arXiv:cs/0606103v22 [cs.DM] 12 Apr 2014

Precision Arithmetic[*]

Chengpu Wang
40 Grossman Street, Melville, NY 11747, USA
Chengpu@gmail.com

Abstract

A new deterministic floating-point arithmetic called *precision arithmetic* is developed to track precision for arithmetic calculations. It uses a novel rounding scheme to avoid the excessive rounding error propagation of conventional floating-point arithmetic. Unlike interval arithmetic, its uncertainty tracking is based on statistics and the central limit theorem, with a much tighter bounding range. Its stable rounding error distribution is approximated by a truncated Gaussian distribution. Generic standards and systematic methods for comparing uncertainty-bearing arithmetics are discussed. The precision arithmetic is found to be superior to interval arithmetic in both uncertainty-tracking and uncertainty-bounding for normal usages.

The arithmetic code is published at `http://precisionarithm.sourceforge.net`.

Keywords: computer arithmetic, error analysis, interval arithmetic, multi-precision arithmetic, numerical algorithms.
AMS subject classifications: 65-00

[*]Submitted: May 20, 2006; Revised: November 10, 2010; July 18, 2011; September 1, 2012; Match 27, 2014;

1 Introduction

1.1 Measurement Precision

Except for the simplest counting, scientific and engineering measurements never give completely precise results [1][2]. The precision of measured values ranges from an order-of-magnitude estimation of astronomical measurements to 10^{-2} to 10^{-4} of common measurements to 10^{-14} of state-of-art measurements of basic physics constants [3]. Such value which has uncertainty is called an *imprecise value*.

In scientific and engineering measurements, the uncertainty of a measurement x usually is characterized by the sample deviation δx [1][2][4]. In certain cases, such as raw reading from an ideal analog-to-digital converter, the uncertainty of a measurement x is given as a bounding range Δx[1] [5]. If $[x - \Delta \mathrm{x},\ x + \Delta x]$ crosses 0, x is neither positive nor negative for certainty due to the following two possibilities:

1. Either Δx is too large to give a precise measurement of x;
2. Or x itself is a measurement of zero.

To distinguish which case it is, additional information is required so that the measurement $x \pm\ \Delta x$ itself is *insignificant* if $[x - \Delta x, x + \Delta x]$ crosses 0. An insignificant value also has conceptual difficulty in participating in many mathematical operations, such as calculating the square root or acting as a divisor.

$P \equiv \delta x / |x|$ is defined here as the (relative) *precision* of the measurement, whose inverse is commonly known as the *significance* [1][2]. Precision represents the reliable information content of a measurement. Finer precision means higher reliability and thus better reproducibility of the measurement [1][2]. Taking the traditional definition in measurement, precision in this paper does not mean the maximal bit count of significand as in the term “arbitrary precision arithmetic”[2] [7].

1.2 Problem of Conventional Floating-Point Arithmetic

The *conventional floating-point arithmetic* [8][9][10] assumes a constant and best-possible precision for each value all the time, and constantly generates artificial information during the calculation [11]. For example, the following calculation is carried out precisely in integer format:

$$\begin{aligned} &64919121 \times 205117922 - 159018721 \times 83739041 = \\ &\quad 13316075197586562 - 13316075197586561 = 1; \end{aligned} \tag{1.1}$$

If Formula (1.1) is carried out using conventional floating-point arithmetic:

$$\begin{aligned} &64919121 \times 205117922 - 159018721 \times 83739041 = \\ &\quad 64919121.000000000 \times 205117922.000000000 \\ &-159018721.000000000 \times 83739041.000000000 = \\ &\quad 13316075197586562. - 13316075197586560. = 2. = 2.0000000000000000; \end{aligned} \tag{1.2}$$

[1] x is normally an integer as the output of an ADC (Analog-to-Digital Converter). Ideally, Δx equals a half bit of ADC. Δx can be larger if the settle time is not long enough, or if the ADC is not ideal.

[2] Arbitrary precision integer means a digital integer which has arbitrary number of bits, while arbitrary precision arithmetic usually means fixed-point arithmetic [6] which has arbitrary fractional bits.

1. The multiplication results exceed the maximal significance of the 64-bit IEEE floating-point representation; so they are rounded off, generating rounding errors;
2. The normalization of the subtraction result amplifies the rounding error to most significant bit (MSB) by padding zeros.

Formula (1.2) is a showcase for the problem of conventional floating-point arithmetic. Because normalization happens after each arithmetic operation [8][9][10], such generation of rounding errors happens very frequently for addition and multiplication, and such amplification of rounding errors happens very frequently for subtraction and division. The accumulation of rounding errors is an intrinsic problem of conventional floating-point arithmetic [12], and in the majority of cases such accumulation is almost uncontrollable [11]. For example, because a rounding error from lower digits quickly propagates to higher digits, the 10^{-7} precision resolution of the 32-bit IEEE floating-point format [8][9][10] is usually not fine enough for calculations involving input data of 10^{-2} to 10^{-4} precision.

Self-censored rules are developed to avoid such rounding error propagation [12][13], such as avoiding subtracting results of large multiplication, as in Formula (1.2). However, these rules are not enforceable, and in many cases are difficult to follow, e.g., even a most carefully crafted algorithm can result in numerical instability after extensive usage. Because the propagation speed of a rounding error depends on the nature of a calculation itself, e.g., generally faster in nonlinear algorithms than linear algorithms[3] [14], propagation of rounding error in conventional floating-point arithmetic is very difficult to quantify generically [15]. Thus, it is difficult to tell if a calculation is improper or becomes excessive for a required result precision. In common practice, reasoning on an individual theoretical base is used to estimate the error and validity of calculation results, such as from the estimated transfer functions of the algorithms used in the calculation [12][16][17]. However, such analysis is both rare and generally very difficult to carry out in practice.

Today most experimental data are collected by an ADC (Analog-to-Digital Converter) [5]. The result obtained from an ADC is an integer with fixed uncertainty; thus, a smaller signal value has a coarser precision. When a waveform containing raw digitalized signals from ADC is converted into conventional floating-point representation, the information content of the digitalized waveform is distorted to favour small signals since all converted data now have the same and best possible precision. However, the effects of such distortion in signal processing are generally not clear.

What is needed is a floating-point arithmetic that tracks precision automatically. When the calculation is improper or becomes excessive, the results become insignificant. All existing uncertainty-bearing arithmetics are reviewed below.

1.3 Interval Arithmetic

Interval arithmetic [13][18][19][20][21][22] is currently a standard method to track calculation uncertainty. It ensures that the value x is absolutely bounded within its *bounding range* $[x] \equiv [\underline{x}, \bar{x}]$, in which $\underline{x}$ and $\bar{x}$ are lower and upper bounds for x, respectively. In this paper, interval arithmetic is simplified and tested as the following

[3]A classic example is the contrast of the uncertainty propagation in the solutions for the 2nd-order linear differential equation vs. in those of Duffing equation (which has a x^3 term in addition to the x term in a corresponding 2nd-order linear differential equation).

arithmetic formulas[4] [20]:

$$[x_1] + [x_2] = [\underline{x}_1 + \underline{x}_2, \bar{x}_1 + \bar{x}_2]\,; \tag{1.3}$$
$$[x_1] - [x_2] = [\underline{x}_1 - \bar{x}_2, \bar{x}_1 - \underline{x}_2]\,; \tag{1.4}$$
$$[x_1] \times [x_2] = [min(\underline{x}_1\underline{x}_2, \underline{x}_1\bar{x}_2, \bar{x}_1\underline{x}_2, \bar{x}_1\bar{x}_2), max(\underline{x}_1\underline{x}_2, \underline{x}_1\bar{x}_2, \bar{x}_1\underline{x}_2, \bar{x}_1\bar{x}_2)]\,; \tag{1.5}$$
$$0 \notin [x_2]:\ [x_1]\,/\,[x_2] = [x_1] \times [1/\bar{x}_2, 1/\underline{x}_2]\,; \tag{1.6}$$

If interval arithmetic is implemented using a floating-point representation with limited resolution, its resulting bounding range is widened further [19].

