta\delta_b - (58 + 100 \times 2\Delta)2\Delta\delta_c + (26 + 66 \times 2\Delta)2\Delta\delta_d\end{aligned}$$

Appendix 3. Bound Deduction through Handelman Representation

For function $f(x, y) = x^2y - xy^2$, the floating point model can be expressed as

$$a = xy(1 + \delta_1)$$

$$b = x^2y(1 + \delta_1)(1 + \delta_2)$$

$$c = xy^2(1 + \delta_1)(1 + \delta_3)$$

$$d = (b - c)(1 + \delta_4)$$

$$= (x^2y - xy^2 + (x^2y - xy^2)\delta_1 + x^2y\delta_2 + x^2y\delta_1\delta_2 - xy^2\delta_3 - xy^2\delta_1\delta_3)(1 + \delta_4)$$

As the value for worst case is known to lie at the extremes, δ_4 can be neglected. Therefore, the floating point model is as following.

$$f(\delta) = x^2y - xy^2 + (x^2y - xy^2)\delta_1 + x^2y\delta_2 + x^2y\delta_1\delta_2 - xy^2\delta_3 - xy^2\delta_1\delta_3$$

Lower Bound

The lower bound is equivalent of the form $f(\delta) - \hat{\gamma}_{lower} = p_{lower\_ghr}$, in which we have to determine the best Handelman representation in order to obtain better bound.

- ***For first iteration***,

$$f_1(\delta) = x^2y - xy^2 + (x^2y - xy^2)\delta_1 + x^2y\delta_2 + x^2y\delta_1\delta_2 - xy^2\delta_3 - \boldsymbol{xy^2\delta_1\delta_3}$$

The highest order monomial: $-\boldsymbol{xy^2\delta_1\delta_3}$

Approach

1	**2**	**3**
$\mu = ([\mathbf{1},\mathbf{0},\mathbf{0}],[\mathbf{0},\mathbf{0},\mathbf{1}])$ $\boldsymbol{\alpha} = (\mathbf{1},\mathbf{0},\mathbf{0})$ $\boldsymbol{\beta} = (\mathbf{0},\mathbf{0},\mathbf{1})$ $\boldsymbol{c} = \boldsymbol{xy^2}$	$\mu = ([1,0,0],[0,0,1])$ $\alpha = (0,0,1)$ $\beta = (1,0,0)$ $c = xy^2$	$\mu = ([1,0,1])$ $\alpha = (2)$ $\beta = (0)$ $c = xy^2$
$\boldsymbol{h_1(\delta)}$		
$\boldsymbol{xy^2(\Delta - \delta_1)(\Delta + \delta_3)}$	$xy^2(\Delta + \delta_1)(\Delta - \delta_3)$	$xy^2(\Delta^2 - \delta_1\delta_3)$
Expansion of $\boldsymbol{h_1(\delta)}$		
$\boldsymbol{xy^2\Delta^2 - xy^2\Delta\delta_1}$ $\boldsymbol{+xy^2\Delta\delta_3 - xy^2\delta_1\delta_3}$	$xy^2\Delta^2 + xy^2\Delta\delta_1$ $-xy^2\Delta\delta_3 - xy^2\delta_1\delta_3$	$xy^2\Delta^2 - xy^2\delta_1\delta_3$
$\boldsymbol{f_2(\delta) = f_1(\delta) - h_1(\delta)}$		
$\boldsymbol{xy(x - y - y\Delta^2)}$ $\boldsymbol{+xy(x - y + y\Delta)\delta_1}$ $\boldsymbol{+x^2y\delta_2 + x^2y\delta_1\delta_2}$ $\boldsymbol{+xy(-y - y\Delta)\delta_3}$	$xy(x - y - y\Delta^2)$ $+xy(xy - y - y\Delta)\delta_1$ $+x^2y\delta_2 + x^2y\delta_1\delta_2$ $+xy(-y + y\Delta)\delta_3$	$xy(x - y - y\Delta^2)$ $+ xy(x - y)\delta_1$ $+ x^2y\delta_2 + x^2y\delta_1\delta_2$ $- xy^2\delta_3$

- ***For second iteration,***

$$f_2(\delta) = xy(x - y - y\Delta^2) + xy(x - y - y\Delta)\delta_1 + x^2y\delta_2 + \boldsymbol{x^2y\delta_1\delta_2} + xy(-y + y\Delta)\delta_3$$

The highest order monomial: $\boldsymbol{x^2y\delta_1\delta_2}$

Approach

1	**2**	**3**
$\mu = ([\mathbf{1},\mathbf{0},\mathbf{0}],[\mathbf{0},\mathbf{0},\mathbf{1}])$ $\boldsymbol{\alpha} = (\mathbf{0},\mathbf{0},\mathbf{0})$ $\boldsymbol{\beta} = (\mathbf{1},\mathbf{1},\mathbf{0})$ $\boldsymbol{c} = \boldsymbol{x^2y}$	$\mu = ([1,0,0],[0,0,1])$ $\alpha = (1,1,0)$ $\beta = (0,0,0)$ $c = x^2y$	$\mu = ([1,0,1])$ $\alpha = (0)$ $\beta = (2)$ $c = x^2y$
$\boldsymbol{h_2(\delta)}$		
$\boldsymbol{x^2y(\Delta + \delta_1)(\Delta + \delta_2)}$	$x^2y(\Delta - \delta_1)(\Delta - \delta_2)$	$x^2y(\Delta^2 + \delta_1\delta_2)$
Expansion of $\boldsymbol{h_2(\delta)}$		
$\boldsymbol{x^2y\Delta^2 + x^2y\Delta\delta_1}$ $\boldsymbol{+x^2y\Delta\delta_2 + x^2y\delta_1\delta_2}$	$x^2y\Delta^2 - x^2y\Delta\delta_1$ $-x^2y\Delta\delta_2 + x^2y\delta_1\delta_2$	$x^2y\Delta^2 + x^2y\delta_1\delta_2$
$\boldsymbol{f_3(\delta) = f_2(\delta) - h_2(\delta)}$		
$\boldsymbol{xy(x - y - x\Delta^2 - y\Delta^2)}$ $\boldsymbol{+xy(x - y - x\Delta}$ $\boldsymbol{- y\Delta)\delta_1}$ $\boldsymbol{+xy(x - x\Delta)\delta_2}$ $\boldsymbol{+xy(-y + y\Delta)\delta_3}$	$xy(x - y - x\Delta^2 - y\Delta^2)$ $+xy(x - y + x\Delta$ $- y\Delta)\delta_1$ $+xy(x + x\Delta)\delta_2$ $+xy(-y + y\Delta)\delta_3$	$xy(x - y - x\Delta^2$ $- y\Delta^2) + xy(x - y$ $- y\Delta)\delta_1 + x^2y\delta_2$ $+ xy(-y + y\Delta)\delta_3$

$$f_3(\delta) = xy(x - y - x\Delta^2 - y\Delta^2) + xy(x - y - x\Delta - y\Delta)\delta_1 + xy(x - x\Delta)\delta_2 + xy(-y + y\Delta)\delta_3$$

As the rest of the monomials are first order monomials, there is only one way to cancel it.