A basic problem is that the bounding range used by interval arithmetic is not compatible with usual scientific and engineering measurements, which instead use the statistical mean and deviations to characterize uncertainty [1][2]. Most measured values are well approximated by a Gaussian distribution [1][2][4], which has no limited bounding range. Let *bounding leakage* be defined as the possibility of the true value to be outside a bounding range. If a bounding range is defined using a statistical rule on bounding leakage, such as the $6\sigma - 10^{-9}$ rule for Gaussian distribution [4] (which says that the bounding leakage is about 10^{-9} for a bounding range of mean $\pm$ 6-fold of standard deviations), there is no guarantee that the calculation result will also obey the $6\sigma - 10^{-9}$ rule using interval arithmetic, since interval arithmetic has no statistical foundation[5].

Another problem is that interval arithmetic only provides the worst case of uncertainty propagation, so that it tends to over-estimate uncertainty in reality. For instance, in addition and subtraction, it gives the result when the two operands are +1 and -1 correlated respectively [24]. However, if the two operands are -1 and +1 correlated respectively instead, the actual bounding range after addition and subtraction reduces, which is called the best case in random interval arithmetic [25]. The vast overestimation of bounding ranges in these two worst cases prompts the development of affine arithmetic [24][26], which traces error sources using a first-order model. Being expensive in execution and depending on approximate modeling even for such basic operations as multiplication and division, affine arithmetic has not been widely used. In another approach, random interval arithmetic [25] reduces the uncertainty over-estimation of standard interval arithmetic by randomly choosing between the best-case and the worst-case intervals.

A third problem is that the results of interval arithmetic may depend strongly on the actual expression of an analytic function $f(x)$. For example, Formula (1.7), Formula (1.8) and Formula (1.9) are different expressions of the same $f(x)$; however, the correct result is obtained only through Formula (1.7), and uncertainty may be exaggerated in the other two forms, e.g., by 67-fold and 33-fold at input range [0.49, 0.51] using Formula (1.8) and Formula (1.9), respectively. This is called the dependence

[4] For the mathematical definition of interval arithmetic, please see [22].

[5] There is some attempt [23] to connect intervals in interval arithmetic to confidence interval or the equivalent so called p-box in statistics. Because this attempt seems to rely heavily on 1) specific properties of the uncertainty distribution within the interval and/or 2) specific properties of the functions upon which the interval arithmetic is used, this attempt does not seem to be generic. Anyway, this attempt seems to be outside the main course of interval arithmetic, which has no statistics in mind.

problem of interval arithmetic [21].

$$f(x) = (x - 1/2)^2 - 1/4; \tag{1.7}$$
$$f(x) = x^2 - x; \tag{1.8}$$
$$f(x) = (x - 1)x; \tag{1.9}$$

Interval arithmetic has very coarse and algorithm-specific precision but constant zero bounding leakage. It represents the other extreme from conventional floating-point arithmetic. To meet practical needs, a better uncertainty-bearing arithmetic should be based on statistical propagation of the rounding error, while also allowing reasonable bounding leakage for normal usages.

1.4 Statistical Propagation of Uncertainty

If each operand is regarded as a random variable, and the statistical correlation between the two operands is known, the resulting uncertainty is given by the *statistical propagation of uncertainty* [27][28], with the following arithmetic equations, in which σ is the deviation of a measured value x, P is its precision, and γ is the correlation between the two operands x_1 and x_2:

$$(x_1 \pm \sigma_1) + (x_2 \pm \sigma_2) = (x_1 + x_2) \quad \pm\sqrt{\sigma_1^2 + \sigma_2^2 + 2\sigma_1\sigma_2\gamma}; \tag{1.10}$$
$$(x_1 \pm \sigma_1) - (x_2 \pm \sigma_2) = (x_1 - x_2) \quad \pm\sqrt{\sigma_1^2 + \sigma_2^2 - 2\sigma_1\sigma_2\gamma}; \tag{1.11}$$
$$(x_1 \pm \sigma_1) \times (x_2 \pm \sigma_2) = (x_1 \times x_2) \quad \pm|x_1 \times x_2|\sqrt{P_1^2 + P_2^2 + 2P_1P_2\gamma}; \tag{1.12}$$
$$(x_1 \pm \sigma_1) / (x_2 \pm \sigma_2) = (x_1 / x_2) \quad \pm|x_1 / x_2|\sqrt{P_1^2 + P_2^2 - 2P_1P_2\gamma}; \tag{1.13}$$

Tracking uncertainty propagation statistically seems an ideal solution. However, in practice, the correlation between two operands is generally not precisely known, so the direct use of statistical propagation of uncertainty is very limited. In this paper, as a proxy for statistical propagation of uncertainty, an *independence arithmetic* always assumes that no correlation exists between any two operands, whose arithmetic equations are Formula (1.10), Formula (1.11), Formula (1.12) and Formula (1.13), where $\gamma = 0$. Independence arithmetic is actually de facto arithmetic in engineering data processing, such as in the common belief that uncertainty after averaging reduces by the square root of number of measurements [1][2], or the ubiquitous Monte Carlo method[6] [30][29], or calculating the mean and variance of a Taylor expansion [31].

1.5 Significance Arithmetic

Significance arithmetic [32] tries to track reliable bits in an imprecise value during the calculation. In the two early attempts [33][34], the implementations of significance arithmetic are based on simple operating rules upon reliable bit counts, rather than on formal statistical approaches. They both treat the reliable bit counts as integers when applying their rules, while in reality a reliable bit count could be a fractional

[6]Most but not all applications of Monte Carlo methods assume independence between any two random variables. In a minority of applications, a Monte Carlo method can be used to construct specified correlation between two random variables [29].

number [35], so they both can cause artificial quantum reduction of significance. The significance arithmetic marketed by Mathematica [35] uses a linear error model that is consistent with a first-order approximation of interval arithmetic [13][20][21], and further provides an arbitrary precision representation which is in the framework of conventional floating-point arithmetic. It is definitely not a statistical approach.

Stochastic arithmetic [15][36], which can also be categorized as significance arithmetic, randomizes the least significant bits (LSB) of each of input floating-point values, repeats the same calculation multiple times, and then uses statistics to seek invariant digits among the calculation results as significant digits. This approach may require too much calculation since the number of necessary repeats for each input is specific to each algorithm, especially when the algorithm contains branches. Its sampling approach may be more time-consuming and less accurate than direct statistical characterization [4], such as directly calculating the mean and deviation of the underlying distribution. It is based on modeling rounding errors in conventional floating-point arithmetic, which is quite complicated. A better approach may be to define arithmetic rules that make error tracking by probability easier.

As the mathematical foundation to significance arithmetic, when a uncertainty-bearing value is multiplied by a constant, the significance or relative precision still holds, while the absolute precision [1][2] scales with the constant. In this respect, fixed-point arithmetic [6], which assumes a fixed absolute precision, does not have a sounding mathematical foundation.

1.6 An Overview of This Paper

In this paper, a new floating-point arithmetic called *precision arithmetic* [37] is developed to track uncertainty during floating-point calculations, as described in Section 2. Generic standards and systematic methods for validating uncertainty-bearing arithmetics are discussed in Section 3. Precision arithmetic is compared with other uncertainty-bearing arithmetics in Section 4 to Section 8. A brief discussion is provided in Section 10.

2 Precision Arithmetic

2.1 Assumptions for Precision Arithmetic

As stated previously, the precision P is defined as the (relative) precision of a measurement in this paper. Precision arithmetic tracks uncertainty distribution during calculations using specially designed arithmetic rules. It has the *uncorrelated uncertainty assumption* as its basic assumption, presuming that the uncertainties of any two different values can be regarded as uncorrelated of each other. This assumption can be turned into a realistic statistical requirement for input data for precision arithmetic.

Because it is not realistic to track the actual uncertainty distributions, which may vary according to each specific algorithm, the objectives of precision arithmetic are to enclose the actual uncertainty distribution with a *bounding distribution*:

1. The bounding distribution is symmetric around an expected value which is the value given by mathematics when there is no uncertainty.
2. The bounding distribution is Gaussian, with deviations calculated by precision arithmetic.

As shown later in this paper, the objectives of precision arithmetic are extended from the central limit theorem [4], so that the bounding distribution is a truncated Gaussian distribution, which approximates the actual uncertainty closely when there is decent amount of arithmetic calculations.

In addition, precision arithmetic uses heavily the *scaling principle* which says that the result precision should not change when an imprecise value is either multiplied or divided by a non-zero constant. The scaling principle can be concluded from Formula (1.12) and Formula (1.13) for statistical propagation of uncertainty. It is also the foundation for significance arithmetic.

Related to the scaling principle, the *recovering principle* says that the imprecise result should restore the original imprecise value if mathematically the original value is restored conceptually, such as when an imprecise value is inverted twice. In precision arithmetic, the value of the imprecise value obeys the recovering principle, while it is questionable if the uncertainty of the imprecise value can be recovered.

2.2 The Uncorrelated Uncertainty Assumption

When there is a good estimation of the sources of uncertainty, the uncorrelated uncertainty assumption can be judged directly, e.g., if noise [1][2] is the major source of uncertainty, the uncorrelated uncertainty assumption is probably true. This criterion is necessary to ascertain repeated measurements of the same signal. Otherwise, the uncorrelated uncertainty assumption can be judged by the correlation and the respectively precisions of two measurements.

Let X, Y, and Z denote three mutually independent random variables [4] with variance $\sigma^2(X)$, $\sigma^2(Y)$ and $\sigma^2(Z)$, respectively. Let α denote a constant. Let $Cov()$ denote the covariance function. Let γ denote the correlation between $(X+Y)$ and $(\alpha X + Z)$. And let:

$$\eta_1^2 \equiv \frac{\sigma^2(Y)}{\sigma^2(X)}; \quad \eta_2^2 \equiv \frac{\sigma^2(Z)}{\sigma^2(\alpha X)} = \frac{\sigma^2(Z)}{\alpha^2\sigma^2(X)}; \tag{2.1}$$

$$\gamma = \frac{Cov(X+Y, \alpha X + Z)}{\sqrt{\sigma^2(X+Y)}\sqrt{\sigma^2(\alpha X + Z)}} = \frac{\alpha/|\alpha|}{\sqrt{1+\eta_1^2}\sqrt{1+\eta_2^2}} \equiv \frac{\alpha/|\alpha|}{1+\eta^2}; \tag{2.2}$$

Formula (2.2) gives the correlation γ between two random variables, each of which contains a completely uncorrelated part and a completely correlated part, with η being the average ratio between these two parts. Formula (2.2) can also be interpreted reversely: if two random variables are correlated by γ, each of them can be viewed as containing a completely uncorrelated part and a completely correlated part, with η being the average ratio between these two parts.

One special application of Formula (2.2) is the correlation between a measured signal and its true signal, in which noise is the uncorrelated part between the two. Figure 1 shows the effect of noise on the most significant two bits of a 4-bit measured signal when $\eta = 1/4$. Its top chart shows a triangular waveform between 0 and 16 as a black line, and a white noise between -2 and +2, using the grey area. The measured signal is the sum of the triangle waveform and the noise. The middle chart of Figure 1 shows the values of the 3rd digit of the true signal as a black line, and the mean values of the 3rd bit of the measurement as a grey line. The 3rd bit is affected by the noise during its transition between 0 and 1. For example, when the signal is slightly below 8, only a small positive noise can turn the 3rd digit from 0 to 1. The bottom chart of Figure 1 shows the values of the 2nd digit of the signal and the measurement as a black line and a grey line, respectively. Figure 1 clearly shows that the correlation between the measurement and the true signal is less at the 2nd digit than at the 3rd digit. Quantitatively, according to Formula (2.2):

1. The overall measurement is 99.2% correlated to the signal with $\eta = 1/8$;
2. The 3rd digit of the measurement is 97.0% correlated to the signal with $\eta = 1/4$;
3. The 2nd digit of the measurement is 89.4% correlated to the signal with $\eta = 1/2$;
4. The 1st digit of the measurement is 70.7% correlated to the signal with $\eta = 1$;
5. The 0th digit of the measurement is 44.7% correlated to the signal with $\eta = 2$.

The above conclusion agrees with the common experiences that, below the noise level of measured signals, noises rather than true signals dominate each digit.

Similarly, while the correlated portion between two values has exactly the same value at each bit of the two values, the ratio of the uncorrelated portion to the correlated portion increases by 2-fold for each bit down from MSB of the two values, regardless of the nature of the uncorrelated portion. Quantitatively, let P denote the larger precision of the two values, and let η_P denote the ratio of the uncorrelated portion to the correlated portion at level of uncertainty; then η_P increases with decreased P according to Formula (2.3). According to Formula (2.2), if two significant values are overall correlated with γ, at the level of uncertainty the correlation between the two values decreases to γ_P according to Formula (2.4).

$$\eta_P = \frac{\eta}{P}, \quad P < 1; \tag{2.3}$$

$$\frac{1}{\gamma_P} - 1 = \left(\frac{1}{\gamma} - 1\right)\frac{1}{P^2}, \quad P < 1; \tag{2.4}$$

Figure 2 plots the relation of γ vs. P for each given γ_P in Formula (2.4). When γ_P is less than a predefined maximal threshold (e.g., 2%, 5% or 10%), the two values can be deemed virtually uncorrelated of each other at the level of uncertainty. If the two values are independent of each other at their uncertainty levels, their uncertainties are uncorrelated of each other. Thus for each independence standard γ_P, there is a maximal allowed correlation between two values below which the uncorrelated uncertainty assumption of precision arithmetic holds. The maximal allowed correlation is a function of the larger precision of the two values according to Formula (2.4). Figure 2 shows that for two precisely measured values, their correlation γ is allowed to be quite high. To be acceptable in precision arithmetic, each of the low-resolution values should contain enough noise in its uncertainty, so that they do not have much correction through the systematic error [1][2]. Thus, the uncertainty assumption uncertainty assumption has much weaker statistical requirement than the assumption for independence arithmetic, which requires the two values to be independent of each other.

It is tempting to add noise to otherwise unqualified values to make their uncertainties uncertainty assumption of each other. As an extreme case of this approach, if two values are constructed by adding noise to the same signal, they are 50% correlated at the uncertainty level so that they will not satisfy the uncorrelated uncertainty assumption[7].

2.3 Precision Representation and Precision Round Up Rule

Let the content of a floating-point number be denoted as $S2^E$, in which S is the significand[8] and E is the exponent of 2 of the floating-point number. In addition, the precision representation $S{\sim}2^E$ contains a carry $\sim$ to indicate its rounding error, which can be:

- +: The rounding error is positive;
- -: The rounding error is negative;
- ?: The sign of the rounding error is unknown;
- #: The precision value contains an error code. Each error code is generated due to a specific illegal arithmetic operation such as dividing by zero. An operand error code is directly transferred to the operation result. In this way, illegal operations can be traced back to the source.

Because there is only limited bits to hold S, a *round up* is needed, which proceeds according to the following round up rule:

- A value of $(2S){\sim}2^E$ is rounded up to $S{\sim}2^{E+1}$.
- A value of $(2S+1){+}2^E$ is rounded up to $(S+1){-}2^{E+1}$.
- A value of $(2S+1){-}2^E$ is rounded up to $S{+}2^{E+1}$.
- A value of $(2S+1)?2^E$ is rounded up to $(S+1){-}2^{E+1}$.

Let the value before any rounding up be the original value, the round-up rule ensures that $S2^E$ is always the closest value with exponent E to the original value. After each round up, the original rounding error is reduced by half for the new significand. If the original significand is odd, the round up generates a new rounding error of 1/2, which is added to the existing rounding error. Since the newly generated rounding

[7]The 50% curve in Figure 2 thus defines the maximal possible correlations between any two measured signals. This other conclusion of Formula (2.4) makes sense because the measurable correlation between two measurements should be limited by the precisions of their measurements.

[8]While "significand" is the official word [10] to describe "The component of a binary floating-point number that consists of an explicit or implicit leading bit to the left of its implied binary point and a fraction field to the right", "mantissa" is often unofficially used instead.

error always cancels the existing rounding error, the rounding error range is limited to half bit of the significand, or the bounding range for the rounding error is [-1/2,+1/2]. The precision arithmetic also tracks the rounding error bounding range R so that the precision representation becomes $S{\sim}R\,2^E$.

If the initial $\sim$ is wrong, it will be corrected by the first round up when S is odd, or R will be reduced to half after each round up when S is even. Hence, the precision round up process is stable and self-correcting.