$xy(x - y - x\Delta - y\Delta)\delta_1 \rightarrow xy(x - y - x\Delta - y\Delta)(\Delta + \delta_1)$

$xy(x - x\Delta)\delta_2 \rightarrow xy(x - x\Delta)(\Delta + \delta_2)$

$xy(-y + y\Delta)\delta_3 \rightarrow xy(-y + y\Delta)(\Delta + \delta_3)$

By applying the $f(\delta) - \hat{\gamma}_{lower} = p_{lower\_ghr}$, where the p_{lower_ghr} can be expressed as

$$p_{lower\_ghr} = xy(x - y - x\Delta - y\Delta)(\Delta + \delta_1) + xy(x - x\Delta)(\Delta + \delta_2) + x^2y(\Delta + \delta_1)(\Delta + \delta_2) + xy^2(\Delta + \delta_1)(\Delta - \delta_3) + xy(-y + y\Delta)(\Delta + \delta_3)$$

Hence, the lower bound $\hat{\gamma}_{lower}$ can be obtained as following.

$$f(\delta) - p_{lower_{ghr}} = \hat{\gamma}_{lower}$$

$$\hat{\gamma}_{lower} = x^2y - xy^2 - 2x^2y\Delta + 2xy^2\Delta + x^2y\Delta^2 - xy^2\Delta^2$$

Upper Bound

The upper bound is equivalent of the form $\hat{\gamma}_{upper} - f(\delta) = p_{upper\_ghr}$, in which we have to determine the best Handelman representation in order to obtain better bound.

$$g(\delta) = -f(\delta) = -x^2y + xy^2 - (x^2y - xy^2)\delta_1 - x^2y\delta_2 - x^2y\delta_1\delta_2 + xy^2\delta_3 + xy^2\delta_1\delta_3$$

- **For first iteration**,

$$g_1(\delta) = -x^2y + xy^2 - (x^2y - xy^2)\delta_1 - x^2y\delta_2 - x^2y\delta_1\delta_2 + xy^2\delta_3 + \boldsymbol{xy^2\delta_1\delta_3}$$

Highest order monomial: $\boldsymbol{xy^2\delta_1\delta_3}$

Approach

1	**2**	**3**
$\mu = ([\mathbf{1},\mathbf{0},\mathbf{0}],[\mathbf{0},\mathbf{0},\mathbf{1}])$ $\boldsymbol{\alpha} = (\mathbf{0},\mathbf{0},\mathbf{0})$ $\boldsymbol{\beta} = (\mathbf{1},\mathbf{0},\mathbf{1})$ $\boldsymbol{c} = \boldsymbol{xy^2}$	$\mu = ([1,0,0],[0,0,1])$ $\alpha = (1,0,1)$ $\beta = (0,0,0)$ $c = xy^2$	$\mu = ([1,0,1])$ $\alpha = (0)$ $\beta = (2)$ $c = xy^2$
$\boldsymbol{h_1(\delta)}$		
$\boldsymbol{xy^2(\Delta + \delta_1)(\Delta + \delta_3)}$	$xy^2(\Delta - \delta_1)(\Delta - \delta_3)$	$xy^2(\Delta^2 + \delta_1\delta_3)$
Expansion of $\boldsymbol{h_1(\delta)}$		
$\boldsymbol{xy^2\Delta^2 + xy^2\Delta\delta_1}$ $\boldsymbol{+xy^2\Delta\delta_3 + xy^2\delta_1\delta_3}$	$xy^2\Delta^2 - xy^2\Delta\delta_1$ $-xy^2\Delta\delta_3 + xy^2\delta_1\delta_3$	$xy^2\Delta^2 + xy^2\delta_1\delta_3$
$\boldsymbol{g_2(\delta) = g_1(\delta) - h_1(\delta)}$		

$\boldsymbol{xy(-x+y-y\Delta^2)}$ $\boldsymbol{+xy(-x+y-y\Delta)\delta_1}$ $\boldsymbol{-x^2y\delta_2-x^2y\delta_1\delta_2}$ $\boldsymbol{+xy(y-y\Delta)\delta_3}$	$xy(-x+y-y\Delta^2)$ $+xy(-x+y+y\Delta)\delta_1$ $-x^2y\delta_2-x^2y\delta_1\delta_2$ $+xy(y+y\Delta)\delta_3$	$xy(-x+y-y\Delta^2)$ $+xy(-x+y)\delta_1$ $-x^2y\delta_2-x^2y\delta_1\delta_2$ $+xy^2\delta_3$

- **For second iteration,**

$g_2(\delta) = -x^2y + xy^2 - xy^2\Delta^2 + (-x^2y + xy^2 - xy^2\Delta)\delta_1 - x^2y\delta_2 - \boldsymbol{x^2y\delta_1\delta_2} + (xy^2 - xy^2\Delta)\delta_3$

Highest order monomial: $\boldsymbol{-x^2y\delta_1\delta_2}$

Approach

1	**2**	**3**
$\mu = (\boldsymbol{[1,0,0],[0,0,1]})$ $\boldsymbol{\alpha = (1,0,0)}$ $\boldsymbol{\beta = (0,1,0)}$ $\boldsymbol{c = x^2y}$	$\mu = ([1,0,0],[0,0,1])$ $\alpha = (0,1,0)$ $\beta = (1,0,0)$ $c = x^2y$	$\mu = ([1,0,1])$ $\alpha = (2)$ $\beta = (0)$ $c = x^2y$
$\boldsymbol{h_2(\delta)}$		
$\boldsymbol{x^2y(\Delta-\delta_1)(\Delta+\delta_2)}$	$x^2y(\Delta+\delta_1)(\Delta-\delta_2)$	$x^2y(\Delta^2-\delta_1\delta_2)$
Expansion of $\boldsymbol{h_2(\delta)}$		
$\boldsymbol{x^2y\Delta^2-x^2y\Delta\delta_1}$ $\boldsymbol{+x^2y\Delta\delta_2-x^2y\delta_1\delta_2}$	$x^2y\Delta^2+x^2y\Delta\delta_1$ $-x^2y\Delta\delta_2-x^2y\delta_1\delta_2$	$x^2y\Delta^2-x^2y\delta_1\delta_2$
$\boldsymbol{g_2(\delta) = g_2(\delta) - h_2(\delta)}$		
$\boldsymbol{xy(-x+y-x\Delta^2-y\Delta^2)}$ $\boldsymbol{+xy(-x+y+x\Delta-y\Delta)\delta_1}$ $\boldsymbol{+xy(-x-x\Delta)\delta_2}$ $\boldsymbol{+xy(y-y\Delta)\delta_3}$	$xy(-x+y-x\Delta^2-y\Delta^2)$ $+xy(-x+y-x\Delta-y\Delta)\delta_1$ $+xy(-x+x\Delta)\delta_2$ $+xy(y-y\Delta)\delta_3$	$xy(-x+y-x\Delta^2-y\Delta^2)$ $+(-x^2y+xy^2-xy^2\Delta)\delta_1$ $-x^2y\delta_2+xy(y-y\Delta)\delta_3$

$$g_3(\delta) = xy(-x + y - x\Delta^2 - y\Delta^2) + xy(-x + y - x\Delta - y\Delta)\delta_1 + xy(-x + x\Delta)\delta_2 + xy(y - y\Delta)\delta_3$$

As the rest of the monomials are first order monomials, there is only one way to cancel it.

$xy(-x + y - x\Delta - y\Delta)\delta_1 \rightarrow xy(-x + y - x\Delta - y\Delta)(\Delta + \delta_1)$

$xy(-x + x\Delta)\delta_2 \rightarrow xy(-x + x\Delta)(\Delta + \delta_2)$

$xy(y - y\Delta)\delta_3 \rightarrow xy(y - y\Delta)(\Delta + \delta_3)$

By applying the $\hat{\gamma}_{upper} - f(\delta) = p_{upper\_ghr}$, where the p_{lower_ghr} can be expressed as

$$p_{upper\_ghr} = xy(-x + y - x\Delta - y\Delta)(\Delta + \delta_1) + x^2y(\Delta + \delta_1)(\Delta - \delta_2) + xy(-x + x\Delta)(\Delta + \delta_2) + xy(y - y\Delta)(\Delta + \delta_3) + xy^2(\Delta + \delta_1)(\Delta + \delta_3)$$

Hence, the upper bound $\hat{\gamma}_{upper}$ can be obtained as following.

$$\hat{\gamma}_{upper} - f(\delta) = p_{upper\_ghr}$$

$$\hat{\gamma}_{upper} = p_{upper\_ghr} - g(\delta)$$

$$\hat{\gamma}_{upper} = -x^2y + xy^2 + 2x^2y\Delta - 2xy^2\Delta - x^2y\Delta^2 + xy^2\Delta^2$$

Appendix 4. Power Distribution

A general power distribution has the pdf which can be defined as:

$$f(x) = \frac{q+1}{2L}\left|\frac{x}{L}\right|^q, where\ q \geq 0\ and\ L\ = \frac{b-a}{2}$$

The *Cumulative Distribution Function* (CDF) can be derived from the pdf model as follow:

- **For** $-L \geq x \geq 0$

$$\int_{-L}^{x} \frac{q+1}{2L}\left|\frac{z}{L}\right|^q dz = \frac{q+1}{2L}\int_{-L}^{x}\left|\frac{z}{L}\right|^q dz$$

$$= \frac{q+1}{2L}\left(\int_{0}^{L}\left(\frac{z}{L}\right)^q dz - \int_{0}^{x}\left(\frac{z}{L}\right)^q dz\right)$$

$$= \frac{q+1}{2L}\left(\int_{0}^{L}\left(\frac{z}{L}\right)^q dz - \int_{0}^{x}\left(\frac{z}{L}\right)^q dz\right)$$

$$= \frac{q+1}{2L}\left(\frac{1}{L^q(q+1)}z^{q+1}\Big|_0^L - \frac{1}{L^q(q+1)}z^{q+1}\Big|_0^x\right)$$

$$= \frac{q+1}{2L}\left(\frac{1}{L^q(q+1)}(L)^{q+1} - 0 - \frac{1}{L^q(q+1)}x^{q+1} - 0\right)$$

$$= \frac{q+1}{2L^{q+1}}\left(\frac{(L)^{q+1} - x^{q+1}}{q+1}\right)$$

$$\boldsymbol{F(x) = \frac{(L)^{q+1} - x^{q+1}}{2L^{q+1}} \approx \frac{1}{2}\left(1 - \frac{x^{q+1}}{L^{q+1}}\right)}$$

- **For** $0 \leq x \leq L$

$$\int_{-L}^{x} \frac{q+1}{2L}\left|\frac{z}{L}\right|^q dz = \frac{q+1}{2L}\int_{-L}^{0}\left|\frac{z}{L}\right|^q dz + \frac{q+1}{2L}\int_{0}^{x}\left|\frac{z}{L}\right|^q dz$$

$$= \frac{q+1}{2L}\left(\int_{-L}^{0}\left|\frac{z}{L}\right|^q dz + \int_{0}^{x}\left|\frac{z}{L}\right|^q dz\right)$$

$$= \frac{q+1}{2L}\left(\int_{0}^{L}\left(\frac{z}{L}\right)^q dz + \int_{0}^{x}\left(\frac{z}{L}\right)^q dz\right)$$

$$= \frac{q+1}{2L}\left(\frac{1}{L^q(q+1)}z^{q+1}\Big|_0^L + \frac{1}{L^q(q+1)}z^{q+1}\Big|_0^x\right)$$

$$= \frac{q+1}{2L}\left(\frac{1}{L^q(q+1)}(L)^{q+1} + \frac{1}{L^q(q+1)}(x)^{q+1}\right)$$

$$= \frac{q+1}{2L^{q+1}}\left(\frac{x^{q+1}+L^{q+1}}{q+1}\right)$$

$$\boldsymbol{F(x) = \frac{x^{q+1} + (L)^{q+1}}{2L^{q+1}} \approx \frac{1}{2}\left(1 + \frac{x^{q+1}}{L^{q+1}}\right)}$$

Then from the cdf equation obtained, given $F(x) \equiv p \in [0,1]$ the distribution can be reconstructed by computing the inverse of the cdf, which is given as follow:

- **For** $-L \geq x \geq 0$

$$F(x) = \frac{(L)^{q+1} - x^{q+1}}{2L^{q+1}}$$

$$F(x)2L^{q+1} = (L)^{q+1} - x^{q+1}$$

$$(1 - F(x)2)L^{q+1} = x^{q+1}$$

$$\log\left((1 - F(x)2)L^{q+1}\right) = \log(x^{q+1})$$

$$\log\left((1 - F(x)2)L^{q+1}\right) = (q+1)\log(x)$$

$$\log\left((1 - F(x)2)L^{q+1}\right)^{\frac{1}{q+1}} = \log(x)$$

$$\boldsymbol{F^{-1}(x) = (1 - 2p)^{\frac{1}{q+1}}L}$$

- **For** $0 \leq x \leq L$

$$F(x) = \frac{x^{q+1} + (L)^{q+1}}{2L^{q+1}}$$

$$F(x)2L^{q+1} = x^{q+1} + (L)^{q+1}$$

$$(F(x)2 - 1)L^{q+1} = x^{q+1}$$

$$\log\left((F(x)2 - 1)L^{q+1}\right) = \log(x^{q+1})$$

$$\log\left((F(x)2 - 1)L^{q+1}\right) = (q+1)\log(x)$$

$$\log\left((F(x)2 - 1)L^{q+1}\right)^{\frac{1}{q+1}} = \log(x)$$

$$\boldsymbol{F^{-1}(x) = (2p - 1)^{\frac{1}{q+1}}L}$$

Appendix 5. Newton's Method

Appendix 5.1. Floating Point Model of Newton's Method

As the computation for division operation is performed independently, the numerator and denominator can be separated into two independent model which is given as follow:

Appendix 5.1.1. Numerator Floating Point Model

Pseudo-code	Floating Point Model
$\boldsymbol{a = x * x;}$	$a = x^2(1+\delta_1)$
$\boldsymbol{b = a * x;}$	$b = x^3(1+\delta_1)(1+\delta_2)$
$\boldsymbol{c = (b + a);}$	$c = (x^2(1+\delta_1) + x^3(1+\delta_1)(1+\delta_2))(1+\delta_3)$
$\boldsymbol{d = 2 * x;}$	$d = 2x(1+\delta_4)$
$\boldsymbol{e = (c - d);}$	$e = ((x^2(1+\delta_1) + x^3(1+\delta_1)(1+\delta_2))(1+\delta_3) - 2x(1+\delta_4))(1+\delta_5)$

Appendix 5.1.2. Denominator Floating Point Model

Pseudo-code	Floating Point Model
$\boldsymbol{a = x * x;}$	$a = x^2(1+\delta_1)$
$\boldsymbol{b = 3 * a;}$	$b = 3x^2(1+\delta_1)(1+\delta_2)$
$\boldsymbol{c = 2 * x;}$	$c = 2x(1+\delta_3)$
$\boldsymbol{d = b + c;}$	$d = (3x^2(1+\delta_1)(1+\delta_2) + 2x(1+\delta_3))(1+\delta_4)$
$\boldsymbol{e = d - 2;}$	$e = (-2 + (3x^2(1+\delta_1)(1+\delta_2) + 2x(1+\delta_3))(1+\delta_4))(1+\delta_5)$

Appendix 5.2. Bound Deduction through Handelman Representation

Appendix 5.2.1. Handelman Representation for the Numerator Model

From the floating point model in Appendix 4.1.1., The Handelman Representation for the Numerator is given as such:

First Iteration,

$$f_1(x) = -2x + x^2 + x^3 + (x^2 + x^3)\delta_1 + x^3\delta_2 + x^3\delta_1\delta_2 + (x^2 + x^3)\delta_3 + (x^2 + x^3)\delta_1\delta_3 + x^3\delta_2\delta_3 + \boldsymbol{x^3\delta_1\delta_2\delta_3} - 2x\delta_4$$

Highest order monomial: $\boldsymbol{x^3\delta_1\delta_2\delta_3}$

Handelman Representation: $h_1(\delta) = x^3(\Delta + \delta_1)(\Delta + \delta_2)(\Delta + \delta_3)$

$$f_2(\delta) = f_1(\delta) - h_1(\delta)$$
$$= -2x + x^2 + x^3 - x^3\Delta^3 + (x^2 + x^3 - x^3\Delta^2)\delta_1 + (x^3 - x^3\Delta^2)\delta_2 + (x^3 - x^3\Delta)\delta_1\delta_2 + (x^2 + x^3 - x^3\Delta^2)\delta_3 + (x^2 + x^3 - x^3\Delta)\delta_1\delta_3 + (x^3 - x^3\Delta)\delta_2\delta_3 - 2x\delta_4$$

Second Iteration,

$$f_2(\delta) = -2x + x^2 + x^3 - x^3\Delta^3 + (x^2 + x^3 - x^3\Delta^2)\delta_1 + (x^3 - x^3\Delta^2)\delta_2 + (x^3 - x^3\Delta)\delta_1\delta_2 + (x^2 + x^3 - x^3\Delta^2)\delta_3 + (x^2 + x^3 - x^3\Delta)\delta_1\delta_3 + (\boldsymbol{x^3 - x^3\Delta})\boldsymbol{\delta_2\delta_3} - 2x\delta_4$$

Highest order monomial: $(\boldsymbol{x^3 - x^3\Delta})\boldsymbol{\delta_2\delta_3}$