2.4 Probability Distribution of Rounding Errors

An ideal floating-point calculation is carried out conceptually to infinitesimal precision before it is rounded up to representation precision [10][13][15]. Thus, rounding up should be a process independent of any calculation, and it should be evaluated separately. To estimate the rounding error distribution within its bounding range [-1/2, +1/2], a large number[9] of positive random integers are converted into precision values and then rounded up once at a step time until each of them has a significand smaller than a predefined minimal significand threshold. The precision value at each step is compared with the original value for the rounding error. Figure 3 shows the result histogram of rounding errors for the minimal significand thresholds 0, 1, 4 and 16, respectively. When each significand bit has an equal chance to be either 0 or 1, the result distribution of the rounding errors is expected to be uniformly distributed within the range [-1/2, +1/2] [38]. However, the precision round up rule changes this equal chance for a few lowest digits of a significand. So when the minimal significand threshold is smaller, the bias in rounding error distribution is larger, as shown in Figure 3, and the result distribution is close to uniform only when the minimal significand threshold is 4 and above.

2.5 Result Uncertainty For Addition and Subtraction

In floating-point arithmetic, rounding errors are uncertainties [10][13][15]. The precision round-up rule incorporates all randomness of an imprecise value into its carry and bounding range so that it preserves the uncorrelated uncertainty assumption between any two values. The uncorrelated uncertainty assumption suggests that the result rounding error distribution of addition is the convolution of the two operand rounding error distributions, while the result rounding error distribution of subtraction is the convolution of the first operand rounding error distribution and the mirror image of the second operand rounding error distribution [4]. Thus, when the exponents of two operands are equal, the results of addition and subtraction are:

$$S_1{\sim_1}R_1 2^E \pm S_2{\sim_2}R_2 2^E = (S_1 \pm S_2){\sim}(R_1 + R_2)2^E; \tag{2.5}$$

Table 1 shows the result $\sim$ for addition, while Table 2 shows the result $\sim$ for subtraction. It will be shown that the $\sim$ immediately after a calculation is actually not important because the precision round up rule frequently is applied after each calculation in precision arithmetic as its normalization process.

[9]For each minimal significand threshold, 64K random integers are used. The actual number of random integers is not important as far as 1) it gives a stable empirical histogram, and 2) the random integers are uniformly distributed without repeat in values.

$\sim_1$ vs. $\sim_2$	$\sim_1 = \sim_2$	$\sim_1 \neq \sim_2$				
		$\sim_1 = ?$	$\sim_2 = ?$	$R_1 > R_2$	$R_1 < R_2$	$R_1 = R_2$
$\sim$	$\sim_1$	$\sim_2$	$\sim_1$	$\sim_1$	$\sim_2$	?

Table 1: Result $\sim$ in $S_1 \sim_1 R_1 2^E + S_2 \sim_2 R_2 2^E = (S_1 + S_2) \sim (R_1 + R_2) 2^E$

$\sim_1$ vs. $\sim_2$	$\sim_1 = \sim_2$				$\sim_1 \neq \sim_2$	
	$\sim_1 = ?$	$R_1 > R_2$	$R_1 < R_2$	$R_1 = R_2$	$\sim_1 = ?$	$\sim_1 \neq ?$
$\sim$	?	$\sim_1$	$-\sim_2$	?	$-\sim_2$	$\sim_1$

Table 2: Result $\sim$ in $S_1 \sim_1 R_1 2^E - S_2 \sim_2 R_2 2^E = (S_1 - S_2) \sim (R_1 + R_2) 2^E$.

Let $P_{\frac{1}{2}}(x)$ be the rounding error distribution after rounding up, which is uniformly distributed between [-1/2, +1/2] according to Formula (2.6). Let $P_{\frac{n}{2}}(x)$ be the convolution of $P_{\frac{1}{2}}(x)$ according to Formula (2.7):

$$P_{\frac{1}{2}}(x) \equiv 1, \quad -1/2 \le x \le +1/2; \tag{2.6}$$

$$P_{\frac{n}{2}}(x) \equiv \int_{-\infty}^{+\infty} P_{\frac{1}{2}}(y) P_{\frac{n-1}{2}}(x-y) dy = \int_{-1/2}^{+1/2} P_{\frac{n-1}{2}}(x-y) dy, \quad n = 2, 3, 4 \dots; \tag{2.7}$$

Formula (2.7) shows that $P_{\frac{n}{2}}(x)$ has a bounding range of $R \equiv \frac{n}{2}$, in which case it is easy to prove that the deviation σ of $P_{\frac{n}{2}}(x)$ is determined by its bounding range R:

$$\sigma^2 = R/6; \tag{2.8}$$

Also, the same bounding range can be reached in any combination:

$$P_{\frac{m+n}{2}}(x) = \int_{-\infty}^{+\infty} P_{\frac{m}{2}}(y) P_{\frac{n}{2}}(x-y) dy; \tag{2.9}$$

In reality, $P_{\frac{1}{2}}(x)$ is not strictly uniformly distributed in its bounding range [-1/2, +1/2]. As the worst case, let $P_{1/2}(x)$ be the rounding error distribution with the minimal significand threshold of 0 in Figure 3. Figure 4 shows the rounding error distribution after addition and subtraction, in which:

- R=1/2: "1" for no addition or subtraction.
- R=2/2: "1+1" for addition once, and "1-1" for subtraction once.
- R=3/2: "1+1+1" for addition twice, "1-1-1" for subtraction twice, "1+1-1" for addition once then subtraction once, and "1-1+1" for subtraction once then addition once.

Figure 4 shows that the rounding error distributions for the same bounding range largely repeat each other, confirming Formula (2.9). Addition and subtraction have a slightly different result distribution due to uneven $P_{\frac{1}{2}}(x)$. In all cases, the distributions quickly approach Gaussian with the increase of bounding ranges.

Even for the worst-case $P_{\frac{1}{2}}(x)$, the deviation σ relates to the bounding range R empirically as $\sigma = 0.423005 R^{0.50000}$ with a reliable factor of 0.9999999, confirming Formula (2.8) empirically.

The Lyapunov form of the central limit theorem [4] states that if X_i is a random variable with mean μ_i and variance σ_i^2 for each i among a series of n mutually independent random variables, then with increased n, the sum $\sum_i^n X_i$ converges in distribution to the Gaussian distribution with mean $\sum_i^n \mu_i$ and variance $\sum_i^n \sigma_i^2$. Applying the central limit theorem to precision arithmetic when a calculation ends with summation:

- $P_{n/2}(x)$ converges in distribution to a Gaussian distribution of mean 0 and deviation σ with an increased n
- Figure 4 shows that such convergence to Gaussian distribution is very quick.
- The stable rounding error distribution is *independent* of any initial rounding error distribution, so that we can extend the rounding error distribution to uncertainty distribution in general.

Because of rounding, $P_{\frac{n}{2}}(x)$ is extended to $P_R(x)$ for 2's fractional R, which is further extended to characterize uncertainty distribution in general.

2.6 Uncertainty Distribution

The probability density function $D_y(y)$ after linear transformation ($y = \alpha x + \beta$) of a generic probability density function $D_x(x)$ is [4]:

$$D_y(y) = D_x((y - \beta)/\alpha)/\alpha; \tag{2.10}$$

Letting $N(x)$ be the density function of a normal distribution, the density function of uncertainty distribution in precision arithmetic is thus:

$$\rho(y) \equiv N(y/\sigma)/\sigma, \quad y \in [-R, +R]; \tag{2.11}$$

The deviation δx and bounding range Δx of $S{\sim}R\ 2^E$ is:

$$\delta x = \sigma \times 2^E, \quad \sigma < \sqrt{\hat{R}/6}; \tag{2.12}$$

$$\Delta x = R \times 2^E, \quad R < \hat{R}; \tag{2.13}$$

$$\Delta x/\delta x = R/\sigma = \sqrt{6R}; \tag{2.14}$$

Using δx and Δx, Formula (2.11) becomes the probability density function $\rho(\widetilde{y})$ in Formula (2.15):

$$\rho(\widetilde{y}) = N(\widetilde{y}/\delta x)/\delta x, \quad \widetilde{y} \in [-\Delta x, +\Delta x]; \tag{2.15}$$

And the precision representation $S{\sim}R\ 2^E$ is interpreted as:.

$$x \pm \delta x = x + \widetilde{y}, \quad \widetilde{y} \in \rho(\widetilde{y}); \tag{2.16}$$

2.7 Uncertainty Rounding and Normalization

According to Formula (2.10), when an imprecise value is rounded up once, its density function becomes $N(y/\frac{\sigma}{2})/\frac{\sigma}{2}, y \in [-\frac{R}{2}, +\frac{R}{2}]$, however the two ways to carry out rounding up can not reach this new distribution ideally:

- By range: When R is reduced to 1/2-fold, σ is reduced to $1/\sqrt{2}$-fold. The new $\rho(y)$ becomes $N(y/\frac{\sigma}{\sqrt{2}})/\frac{\sigma}{\sqrt{2}}, y \in [-\frac{R}{2}, +\frac{R}{2}]$. This approach distorts the probability but retains strict bounding, which aligns with interval arithmetic.

- By deviation: When σ is reduced to 1/2-fold, R is reduced to 1/4-fold. The new $\rho(y)$ becomes $N(y/\frac{\sigma}{2})/\frac{\sigma}{2}, y \in [-\frac{R}{4}, +\frac{R}{4}]$. This approach ignores the probability distribution on the Gaussian tails outside $[-\frac{R}{4}, +\frac{R}{4}]$, but preserves the overall characteristics of the distribution.

Figure 5 compares these two ways of rounding up when the original rounding error range is R=8, in which R=4 is rounded up by range, while R=2 is rounded up by deviation. It clearly shows that rounding up by deviation results in a more similar rounding error distribution. Rounding up by deviation is required by the scaling principle, so it is used universally in precision arithmetic. Rounding up by deviation also introduces bounding leakage called *round-up leakage*. In Figure 5, the 8/2 distribution of the rounding error outside the range [-2, +2] contributes to a round-up leakage of 0.05%.

Because the tail of the Gaussian distribution deceases by e^{-6Ry^2}, the round-up leakage decreases exponentially with the increased bounding range R. Smaller round-up leakage also means that the actual rounding error distribution becomes more similar to the rounding error distribution with increased bounding range R. When R is above a threshold $\hat{R}$, round-up leakage is small enough so that rounding up by deviation can be applied repeatedly. This is the *normalization* process in precision arithmetic. When $\hat{R} = 16$, the maximal normalization leakage is 10^{-6}, which is small enough for most applications, and which has a comparable bounding range as the de facto $6\sigma - 10^{-9}$ rule for negligible bounding leakages in statistics. In addition to limit the bit count for both R and S, normalization also enforce the correctness of carry sign $\sim$ in precision representation $S{\sim}R\ 2^E$.

Because the precision round up rule only looks at the sign of the current rounding error, rounding up by either deviation or range will not change the rounding error distribution.

As an inverse operation to rounding up by deviation, a precision round-down rule is defined using the scaling principle. After rounding down once, $S{\sim}R\ 2^E$ becomes $(2S){\sim}(4R)2E-1$. Round down reduces bounding leakage. To add or subtract two operands with different exponents, the operand with a larger exponent is first rounded down to the other exponent, and the result of addition or subtraction using Formula (2.5) is normalized afterwards.

2.8 Uncertainty Initiation

An integer S is initialized as $S2^0$.

A conventional 64-bit floating-point value $S2^E$ is usually initialized as $S?\frac{1}{2}2^E$ because the IEEE floating-point standard [10] guarantees accuracy to half bit of a significand.

A mean-deviation pair $(x \pm \delta x)$ of 64-bit conventional floating-point values is initialized as $S{\sim}R\ 2^E$ by:

1. rounding up δx until Formula (2.12) is satisfied;
2. obtaining R and E from final δx;
3. rounding up x to E; and
4. obtaining S and $\sim$ from final x.

2.9 Result Uncertainty For Multiplication

After $S{\sim}R\ 2^E$ is multiplied by 2, both its range and deviation increase by 2-fold. If the scaling principle is applied, the result is $2S{\sim}2^2R\ 2^E$. When $S{\sim}R\ 2^E$ is normalized, the result is then normalized as $S{\sim}R\ 2^{E+1}$, and there is neither bound widening nor bounding leakage. Generally, the direct result of multiplying $S_1{\sim}_1R_1 2^{E_1}$ by $S_2 2^{E_2}$ is:

$$S_1{\sim}_1R_1 2^{E_1} \times S_2 2^{E_2} = (S_1S_2){\sim}_1({S_2}^2R_1)2^{E_1+E_2}; \tag{2.17}$$

Because Formula (2.17) obeys scaling principle, the result uncertainty is still ρ-distributed.

According to uncorrelated uncertainty assumption, the product bounding range of multiplying $0{\sim}_1R_1 2^{E_1}$ by $0{\sim}_2R_2 2^{E_2}$ is $R_1R_2 2^{E_1+E_2}$. Thus:

$$\begin{aligned} S_1{\sim}_1R_1 2^{E_1} \times S_2{\sim}_2R_2 2^{E_2} = (S_1S_2)2^{E_1+E_2} + 0({\sim}_1{\sim}_2)(R_1R_2)2^{E_1+E_2} \\ + 0{\sim}_1({S_2}^2R_1)2^{E_1+E_2} + 0{\sim}_2({S_1}^2R_2)2^{E_1+E_2}; \end{aligned} \tag{2.18}$$

In Formula (2.18), both $0{\sim}_1({S_2}^2R_1)2^{E_1+E_2}$ and $0{\sim}_2({S_1}^2R_2)2^{E_1+E_2}$ are ρ-distributed. The probability density function for $0({\sim}_1{\sim}_2)(R_1R_2)2^{E_1+E_2}$ is calculated as:

$$\rho_{mul}(x) = \frac{d}{dx}\int_{x_1\times x_2<x} \frac{1}{\sqrt{2\pi}\delta x_1}e^{-\frac{x_1^2}{2(\delta x_1)^2}}\frac{1}{\sqrt{2\pi}\delta x_2}e^{-\frac{x_2^2}{2(\delta x_2)^2}}dx_1dx_2; \tag{2.19}$$

Letting $\delta x \equiv (\delta x_1)(\delta x_2)$, $y_1 \equiv x_1/(\delta x_1)$, $y_2 \equiv x_2/(\delta x_2)$ and $y \equiv x/(\delta x)$, Formula (2.19) is simplified as:

$$\rho_{mul}(x) = \frac{1}{2\pi\delta x}\frac{d}{dy}\int_{y_1\times y_2<y} e^{-\frac{y_1^2+y_2^2}{2}}dy_1dy_2; \tag{2.20}$$

Using polar coordinate (r,θ) instead Cartesian coordinate (y_1,y_2), Formula (2.20) is simplified as:

$$\begin{aligned} \rho_{mul}(x) &= \frac{1}{2\pi\delta x}\frac{d}{dy}\int_0^{2\pi}d\theta\int_0^{\frac{y}{2\sin(\theta)\cos(\theta)}} e^{-\frac{r^2}{2}}d\frac{r^2}{2} \\ &= \frac{1}{2\pi\delta x}\frac{d}{dy}\int_0^{2\pi}d\theta\left(1-e^{-\frac{y}{\sin(2\theta)}}\right) = \frac{1}{2\pi\delta x}\int_0^{\pi}\frac{e^{-\frac{|y|}{\sin(\theta)}}}{\sin(\theta)}d\theta; \end{aligned} \tag{2.21}$$

Similarly, letting $\delta x \equiv \sqrt{(\delta x_1)^2+(\delta x_2)^2}$, $y_1 \equiv x_1/(\delta x_1)$, $y_2 \equiv x_2/(\delta x_2)$ and $y \equiv x/\delta x$, the distribution for $0{\sim}R_1 2^E + 0{\sim}R_2 2^E = 0{\sim}(R_1+R_2)2^E$ is calculated as:

$$\begin{aligned} \rho_{add}(x) &= \frac{1}{2\pi\delta x}\frac{d}{dy}\int_0^{2\pi}d\theta\int_0^{\frac{y}{\sin(\theta)+\cos(\theta)}} e^{-\frac{r^2}{2}}rdr \\ &= \frac{1}{2\pi\delta x}\frac{d}{dy}\int_0^{2\pi}d\theta\left(1-e^{-\frac{y^2}{2}\frac{1}{1+\sin(2\theta)}}\right) = \frac{1}{2\pi\delta x}\int_0^{2\pi}d\theta|y|\frac{e^{-\frac{y^2}{2}\frac{1}{1+\sin(2\theta)}}}{1+\sin(2\theta)}; \end{aligned} \tag{2.22}$$