Handelman Representation: $h_2(\delta) = (x^3 - x^3\Delta)(\Delta + \delta_2)(\Delta + \delta_3)$

$$f_3(\delta) = f_2(\delta) - h_2(\delta)$$
$$= -2x + x^2 + x^3 - x^3\Delta^2 + (x^2 + x^3 - x^3\Delta^2)\delta_1 + (x^3 - x^3\Delta)\delta_2 + (x^3 - x^3\Delta)\delta_1\delta_2 + (x^2 + x^3 - x^3\Delta)\delta_3 + (x^2 + x^3 - x^3\Delta)\delta_1\delta_3 - 2x\delta_4$$

Third Iteration,

$$f_3(\delta) = -2x + x^2 + x^3 - x^3\Delta^2 + (x^2 + x^3 - x^3\Delta^2)\delta_1 + (x^3 - x^3\Delta)\delta_2 + (x^3 - x^3\Delta)\delta_1\delta_2 + (x^2 + x^3 - x^3\Delta)\delta_3 + (\boldsymbol{x^2 + x^3 - x^3\Delta})\boldsymbol{\delta_1\delta_3} - 2x\delta_4$$

Highest order monomial: $(\boldsymbol{x^2 + x^3 - x^3\Delta})\boldsymbol{\delta_1\delta_3}$

Handelman Representation: $h_3(\delta) = (x^2 + x^3 - x^3\Delta)(\Delta + \delta_1)(\Delta + \delta_3)$

$$f_4(\delta) = f_3(\delta) - h_3(\delta)$$
$$= -2x + x^2 + x^3 - x^2\Delta^2 - 2x^3\Delta^2 + x^3\Delta^3 + (x^2 + x^3 - x^2\Delta - x^3\Delta)\delta_1 + (x^3 - x^3\Delta)\delta_2 + (x^3 - x^3\Delta)\delta_1\delta_2 + (x^2 + x^3 - x^2\Delta - 2x^3\Delta + x^3\Delta^2)\delta_3 - 2x\delta_4$$

Fourth Iteration,

$$f_4(\delta) = -2x + x^2 + x^3 - x^2\Delta^2 - 2x^3\Delta^2 + x^3\Delta^3 + (x^2 + x^3 - x^2\Delta - x^3\Delta)\delta_1 + (x^3 - x^3\Delta)\delta_2 + (\boldsymbol{x^3 - x^3\Delta})\boldsymbol{\delta_1\delta_2} + (x^2 + x^3 - x^2\Delta - 2x^3\Delta + x^3\Delta^2)\delta_3 - 2x\delta_4$$

Highest order monomial: $(\boldsymbol{x^3 - x^3\Delta})\boldsymbol{\delta_1\delta_2}$

Handelman Representation: $h_4(\delta) = (x^3 - x^3\Delta)(\Delta^2 + \delta_1\delta_2)$

$$f_4(\delta) = f_3(\delta) - h_3(\delta)$$
$$= -2x + x^2 + x^3 - x^2\Delta^2 - 3x^3\Delta^2 + 2x^3\Delta^3 + (x^2 + x^3 - x^2\Delta - x^3\Delta)\delta_1 + (x^3 - x^3\Delta)\delta_2 + (x^2 + x^3 - x^2\Delta - 2x^3\Delta + x^3\Delta^2)\delta_3 - 2x\delta_4$$

As the rest of the monomials are first order monomials, there is only one way to cancel it.

$(x^2 + x^3 - x^2\Delta - x^3\Delta)\delta_1 \rightarrow (x^2 + x^3 - x^2\Delta - x^3\Delta)(\Delta + \delta_1)$

$(x^3 - x^3\Delta)\delta_2 \rightarrow (x^3 - x^3\Delta)(\Delta + \delta_2)$

$(x^2 + x^3 - x^2\Delta - 2x^3\Delta + x^3\Delta^2)\delta_3 \rightarrow (x^2 + x^3 - x^2\Delta - 2x^3\Delta + x^3\Delta^2)(\Delta + \delta_3)$

$-2x\delta_4 \rightarrow 2x(\Delta - \delta_4)$

By applying the $f(\delta) - \hat{\gamma} = p_{ghr}$, where the p_{ghr} can be expressed as

$$p_{ghr} = 2x\Delta + 2x^2\Delta + 3x^3\Delta - x^2\Delta^2 - x^3\Delta^2 - x^3\Delta^3 + x^2\delta_1 + x^3\delta_1 + x^3\delta_2 + x^3\delta_1\delta_2 + x^2\delta_3 + x^3\delta_3 + x^2\delta_1\delta_3 + x^3\delta_1\delta_3 + x^3\delta_2\delta_3 + x^3\delta_1\delta_2\delta_3 - 2x\delta_4$$

Hence the bound $\hat{\gamma}$ can be obtained as following.

$$f(\delta) - p_{ghr} = \hat{\gamma}$$

$$\hat{\gamma} = -2x + x^2 + x^3 - 2x\Delta - 2x^2\Delta - 3x^3\Delta + x^2\Delta^2 + x^3\Delta^2 + x^3\Delta^3$$

Appendix 5.2.2. Handelman Representation for the Denominator Model

From the floating point model in Appendix 4.1.2., The Handelman Representation for the Numerator is given as such:

First Iteration,

$$g_1(\delta) = -2 + 2x + 3x^2 + 3x^2\delta_1 + 3x^2\delta_2 + 3x^2\delta_1\delta_2 + 2x\delta_3 + (2x + 3x^2)\delta_4 + 3x^2\delta_1\delta_4 + 3x^2\delta_2\delta_4 + \boldsymbol{3x^2\delta_1\delta_2\delta_4} + 2x\delta_3\delta_4$$

Highest order monomial: $\boldsymbol{3x^2\delta_1\delta_2\delta_4}$

Handelman Representation: $h_1(\delta) = 3x^2(\Delta + \delta_1)(\Delta + \delta_2)(\Delta + \delta_4)$

$$g_2(\delta) = g_1(\delta) - h_1(\delta)$$
$$= -2 + 2x + 3x^2 - 3x^2\Delta^3 + (3x^2 - 3x^2\Delta^2)\delta_1 + (3x^2 - 3x^2\Delta^2)\delta_2 + (3x^2 - 3x^2\Delta)\delta_1\delta_2 + 2x\delta_3 + (2x + 3x^2 - 3x^2\Delta^2)\delta_4 + (3x^2 - 3x^2\Delta)\delta_1\delta_4 + (3x^2 - 3x^2\Delta)\delta_2\delta_4 + 2x\delta_3\delta_4$$

Second Iteration,

$$g_2(\delta) = -2 + 2x + 3x^2 - 3x^2\Delta^3 + (3x^2 - 3x^2\Delta^2)\delta_1 + (3x^2 - 3x^2\Delta^2)\delta_2 + (3x^2 - 3x^2\Delta)\delta_1\delta_2 + 2x\delta_3 + (2x + 3x^2 - 3x^2\Delta^2)\delta_4 + (3x^2 - 3x^2\Delta)\delta_1\delta_4 + (3x^2 - 3x^2\Delta)\delta_2\delta_4 + \boldsymbol{2x\delta_3\delta_4}$$

Highest order monomial: $\boldsymbol{2x\delta_3\delta_4}$

Handelman Representation: $h_2(\delta) = 2x(\Delta + \delta_3)(\Delta + \delta_4)$

$$g_3(\delta) = g_2(\delta) - h_2(\delta)$$
$$= -2 + 2x + 3x^2 - 2x\Delta^2 - 3x^2\Delta^3 + (3x^2 - 3x^2\Delta^2)\delta_1 + (3x^2 - 3x^2\Delta^2)\delta_2 + (3x^2 - 3x^2\Delta)\delta_1\delta_2 + (2x - 2x\Delta)\delta_3 + (2x + 3x^2 - 2x\Delta - 3x^2\Delta^2)\delta_4 + (3x^2 - 3x^2\Delta)\delta_1\delta_4 + (3x^2 - 3x^2\Delta)\delta_2\delta_4$$

Third Iteration,

$$g_3(\delta) = -2 + 2x + 3x^2 - 2x\Delta^2 - 3x^2\Delta^3 + (3x^2 - 3x^2\Delta^2)\delta_1 + (3x^2 - 3x^2\Delta^2)\delta_2 + (3x^2 - 3x^2\Delta)\delta_1\delta_2 + (2x - 2x\Delta)\delta_3 + (2x + 3x^2 - 2x\Delta - 3x^2\Delta^2)\delta_4 + (3x^2 - 3x^2\Delta)\delta_1\delta_4 + (\boldsymbol{3x^2 - 3x^2\Delta})\boldsymbol{\delta_2\delta_4}$$

Highest order monomial: $(\boldsymbol{3x^2 - 3x^2\Delta})\boldsymbol{\delta_2\delta_4}$

Handelman Representation: $h_3(\delta) = (3x^2 - 3x^2\Delta)(\Delta + \delta_2)(\Delta + \delta_4)$

$$g_4(\delta) = g_3(\delta) - h_3(\delta)$$
$$= -2 + 2x + 3x^2 - 2x\Delta^2 - 3x^2\Delta^2 + (3x^2 - 3x^2\Delta^2)\delta_1 + (3x^2 - 3x^2\Delta)\delta_2 + (3x^2 - 3x^2\Delta)\delta_1\delta_2 + (2x - 2x\Delta)\delta_3 + (2x + 3x^2 - 2x\Delta - 3x^2\Delta)\delta_4 + (3x^2 - 3x^2\Delta)\delta_1\delta_4$$

Fourth Iteration,

$$g_4(\delta) = -2 + 2x + 3x^2 - 2x\Delta^2 - 3x^2\Delta^2 + (3x^2 - 3x^2\Delta^2)\delta_1 + (3x^2 - 3x^2\Delta)\delta_2 + (3x^2 - 3x^2\Delta)\delta_1\delta_2 + (2x - 2x\Delta)\delta_3 + (2x + 3x^2 - 2x\Delta - 3x^2\Delta)\delta_4 + (\boldsymbol{3x^2 - 3x^2\Delta})\boldsymbol{\delta_1\delta_4}$$

Highest order monomial: $(\boldsymbol{3x^2 - 3x^2\Delta})\boldsymbol{\delta_1\delta_4}$

Handelman Representation: $h_4(\delta) = (3x^2 - 3x^2\Delta)(\Delta^2 + \delta_1\delta_4)$

$$g_5(\delta) = g_4(\delta) - h_4(\delta)$$
$$= -2 + 2x + 3x^2 - 2x\Delta^2 - 6x^2\Delta^2 + 3x^2\Delta^3 + (3x^2 - 3x^2\Delta^2)\delta_1 + (3x^2 - 3x^2\Delta)\delta_2 + (3x^2 - 3x^2\Delta)\delta_1\delta_2 + (2x - 2x\Delta)\delta_3 + (2x + 3x^2 - 2x\Delta - 3x^2\Delta)\delta_4$$

Fifth Iteration,

$$g_5(\delta) = -2 + 2x + 3x^2 - 2x\Delta^2 - 6x^2\Delta^2 + 3x^2\Delta^3 + (3x^2 - 3x^2\Delta^2)\delta_1 + (3x^2 - 3x^2\Delta)\delta_2 + (\boldsymbol{3x^2 - 3x^2\Delta})\boldsymbol{\delta_1\delta_2} + (2x - 2x\Delta)\delta_3 + (2x + 3x^2 - 2x\Delta - 3x^2\Delta)\delta_4$$

Highest order monomial: $(\boldsymbol{3x^2 - 3x^2\Delta})\boldsymbol{\delta_1\delta_2}$

Handelman Representation: $h_5(\delta) = (3x^2 - 3x^2\Delta)(\Delta^2 + \delta_1\delta_2)$

$$g_5(\delta) = g_4(\delta) - h_4(\delta)$$
$$= -2 + 2x + 3x^2 - 2x\Delta^2 - 9x^2\Delta^2 + 6x^2\Delta^3 + (3x^2 - 3x^2\Delta^2)\delta_1 + (3x^2 - 3x^2\Delta)\delta_2 + (2x - 2x\Delta)\delta_3 + (2x + 3x^2 - 2x\Delta - 3x^2\Delta)\delta_4$$

As the rest of the monomials are first order monomials, there is only one way to cancel it.

$(3x^2 - 3x^2\Delta^2)\delta_1 \rightarrow (3x^2 - 3x^2\Delta^2)(\Delta + \delta_1)$

$(3x^2 - 3x^2\Delta)\delta_2 \rightarrow (3x^2 - 3x^2\Delta)(\Delta + \delta_2)$

$(2x - 2x\Delta)\delta_3 \rightarrow (2x - 2x\Delta)(\Delta + \delta_3)$

$(2x + 3x^2 - 2x\Delta - 3x^2\Delta)\delta_4 \rightarrow (2x + 3x^2 - 2x\Delta - 3x^2\Delta)(\Delta + \delta_4)$

By applying the $f(\delta) - \hat{\gamma} = p_{ghr}$, where the p_{ghr} can be expressed as

$$p_{ghr} = 4x\Delta + 9x^2\Delta - 2x\Delta^2 + 3x^2\Delta^2 - 9x^2\Delta^3 + 3x^2\delta_1 + 3x^2\delta_2 + 3x^2\delta_1\delta_2 + 2x\delta_3 + 2x\delta_4 + 3x^2\delta_4 + 3x^2\delta_1\delta_4 + 3x^2\delta_2\delta_4 + 3x^2\delta_1\delta_2\delta_4 + 2x\delta_3\delta_4$$

Hence the bound $\hat{\gamma}$ can be obtained as following.

$$f(\delta) - p_{ghr} = \hat{\gamma}$$

$$\hat{\gamma} = -2 + 2x + 3x^2 - 4x\Delta - 9x^2\Delta + 2x\Delta^2 - 3x^2\Delta^2 + 9x^2\Delta^3$$

Appendix 5.3. Bound comparison of Newton's Method model

Since division operation requires the bound to be computed independently, there are 5 stages of computation in which the bound comparison is given in table below.

Method	$X[k]$	**Lower Bound**	**Upper Bound**	**Time**
IA	$z1$	-2.1740	4.2448	0.445439s
	$z2$	0.8621	7.7833	0.498063s
	$z3$	-2.5242	4.9285	0.537931s
	$z4$	-4.2326	4.0281	0.543539s
	$z5$	-8.2234	8.1844	0.559673s
AA	$z1$	-1.6826	2.7046	0.647533s
	$z2$	0.8386	7.8836	0.691083s
	$z3$	-2.0142	2.9372	0.825858s
	$z4$	-1.4428	3.2020	0.833091s
	$z5$	-4.4026	4.3902	0.882249s
MC with 10^7 *iterations*	$z1$	-0.5805	2.6945	40.306572s

	$z2$	0.8460	7.8593	40.393770s
	$z3$	-0.6703	0.3468	40.484554s
	$z4$	0.9925	1.3661	40.557039s
	$z5$	-0.0147	0.0155	40.624649s
MC with 10^8 *iterations*	$z1$	-0.5809	2.6956	371.881896s
	$z2$	0.8429	7.8735	371.965702s
	$z3$	-0.6723	0.3473	372.070656s
	$z4$	0.9924	1.3676	372.160649s
	$z5$	-0.0148	0.0159	372.238360s
GHR	$z1$	-0.5803	2.5562	NA
	$z2$	0.8418	7.6474	NA
	$z3$	-0.6920	3.0484	NA
	$z4$	-2.3391	2.1833	NA
	$z5$	-3.6904	1.0221	NA
Moment	$z1$	-0.6340	2.7046	71.160727s
	$z2$	0.8345	7.8925	144.579594s
	$z3$	-0.6920	3.0484	145.217412s
	$z4$	-2.3391	2.1833	163.977053s
	$z5$	-3.9637	1.3205	179.337622s
SMT	$z1$	-0.5839	2.6969	NA
	$z2$	0.8386	7.8836	NA
	$z3$	-0.6989	0.3554	NA
	$z4$	0.9854	1.4044	NA
	$z5$	-0.0491	0.0527	NA
*Estimated True Bounds**	$z1$	-0.5826	2.7046	6406.383688s
	$z2$	0.8397	7.8836	6406.782695s
	$z3$	-0.6736	0.3478	6407.073029s
	$z4$	1.1537	1.3682	6407.242726s
	$z5$	-0.0155	0.0165	6407.381004s

In order to estimate the tightest bound possible, the moment method used different probability level and input distribution, which are as follow:

$z1$ → Probability level of $1e-4$ with $q = 7$

$z2$ → Probability level of $1e-4$ with $q = 25$

$z4$ → Probability level of $1e-4$ with $q = 30$

Appendix 6. Determinant of a Toeplitz Matrix

Appendix 6.1. Floating Point Model for Determinant of a Toeplitz Matrix

Given a, b, c, d are the element of the symmetrical Toeplitz matrix which is given as constant which is represented in floating point format, which can be written as $a \equiv x(1+\delta) \equiv [x - 2^{-m}x, x + 2^{-m}x] = x \pm 2^{-m}x$. The determinant model can be derived as such:

a. Matrix 2×2

$$\begin{vmatrix} a & b \\ b & a \end{vmatrix} = \left(a^2(1+\delta_1) - b^2(1+\delta_2)\right)(1+\delta_3)$$
$$= (a^2 + a^2\delta_1 - b^2 - b^2\delta_2)(1+\delta_3)$$
$$= a^2 - b^2 + a^2\delta_1 - b^2\delta_2 + a^2\delta_3 - b^2\delta_3 + a^2\delta_1\delta_3 - b^2\delta_2\delta_3$$

b. Matrix 3×3

$$\begin{vmatrix} a & b & c \\ b & a & b \\ c & b & a \end{vmatrix} = ((a\begin{vmatrix} a & b \\ b & a \end{vmatrix}(1+\delta_4) - b\begin{vmatrix} b & b \\ c & a \end{vmatrix}(1+\delta_8))(1+\delta_9) + c\begin{vmatrix} b & a \\ c & b \end{vmatrix}(1+\delta_{13}))(1+\delta_{14})$$

And every determinant 2×2 can be derived as such:

$$\begin{vmatrix} a & b \\ b & a \end{vmatrix} = \left(a^2(1+\delta_1) - b^2(1+\delta_2)\right)(1+\delta_3)$$
$$= a^2 - b^2 + a^2\delta_1 - b^2\delta_2 + a^2\delta_3 - b^2\delta_3 + a^2\delta_1\delta_3 - b^2\delta_2\delta_3$$

$$\begin{vmatrix} b & b \\ c & a \end{vmatrix} = \left(ab(1+\delta_5) - bc(1+\delta_6)\right)(1+\delta_7)$$
$$= ab - bc + ab\delta_5 - bc\delta_6 + ab\delta_7 - bc\delta_7 + ab\delta_5\delta_7 - bc\delta_6\delta_7$$

$$\begin{vmatrix} b & a \\ c & b \end{vmatrix} = \left(b^2(1+\delta_{10}) - ac(1+\delta_{11})\right)(1+\delta_{12})$$
$$= b^2 - ac + b^2\delta_{10} - ac\delta_{11} + b^2\delta_{12} - ac\delta_{12} + b^2\delta_{10}\delta_{12} - ac\delta_{11}\delta_{12}$$

Therefore, the overall floating point model can be written as follow:

$$\begin{aligned} |M| = ((&a(a^2 - b^2 + a^2\delta_1 - b^2\delta_2 + a^2\delta_3 - b^2\delta_3 + a^2\delta_1\delta_3 - b^2\delta_2\delta_3)(1+\delta_4) \\ &- b(ab - bc + ab\delta_5 - bc\delta_6 + ab\delta_7 - bc\delta_7 + ab\delta_5\delta_7 - bc\delta_6\delta_7)(1 \\ &+ \delta_8))(1+\delta_9) \\ &+ c(b^2 - ac + b^2\delta_{10} - ac\delta_{11} + b^2\delta_{12} - ac\delta_{12} + b^2\delta_{10}\delta_{12} - ac\delta_{11}\delta_{12})(1 \\ &+ \delta_{13}))(1+\delta_{14}) \end{aligned}$$

$$
\begin{aligned}
|M| = (a^3 - 2ab^2 + 2b^2c - ac^2 + a^3\delta_1 - ab^2\delta_2 + a^3\delta_3 - ab^2\delta_3 + a^3\delta_1\delta_3 - ab^2\delta_2\delta_3 \\
+ a^3\delta_4 - ab^2\delta_4 + a^3\delta_1\delta_4 - ab^2\delta_2\delta_4 + a^3\delta_3\delta_4 - ab^2\delta_3\delta_4 + a^3\delta_1\delta_3\delta_4 \\
- ab^2\delta_2\delta_3\delta_4 - ab^2\delta_5 + b^2c\delta_6 - ab^2\delta_7 + b^2c\delta_7 - ab^2\delta_5\delta_7 + b^2c\delta_6\delta_7 \\
- ab^2\delta_8 + b^2c\delta_8 - ab^2\delta_5\delta_8 + b^2c\delta_6\delta_8 - ab^2\delta_7\delta_8 + b^2c\delta_7\delta_8 - ab^2\delta_5\delta_7\delta_8 \\
+ b^2c\delta_6\delta_7\delta_8 + a^3\delta_9 - 2ab^2\delta_9 + b^2c\delta_9 + a^3\delta_1\delta_9 - ab^2\delta_2\delta_9 + a^3\delta_3\delta_9 \\
- ab^2\delta_3\delta_9 + a^3\delta_1\delta_3\delta_9 - ab^2\delta_2\delta_3\delta_9 + a^3\delta_4\delta_9 - ab^2\delta_4\delta_9 + a^3\delta_1\delta_4\delta_9 \\
- ab^2\delta_2\delta_4\delta_9 + a^3\delta_3\delta_4\delta_9 - ab^2\delta_3\delta_4\delta_9 + a^3\delta_1\delta_3\delta_4\delta_9 - ab^2\delta_2\delta_3\delta_4\delta_9 \\
- ab^2\delta_5\delta_9 + b^2c\delta_6\delta_9 - ab^2\delta_7\delta_9 + b^2c\delta_7\delta_9 - ab^2\delta_5\delta_7\delta_9 + b^2c\delta_6\delta_7\delta_9 \\
- ab^2\delta_8\delta_9 + b^2c\delta_8\delta_9 - ab^2\delta_5\delta_8\delta_9 + b^2c\delta_6\delta_8\delta_9 - ab^2\delta_7\delta_8\delta_9 + b^2c\delta_7\delta_8\delta_9 \\
- ab^2\delta_5\delta_7\delta_8\delta_9 + b^2c\delta_6\delta_7\delta_8\delta_9 + b^2c\delta_{10} - ac^2\delta_{11} + b^2c\delta_{12} - ac^2\delta_{12} \\
+ b^2c\delta_{10}\delta_{12} - ac^2\delta_{11}\delta_{12} + b^2c\delta_{13} - ac^2\delta_{13} + b^2c\delta_{10}\delta_{13} - ac^2\delta_{11}\delta_{13} \\
+ b^2c\delta_{12}\delta_{13} - ac^2\delta_{12}\delta_{13} + b^2c\delta_{10}\delta_{12}\delta_{13} - ac^2\delta_{11}\delta_{12}\delta_{13})(1 + \delta_{14})
\end{aligned}
$$

From the model above, it can be observed that the error term δ are moving in tandem, which can be said that positive correlation exist for each variables. Therefore it can be concluded that there are strong correlation exist between each operands.

Appendix 7. Matrix Multiplications

Appendix 7.1. Matrix Multiplication Model with Dependency

x[k]	MM[4]		MC		IA		AA		Estimated True Bound	
	Lower Bound	**Upper Bound**	**Lower Bound**	**Upper Bound**	**Lower Bound**	**Upper Bound**	**Lower Bound**	**Upper Bound**	**Lower Bound**	**Upper Bound**
1	-0.3822	16.8575	0.0000	17.0009	-10.0391	17.0664	-6.0352	6.0352	0.0000	17.0294
2	-0.0118	0.0118	-0.0117	0.0117	-0.0275	0.0275	-6.0352	6.0352	-0.0118	0.0118
3	-0.3822	16.8575	0.0000	17.0273	-10.0391	17.0664	-6.0352	6.0352	0.0000	17.0358
4	-0.0118	0.0118	-0.0117	0.0117	-0.0275	0.0275	-0.0157	0.0157	-0.0118	0.0118
5	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	-0.0157	0.0157	0.0000	0.0000
6	-0.0118	0.0118	-0.0117	0.0117	-0.0275	0.0275	-0.0157	0.0157	-0.0118	0.0118
7	-0.3822	16.8575	0.0000	17.0043	-10.0391	17.0664	-6.0352	6.0352	0.0000	17.0413
8	-0.0118	0.0118	-0.0117	0.0117	-0.0275	0.0275	-6.0352	6.0352	-0.0118	0.0118
9	-0.3822	16.8575	0.0000	17.0235	-10.0391	17.0664	-6.0352	6.0352	0.0000	17.0358

[4] The bound of moment method is estimated by considering 10^{th} order moment and probability level of $1e-4$ with $q=1$

Appendix 7.2. Discrete Cosine Transform of a vector

Complete result obtained for the DCT of a vector is summarized in the following table.