The result of Formula (2.22) is known as Formula (2.23). Because $2\sin(\theta)|_{\theta\in[0,\pi]}$ almost repeats $1+\sin(\theta)|_{\theta\in[-\frac{\pi}{2},\frac{3\pi}{2}]}$, the result of Formula (2.21) is estimated as Formula (2.24) and Formula (2.25):

$$y = \frac{x}{\sqrt{{\delta x_1}^2 + {\delta x_2}^2}} : \quad \rho_{add}(x) = \frac{1}{\sqrt{2\pi}\sqrt{{\delta x_1}^2 + {\delta x_2}^2}} e^{-\frac{y^2}{2}}; \tag{2.23}$$

$$y = \frac{x}{\delta x_1 \delta x_2} : \quad \rho_{mul}(x) = \frac{1}{\sqrt{2\pi}(\delta x_1 \delta x_2)} \frac{e^{-2|y|}}{\sqrt{|y|}}; \tag{2.24}$$

$$z \equiv 2\sqrt{y} = 2\sqrt{\frac{x}{\delta x_1 \delta x_2}} : \quad \rho_{mul}(x) = \frac{1}{\sqrt{2\pi}\sqrt{\delta x_1 \delta x_2}} e^{-\frac{z^2}{2}}; \tag{2.25}$$

Formula (2.25) shows that $0(\sim_1 \sim_2) R_1 R_2 @ (E_1 + E_2)$ is ρ-distributed with deviation $\sqrt{(\delta x_1)(\delta x_2)}$ or range $R_1 R_2$, which adds to the rest two ρ-distributed terms of Formula (2.18). Thus, the result of multiplication is also ρ-distributed.

The result precision P of multiplication is:

$$P^2 = \frac{R_1 R_2 + {S_2}^2 R_1 + {S_1}^2 R_2}{(S_1 S_2)^2} = {P_1}^2 + {P_2}^2 + \frac{1}{6}{P_1}^2 {P_2}^2; \tag{2.26}$$

Formula (2.26) shows that precision cannot be improved during multiplication. It is identical to Formula (1.12) except their cross term, representing difference in their statistical requirements.

2.10 Result Uncertainty For Division

The reverse of Formula (2.17) defines:

$$\frac{S_1 \sim_1 R_1 2^{E_1}}{S_2 2^{E_2}} = \frac{S_1}{S_2} \sim_1 \frac{R_1}{{S_2}^2} 2^{E_1 - E_2} \tag{2.27}$$

In Formula (2.27), the rounding error decreases by S_2-fold, but the bounding range decreases by ${S_2}^2$-fold, so there is bounding leakage. To limit the bounding leakage to acceptable level, the result is rounded down until normalized. In this case, rounding down the direct result is equivalent to round down the dividend $S_1 \sim R_1 2^{E_1}$ by the same amount before the division.

Using the methodology defined in [4], the probability density function for $1/(x+\delta x)$ is calculated as:

$$|\frac{1}{y} - x| < \Delta x : \quad \rho_{inv}(y) = \frac{1}{\sqrt{2\pi}\delta x} \frac{e^{-\frac{(\frac{1}{y} - x)^2}{2(\delta x)^2}}}{y^2}; \tag{2.28}$$

Letting $z \equiv (1/y - x)/\delta x$, whose range is $\Delta x / \delta x = \sqrt{6R}$ according to Formula (2.14), the probability density function for $1/(x + \delta x)$ becomes:

$$|z| < \sqrt{6R} : \quad \rho_{inv}(\frac{1}{z\delta x + x}) = \frac{1}{\sqrt{2\pi}\delta x} e^{\frac{-z^2}{2}} \tag{2.29}$$

Figure 6 shows the probability density function $p(x)$ for $1/(x + \delta x)$ in which $\delta x = 1$:

- when $x = 0$: $p(\infty) = N(0)$, in which $N(x)$ is normal distribution, which is also displayed in the figure. This distribution has no moment defined, thus it has no mean, deviation and etc.
- when $x = 2$: $p(\infty) = N(2)$. This distribution has mode at $x = 1/2$.

- when $x = 5$: $p(\infty) = N(5)$. This distribution has mode at $x = 1/5$. It has finer precision and it looks closer to Gaussian than the previous case.

The mean for $1/(x + \delta x)$ is calculated as:

$$\mu(\frac{1}{x}) = \frac{1}{\sqrt{2\pi}\delta x} \int_{|\frac{1}{y} - x| < \Delta x} y \rho_{inv}(y) dy = \frac{1}{x\sqrt{2\pi}} \int_{-\frac{\Delta x}{\delta x}}^{+\frac{\Delta x}{\delta x}} \frac{e^{-\frac{z^2}{2}}}{zP(x) + 1} dz; \tag{2.30}$$

Formula (2.30) diverges at $z = -1/P(x)$, which is excluded from the integration range when $\Delta x \ll |x|$. In such a case, replacing $\Delta x/\delta x$ with $\bar{R} \equiv \sqrt{6\hat{R}}$ for the integration range, the mean is calculated as:

$$M(j) \equiv \frac{1}{\sqrt{2\pi}} \int_{-\bar{R}}^{+\bar{R}} z^j e^{-\frac{z^2}{2}} dz; \tag{2.31}$$

$$\mu(\frac{1}{x}) = \frac{1}{x} \sum_{j=0}^{+\infty} P(x)^j M(j); \tag{2.32}$$

$\hat{R}$ has to be small enough so that Formula (2.32) can converge for non-zero $P(x)$.

The moment for uncertainty distribution is generally defined in Formula (2.31), which is 0 when j is odd. $M(2j)$ is calculated as:

$$\begin{aligned} M(2j) &= (2j-1)M(2j-2) - \frac{2}{\sqrt{2\pi}} \bar{R}^{2j-1} e^{-\frac{\bar{R}^2}{2}} \\ &= (2j-1)!!(M(0) - \frac{2}{\sqrt{2\pi}} e^{-\frac{\bar{R}^2}{2}} \sum_{k=1}^{j} \frac{\bar{R}^{2k-1}}{(2k-1)!!}); \end{aligned} \tag{2.33}$$

Formula (2.34) [39] shows that $M(2j)$ eventually becomes 0 for large enough j:

$$\begin{aligned} &\lim_{j\to\infty} \sum_{k=1}^{j} \frac{\bar{R}^{2k-1}}{(2k-1)!!} = e^{\frac{\bar{R}^2}{2}} \int_{0}^{+\bar{R}} e^{-\frac{z^2}{2}} dz; \\ &\lim_{j\to\infty} M(2j) = (2j-1)!!(M(0) - M(0)) = 0; \end{aligned} \tag{2.34}$$

Unfortunately, such convergence to 0 is very slow, and lower orders of $M(2j)$ actually increases exponentially according to Figure 7. Empirically:

$$j < 400: \quad M(2j) = (2.4429\hat{R}^{0.4998})^{2j}; \tag{2.35}$$

Specifically, $\hat{R} = 4 : M(2j) \sim 4.8844^{2j}$. When $P(x) < 1/4.8844 = 0.20473$, $1/(x + \delta x)$ converges so that inversion can be defined statistically:

$$\mu(\frac{1}{x}) = \frac{1}{x}(1 + P(x)^2 + 3P(x)^4 + 15P(x)^6 + ...); \tag{2.36}$$

However, Formula (2.36) does not obey the recovering principle, so that a better approach is to let the value of inversion to be $1/x$ and the inversion deviation defined around $1/x$, as:

$$\begin{aligned} (\delta\frac{1}{x})^2 &\equiv \frac{1}{\sqrt{2\pi}\delta x} \int_{|\frac{1}{y} - x| < \Delta x} (y - \frac{1}{x})^2 \rho_{inv}(y) dy \\ &= \frac{1}{x^2\sqrt{2\pi}} \int_{-\bar{R}}^{+\bar{R}} e^{-\frac{z^2}{2}} (\frac{1}{zP(x) + 1} - 1)^2 dz; \\ P(\frac{1}{x})^2 &= \sum_{j=1}^{+\infty} P(x)^{2j} M(2j)(2j-1) = P(x)^2 + 9P(x)^4 + 75P(x)^6 + ...; \end{aligned} \tag{2.37}$$

An equivalent way to calculate Formula (2.37) is to use Taylor expansion around $1/x$ in $\rho(y)$ space instead of in $\rho_{inv}(y)$ space:

$$\begin{aligned}(\delta\frac{1}{x})^2 &\equiv \frac{1}{\sqrt{2\pi}\delta x}\int_{|\frac{1}{y}-x|<\Delta x}(y-\frac{1}{x})^2\rho(y)_{inv}dy \\ &= \frac{1}{\sqrt{2\pi}\delta x}\int_{|\frac{1}{y}-x|<\Delta x}(\frac{1}{y}-\frac{1}{x})^2\rho(y)dy = \frac{1}{x^2\sqrt{2\pi}}\int_{-\bar{R}}^{+\bar{R}}e^{-\frac{z^2}{2}}(\frac{1}{zP(x)+1}-1)^2dz;\end{aligned} \tag{2.38}$$

Formula (2.37) obeys the recovering principle for uncertainty only approximately when $P(x)^2 \ll 1$.