Method	$X[k]$	**Lower Bound**	**Upper Bound**	**Time**
IA	1	-7.8125e-03	7.8125e-03	0.485071s
	2	3.5659e-01	3.6504e-01	
	3	-7.8125e-03	7.8125e-03	
	4	-8.8129e-01	-8.6087e-01	
AA	1	0.0000e+00	0.0000e+00	0.816236s
	2	3.5659e-01	3.6504e-01	
	3	-3.0557e-05	3.05573e-05	
	4	-8.8129e-01	-8.6087e-01	
MC with 10^8 *iterations*	1	-7.4577e-03	7.4546e-03	1856.884827s
	2	3.5682e-01	3.6483e-01	
	3	-7.4674e-03	7.4095e-03	
	4	-8.8075e-01	-8.6133e-01	
GHR	-	-	-	NA
Moment[5]	1	-7.8535e-03	7.8535e-03	5.257635s
	2	3.5655e-01	3.6505e-01	
	3	-7.8535e-03	7.8535e-03	
	4	-8.8130e-01	-8.6078e-01	
SMT	1	-2.6550e-3	2.6550e-3	NA
	2	-1.4603e-3	1.460e-3	
	3	-2.6550e-3	2.6550e-3	
	4	-3.4501e-3	3.4501e-3	
*Estimated True Bounds**	1	-7.8125e-03	7.8125e-03	NA
	2	3.5659e-01	3.6504e-01	
	3	-7.8125e-03	7.8125e-03	
	4	-8.8129e-01	-8.6087e-01	

[5] The bound of the moment method is estimated with $q = 0$ and probability level of $1e-4$

Appendix 7.3. 2-Dimensional Discrete Cosine Transform

The complete result obtained for 2-dimensional DCT matrix multiplication model 5×5 is summarized in the table below.

	MM		MC		IA		AA		Estimated True Bound	
x[k]	**Lower Bound**	**Upper Bound**	**Lower Bound**	**Upper Bound**	**Lower Bound**	**Upper Bound**	**Lower Bound**	**Upper Bound**	**Lower Bound**	**Upper Bound**
1	-0.0282	4.5694	0.0000	3.8755	0.0000	5.0391	0.4963	4.5291	0.0000	5.0391
2	-2.2087	2.0694	-1.4459	1.3712	-2.1933	2.1933	-1.7715	1.7732	-2.1933	2.1933
3	-2.8869	2.0694	-1.3434	1.4213	-2.3062	2.3062	-2.0822	2.0817	-2.3062	2.3062
4	-2.2087	2.0694	-1.3962	1.3886	-2.1933	2.1933	-1.5194	1.5109	-2.1933	2.1933
5	-2.8869	2.0694	-1.4125	1.4331	-2.3062	2.3062	-1.7243	1.7279	-2.3062	2.3062
6	-2.2086	2.0693	-1.3818	1.4428	-2.1933	2.1933	-0.0085	0.0085	-2.1933	2.1933
7	-1.9359	1.9359	-1.3136	1.3161	-1.9093	1.9093	-0.0074	0.0074	-1.9093	1.9093
8	-2.0307	2.0288	-1.4067	1.3986	-2.0075	2.0075	-0.0078	0.0078	-2.0075	2.0075
9	-1.9359	1.9358	-1.4225	1.3356	-1.9093	1.9093	-0.0074	0.0074	-1.9093	1.9093
10	-2.0307	2.0300	-1.3599	1.3637	-2.0075	2.0075	-0.0078	0.0078	-2.0075	2.0075
11	-2.8868	2.0693	-1.3880	1.3265	-2.3062	2.3062	-0.0089	0.0089	-2.3062	2.3062
12	-2.0307	2.0288	-1.3566	1.3653	-2.0075	2.0075	-0.0077	0.0077	-2.0075	2.0075
13	-2.1214	2.0693	-1.3496	1.3570	-2.1108	2.1108	-0.0081	0.0081	-2.1108	2.1108
14	-2.0307	2.0288	-1.3450	1.3234	-2.0075	2.0075	-0.0077	0.0077	-2.0075	2.0075
15	-2.1214	2.0693	-1.3517	1.3604	-2.1108	2.1108	-0.0081	0.0081	-2.1108	2.1108
16	-2.2086	2.0693	-1.4248	1.4044	-2.1933	2.1933	-0.0085	0.0085	-2.1933	2.1933
17	-1.9359	1.9358	-1.3265	1.3416	-1.9093	1.9093	-0.0074	0.0074	-1.9093	1.9093
18	-2.0307	2.0288	-1.3290	1.4118	-2.0075	2.0075	-0.0078	0.0078	-2.0075	2.0075
19	-1.9359	1.9358	-1.4388	1.3265	-1.9093	1.9093	-0.0074	0.0074	-1.9093	1.9093
20	-2.0307	2.0300	-1.3138	1.3458	-2.0075	2.0075	-0.0078	0.0078	-2.0075	2.0075
21	-2.8868	2.0693	-1.3977	1.4349	-2.3062	2.3062	-0.0090	0.0090	-2.3062	2.3062
22	-2.0307	2.0300	-1.3592	1.3845	-2.0075	2.0075	-0.0078	0.0078	-2.0075	2.0075

23	-2.0307	2.0300	-1.3547	1.3984	2.0075	2.1108	-0.0082	0.0082	2.0075	2.1108
24	-2.0307	2.0300	-1.3547	1.3639	-2.0075	2.0075	-0.0078	0.0078	-2.0075	2.0075
25	-2.1214	2.0693	-1.3645	1.3690	-2.1108	2.1108	-0.0082	0.0082	-2.1108	2.1108

In order to estimate tightest bound possible, different probability level is used to estimate the bounds for each element of the matrix, which is given as follow:

- Probability Level of 1e-11 for 7^{th}, 9^{th}, 17^{th}, and 19^{th} element
- Probability Level of 1e-12 for 10^{th}, 12^{th}, 14^{th}, 18^{th}, 20^{th}, 22^{nd}, 23^{rd}, and 24^{th} element
- Probability Level of 1e-13 for 13^{th}, 15^{th}, and 25^{th} element
- Probability Level of 1e-14 for 2^{nd}, 4^{th}, 6^{th} and 16^{th} element
- Probability Level of 1e-17 for first element
- Probability Level of 1e-28 for 3^{rd}, 5^{th}, 11^{th}, and 21^{st} element.

Appendix 8. Ethical Compliance Form for ECSE FYP Projects

School of Engineering
Monash University

This form should be used in conjunction with your final report and should be attached to it as an appendix.

Prior to conducting your project, you and your supervisor will have discussed the ethical implications of your research. Monash ethical guidelines are comprehensively detailed in the Monash Research Office website, http://www.monash.edu.au/researchoffice/ethics.php. Some of FYP projects may have ethical concerns related to research integrity, research data management and human ethics. Students should identify any ethical implications related to the project and should complete relevant forms if these ethical issues are relevant.