The sign of the representation $S{\sim}R\,2^E$ becomes negation of itself after inversion.

Dividing an operand by itself results in precise 1.

2.11 Result Uncertainty For Square and Square Root

A special case of multiplication is between an operand and itself. If x is $N(x)$ distributed, x^2 is χ^2-distributed with freedom 1 [4], which has mean 1 and variance of 2. $N(x)$ and $\chi^2(x)$ have quite different characteristics, e.g., $\chi^2(x)$ only roughly resembles half of $N(x)$. The bounding goal of precision arithmetic extends χ^2 distribution to the other side of the mathematically expected value, and absorb the square of mean of the χ^2 distribution into final variance:

$$(S_1{\sim}R_1 2^{E_1})^2 \equiv {S_1}^2 2^{2E_1} + 0{\sim}4R_1{S_1}^2 2^{2E_1} + 0{\sim}3{R_1}^2 2^{2E_1}; \tag{2.39}$$

The result precision of square is:

$$P^2 = \frac{4R_1{S_1}^2 + 3{R_1}^2}{{S_1}^4} = 4{P_1}^2 + 3{P_1}^4; \tag{2.40}$$

Similar to Formula (2.38), with $z = (y-x)/\delta x$ the deviation for x^2 is calculated as Formula (2.41), which confirms Formula (2.40):

$$\begin{aligned}(\delta x^2)^2 &= \frac{1}{\sqrt{2\pi}\delta x}\int_{|y^2-x|<\Delta x}(y^2-x^2)^2\rho(y)dy \\ &= \frac{x^4}{\sqrt{2\pi}}\int_{-\bar{R}}^{+\bar{R}}e^{-\frac{z^2}{2}}((1+zP(x))^2-1)^2dz; \\ P(x^2)^2 &= 4M(2)P(x)^2 + 4M(3)P(x)^3 + M(4)P(x)^4 = 4P(x)^2 + 3P(x)^4;\end{aligned} \tag{2.41}$$

And the deviation for $\sqrt{x}$ is calculated as Formula (2.42).

$$\begin{aligned}(\delta\sqrt{x})^2 &= \frac{1}{\sqrt{2\pi}\delta x}\int_{|\sqrt{y}-x|<\Delta x}(\sqrt{y}-\sqrt{x})^2\rho(y)dy \\ &= \frac{|x|}{\sqrt{2\pi}}\int_{-\bar{R}}^{+\bar{R}}e^{-\frac{z^2}{2}}(\sqrt{1+zP(x)}-1)^2dz \\ &= \frac{|x|}{\sqrt{2\pi}}\int_{-\bar{R}}^{+\bar{R}}e^{-\frac{z^2}{2}}(\frac{zP(x)}{2}-\frac{z^2P(x)^2}{8}+\frac{z^3P(x)^3}{16}+...)^2dz; \\ P(\sqrt{x})^2 &= \frac{1}{4}P(x)^2 + \frac{15}{64}P(x)^4 + ...;\end{aligned} \tag{2.42}$$

The above Taylor-expansion method can be extended to calculate power x^n and root $\sqrt[n]{x}$ for any integer n.

The combination of Formula (2.41) and Formula (2.42) as Formula (2.43) shows again that the uncertainty deviation obeys the recovery principle only approximately when $P(x)^2 \ll 1$. Furthermore, the result of $\sqrt{x^2}$ and $(\sqrt{x})^2$ are different in Formula (2.43), showing dependency problem. The reason why the uncertainty deviation can not obey the recovery principle strictly is not clear at this moment.

$$\begin{aligned} P(\sqrt{x^2})^2 &= \frac{1}{4}(4P(x)^2 + 3P(x)^4) + \frac{15}{64}(4P(x)^2 + 3P(x)^4) + ... \\ &= P(x)^2 + \frac{9}{2}P(x)^4 + ...; \\ P((\sqrt{x})^2)^2 &= 4(\frac{1}{4}P(x)^2 + \frac{15}{64}P(x)^4 + ...) + 3(\frac{1}{4}P(x)^2 + \frac{15}{64}P(x)^4 + ...)^2; \\ &= P(x)^2 + \frac{9}{8}P(x)^4 + ...; \end{aligned} \tag{2.43}$$

2.12 Result Uncertainty For Function Evaluation

Extending Formula (2.38), the uncertainty of the function $f(x)$ at $(x \pm \delta x)$ is evaluated by the set $\{f(x+\widetilde{y}) - f(x)\}$, as Formula (2.44), in which $\widetilde{y}$ and $\rho(\widetilde{y})$ are defined by Formula (2.15).

$$(\delta f)^2 \equiv \int (f(x+\widetilde{y}) - f(x))^2 \rho(\widetilde{y})\, d\widetilde{y}; \tag{2.44}$$

For example, Formula (2.45) provides uncertainty for generic polynomial to J-th order.

$$\begin{aligned} C_j^k &\equiv \frac{j!}{k!(j-k)!}; \\ (\delta \sum_{j=0}^{J} a_j x^j)^2 &= \int (\sum_{j=0}^{J} a_j (x+\widetilde{y})^j - \sum_{j=0}^{J} a_j x^j)^2 \rho(\widetilde{y}) d\widetilde{y} \\ &= \int (\sum_{j=1}^{J} a_j \sum_{k=1}^{j} C_j^k \widetilde{y}^k x^{j-k})^2 \rho(\widetilde{y}) d\widetilde{y} \\ &= \sum_{j_1=1}^{J} \sum_{j_2=1}^{J} a_{j_1} a_{j_2} \sum_{k_1=1}^{j_1} \sum_{k_2=1}^{j_2} C_{j_1}^{k_1} C_{j_2}^{k_2} M(k_1+k_2)(\delta x)^{k_1+k_2} x^{j_1+j_2-k_1-k_2}; \end{aligned} \tag{2.45}$$

If the function $f(x)$ is Taylor expandable at x, $f(x+\widetilde{y}) - f(x)$ is calculated according to Formula (2.46), in which $f^{(n)}$ denotes the nth derivatives of $f(x)$ at x. $(\delta f)^2$ is given by Formula (2.47), in which $M(j)$ is defined by Formula (2.31) and (2.31).

$$f(x+\widetilde{y}) - f(x) = \sum_{n=1}^{\infty} \frac{f^{(n)}(x)}{n!} \widetilde{y}^n; \tag{2.46}$$

$$(\delta f)^2 = \sum_{n=0}^{\infty} \sum_{j=0}^{\infty} \frac{f^{(n)}(x)}{n!} \frac{f^{(j)}(x)}{j!} (\delta x)^{n+j} M(n+j) - f(x)^2; \tag{2.47}$$

Let $P(f(x)) \equiv \delta f(x)/|f(x)|$ be defined as the precision for $f(x)$; and let α be a constant. According to Formula (2.47):

$$\delta(\alpha x) = \alpha(\delta x); \quad P(\alpha x) = P(x); \tag{2.48}$$

$$(\delta\frac{1}{x})^2 = \frac{1}{x^2}\sum_{j=1}^{\infty}(2j-1)M(2j)P(x)^{2j}; \quad P(\frac{1}{x})^2 = \sum_{j=1}^{\infty}(2j-1)M(2j)P(x)^{2j}; \tag{2.49}$$

$$(\delta x^2)^2 = 4x^2(\delta x)^2 + 3(\delta x)^4; \quad P(x^2)^2 = 4P(x)^2 + 3P(x)^4; \tag{2.50}$$

Formula (2.48) confirm the scaling principle, while Formula (2.50) and (2.49) confirm Formula (2.40) and (2.37), respectively.