Are any of the above ethical concerns related to your project? (yes/no)
If yes to the previous question, have you completed the relevant clearance forms? (yes/no)
(All forms could be downloaded from above link)

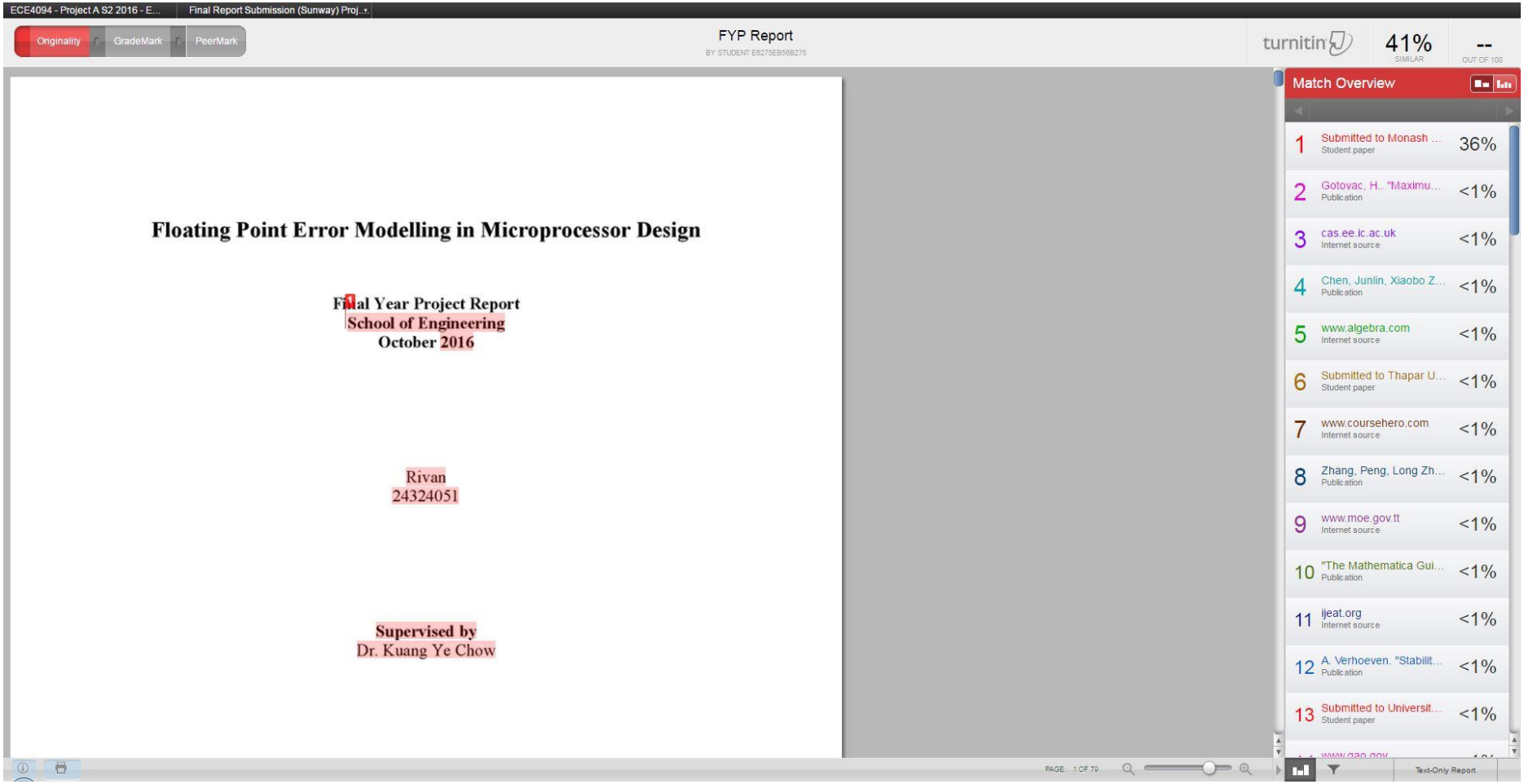
Project title: Floating Point Error Modelling in Microprocessor Design
Student's Name: Rivan
Student's ID Number: 24324051
Student's Signature: Rivan
Date: 20/10/2016

Appendix 9. Organization of the DVD

CD Contents:

1. **Article Folder**
2. **Source Codes Folder**
   - 2.1. Mathematica
   - 2.2. MATLAB
3. **Comparative Studies**
   - 3.1. Model 1. Simple Polynomial
   - 3.2. Model 2. Newton's Method
   - 3.3. Model 3. Toeplitz Matrix
   - 3.4. Model 4. Matrix Multiplication: Discrete Cosine Transform
4. **Final Report**
5. **Project A Folder**
   - 5.1. Design Document
   - 5.2. Requirement Analysis
   - 5.3. Risk Management Worksheet

Appendix 10. Turnitin Report



Reference

[1] C. F. Fang, R. A. Rutenbar, M. Puschel, and C. Tsuhan, "Toward efficient static analysis of finite-precision effects in DSP applications via affine arithmetic modeling," in *Design Automation Conference, 2003. Proceedings*, 2003, pp. 496-501.

[2] D. Boland and G. A. Constantinides, "A Scalable Precision Analysis Framework," *IEEE Transactions on Multimedia*, vol. 15, pp. 242-256, 2013.

[3] D. U. Lee, A. A. Gaffar, R. C. C. Cheung, O. Mencer, W. Luk, and G. A. Constantinides, "Accuracy-Guaranteed Bit-Width Optimization," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 25, pp. 1990-2000, 2006.

[4] C. F. Fang, R. A. Rutenbar, and C. Tsuhan, "Fast, accurate static analysis for fixed-point finite-precision effects in DSP designs," in *Computer Aided Design, 2003. ICCAD-2003. International Conference on*, 2003, pp. 275-282.

[5] A. B. Kinsman and N. Nicolici, "Bit-Width Allocation for Hardware Accelerators for Scientific Computing Using SAT-Modulo Theory," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 29, pp. 405-413, 2010.

[6] D. Boland and G. A. Constantinides, "Bounding Variable Values and Round-Off Effects Using Handelman Representations," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 30, pp. 1691-1704, 2011.

[7] Y. Pang, K. Radecka, and Z. Zilic, "Optimization of Imprecise Circuits Represented by Taylor Series and Real-Valued Polynomials," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 29, pp. 1177-1190, 2010.

[8] M. L. Chang and S. Hauck, "Automated least-significant bit datapath optimization for FPGAs," in *Field-Programmable Custom Computing Machines, 2004. FCCM 2004. 12th Annual IEEE Symposium on*, 2004, pp. 59-67.

[9] D. Boland and G. A. Constantinides, "Automated Precision Analysis: A Polynomial Algebraic Approach," in *Field-Programmable Custom Computing Machines (FCCM), 2010 18th IEEE Annual International Symposium on*, 2010, pp. 157-164.

[10] G. A. Constantinides, P. Y. K. Cheung, and W. Luk, "Wordlength optimization for linear digital signal processing," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 22, pp. 1432-1442, 2003.

[11] K. Ki-Il and S. Wonyong, "Combined word-length optimization and high-level synthesis of digital signal processing systems," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 20, pp. 921-930, 2001.

[12] P. Yu and H. Yajun, "An automated, efficient and static bit-width optimization methodology towards maximum bit-width-to-error tradeoff with affine arithmetic model," in *Asia and South Pacific Conference on Design Automation, 2006.*, 2006, p. 6 pp.

[13] A. B. Kinsman and N. Nicolici, "Finite Precision bit-width allocation using SAT-Modulo Theory," in *2009 Design, Automation & Test in Europe Conference & Exhibition*, 2009, pp. 1106-1111.

[14] L. H. d. F. J. STOLFI, "An Introduction to Affine Arithmetic," *TEMA Tend. Mat. Apl. Comput.*, vol. 4, pp. 297-312, 2003.

[15] R. Moore, *Methods and Applications of Interval Analysis*. Philadelphia: SIAM, 1979.

[16] R. E. Moore, *Interval analysis*: Prentice-Hall, 1966.

[17] L. H. d. F. J. STOLFI, *Affine Arithmetic*. Rio de Janeiro, Brazil: Brazilian Mathematics Colloquium, 1997.

[18] P. H. Sterbenz, *Floating-point computation*: Prentice-Hall, 1974.
[19] N. Higham, *Accuracy and Stability of Numerical Algorithms*, 2nd ed. Philadelphia, PA, USA: Soc for Industrial & Applied Math, 2002.
[20] D. Handelman, "Representing polynomials by positive linear functions on compact convex polyhedra," pp. 35-62, 1988 1988.
[21] Y. C. Kuang, A. Rajan, M. P.-L. Ooi, and T. C. Ong, "Standard uncertainty evaluation of multivariate polynomial," *Measurement*, vol. 58, pp. 483-494, 2014.
[22] A. Rajan, M. P. L. Ooi, Y. C. Kuang, and S. N. Demidenko, "Analytical Standard Uncertainty Evaluation Using Mellin Transform," *IEEE Access*, vol. 3, pp. 209-222, 2015.
[23] H. Gotovac and B. Gotovac, "Maximum entropy algorithm with inexact upper entropy bound based on Fup basis functions with compact support," *Journal of Computational Physics*, vol. 228, pp. 9079-9091, 12/20/ 2009.
[24] S. M. Rump, *INTLAB - INTerval LABoratory*. Dordrecht: Kluwer Academic Publishers, 1999.