Due to uncorrelated uncertainty assumption, Taylor expansion can also be used to find the result $(\delta f)^2$ of the function $f(x_1, x_2)$, in which $f^{(m,n)}$ denotes the mth and nth partial derivatives of x_1 and x_2, respectively; and the uncorrelated uncertainty assumption between x_1 and x_2 leads to independence between the random variables $\widetilde{y}_1$ and $\widetilde{y}_2$ in Formula (2.52):

$$f(x_1+\widetilde{y}_1, x_2+\widetilde{y}_2) - f(x_1,x_2) = \sum_{m=0}^{\infty}\sum_{n=0}^{\infty}\frac{f^{(m,n)}}{m!n!}\widetilde{y}_1^m\widetilde{y}_2^n - f(x_1,x_2); \tag{2.51}$$

$$(\delta f)^2 = \sum_{m=0}^{\infty}\sum_{n=0}^{\infty}\sum_{i=0}^{\infty}\sum_{j=0}^{\infty}\frac{f^{(m,n)}}{m!n!}\frac{f^{(i,j)}}{i!j!}(\delta x_1)^{m+i}(\delta x_2)^{n+j}M(m+i)M(n+j) - f(x_1,x_2)^2; \tag{2.52}$$

Such an approach can be extended to a function of an arbitrary number of input variables. Formula (2.52) shows that an input contributes to the result uncertainty in more than one way, in the same way as Δx appears in more than one term in the Taylor expansion of $\Delta f(x, y, z)$.

According to Formula (2.52):

$$\delta(x_1 \pm x_2)^2 = (\delta x_1)^2 + (\delta x_2)^2; \tag{2.53}$$

$$\delta(x_1 \times x_2)^2 = (\delta x_1)^2 {x_2}^2 + {x_1}^2(\delta x_2)^2 + (\delta x_1)^2(\delta x_2)^2; \tag{2.54}$$

$$P(x_1 \times x_2)^2 = P(x_1)^2 + P(x_2)^2 + \frac{1}{6}P(x_1)^2 P(x_2)^2; \tag{2.55}$$

$$\delta(\frac{x_1}{x_2})^2 = \frac{(\delta x_1)^2}{{x_2}^2} + {x_1}^2(\delta\frac{1}{x_2})^2 + (\delta x_1)^2(\delta\frac{1}{x_2})^2 \simeq \frac{\delta(x_1 \times x_2)^2}{{x_2}^4}; \tag{2.56}$$

Formula (2.53) and (2.55) confirm Formula (2.5) and (2.26), respectively.

If $f(x)$ is a black-box function, because the normal distribution $N(x)$ is well known, standard methods exist to divide the range $[x - \Delta x, x + \Delta x]$ into equal-probability quantiles [4], and δf can be found numerically by sampling. For example, a κ-point monotonic sampling requires that 1) each numerically monotonic region contains at least κ consecutive sampling points; and 2) the whole range $[x - \Delta x, x + \Delta x]$ has been divided into monotonic regions only. When κ is sufficiently large, the chance of missing a peak or a valley is small enough so that the sampling is fair enough. The range $[x - \Delta x, x + \Delta x]$ is first divided into κ equal-probability quantiles, and at each additional step, each quantile is further divided into an equal number of sub quantiles until both sampling requirements are met. Then δf^2 is calculated as the sum of 1) the sample variance and 2) the square of the sample mean.

2.13 Dependence Problem

Formula (2.47) and its multi-dimension extensions such as Formula (2.52) accurately account for all contribution to the result uncertainty within $f(x)$, providing a clean and deterministic solution for $(\delta f)^2$, e.g., it gives the same result for Formula (1.7), Formula (1.8) and Formula (1.9). Therefore, precision arithmetic has no expression-based dependence problem.

It is tempting to define basic arithmetic operations as Formula (2.53), Formula (2.54) and Formula (2.56), and apply them progressively to calculate $f(x)$, similar to how basic arithmetic operations are used in conventional floating arithmetic. However, such an approach may apply the uncorrelated uncertainty assumption wrongly between a value and its mathematical expression, such as between x and x^2, so that it may result in the dependence problem similar to that of interval arithmetic [21]. For example, for such a use of precision arithmetic, only Formula (1.7) gives the correct result $4(x-\frac{1}{2})^2(\delta x)^2 + 3(\delta x)^4$, while Formula (1.8) over-estimates the result uncertainty by $4x(\delta x)^2$, which has the largest fold of over-estimation at $x = \frac{1}{2}$ when $\delta x < 1$. Let x, y and z be three values satisfying the uncorrelated uncertainty assumption. Functions $f(x, y)$ and $g(x, z)$ are correlated through x, and they need to be tested for the uncorrelated uncertainty assumption before they can be used to calculate $h(f, g)$ using precision arithmetic. For example, using precision arithmetic, the correlation γ between $(x \pm \delta x)$ and $(x \pm \delta x)^2$ is calculated by Formula (2.57), which shows that γ increases with decreased precision $P(x) \equiv \delta x/|x|$ of x, in contrast with how the uncorrelated uncertainty assumption favors finer precision P in Formula (2.4). After applying Formula (2.4), the correlation on the uncertainty level γ_P is no less than $\frac{16}{19}$, so that precision arithmetic rejects calculating Formula (1.8) progressively using the basic arithmetic operations.

$$\gamma = \frac{\int \left((x+\widetilde{y})^2 - x^2\right)\left((x+\widetilde{y}) - x\right)\rho(\widetilde{y})\, d\widetilde{y}}{\sqrt{\int \left((x+\widetilde{y})^2 - x^2\right)^2 \rho(\widetilde{y})\, d\widetilde{y}}\sqrt{\int ((x+\widetilde{y}) - x)^2 \rho(\widetilde{y})\ d\widetilde{y}}} = \frac{1}{\sqrt{1 + \frac{3}{4}P(x)^2}} \tag{2.57}$$

In other words, converting a numerical algorithm from using conventional floating-point arithmetic to using precision arithmetic may be more complicated than directly replacing the variable types and the arithmetic being used. To avoid the dependence problem, the safest approach is to obtain an analytic form of the final expression of an algorithm before applying Formula (2.47) and its multi-dimension extensions, similar to how symbolic calculations are currently used in affine arithmetic [40].

2.14 Conditional Execution

Conditional execution based on the comparison relation between two values is frequently used in practical algorithms [12]. When each value has associated uncertainty, the comparison relation between two values becomes quite different. This is particularly true for interval arithmetic, in which a value can be anywhere inside its bounding range [18]. In precision arithmetic, each value has a mathematically expected value plus a well-defined bounding distribution for uncertainty. The comparison relation between two imprecise values in precision representation can be defined either by their mathematically expected values, or by their statistical comparison relations based on confidence [4].

However, the usage of condition execution in a traditional algorithm needs to be re-evaluated conceptually with uncertainty statistics in mind when upgrading an algorithm to use precision arithmetic, because most conditional executions are created to optimize implementation. For example, LU decomposition [12] carefully chooses the sequence of execution to minimize rounding errors, so that it introduces additional dependence problem due to conditional execution, e.g., small value change of a matrix item can result in different conditional execution path and large result difference. In other words, to solve the linear equation $Ax = b$, in which A is a matrix, x and b are two vectors, with the uncertainty of A^{-1} analytically solvable using Taylor expansion (as demonstrated in Section 5), precision arithmetic prefers to solve it as $x = A^{-1}b$ than to use the LU decomposition method.

2.15 Calculation Inside Uncertainty

Formula (2.58) shows that the current choice of $\hat{R}$ calculates 0-bit inside uncertainty, e.g., 1 with precision 10^{-3} is represented as 1024 in S. In contrast, all other arithmetic represents the value as $2^{53}/2^{53}$. While calculating many bits inside uncertainty does not seem meaningful according to significance arithmetic [32], not calculating at all inside uncertainty may not be an optimal approach either. Thus, Formula (2.12) is modified as Formula (2.59), in which χ is a small constant positive integer, to introduce the χ-bit calculation inside uncertainty by providing an altered interpretation of the precision for $S{\sim}R\,2^E$.

$$\begin{aligned} 1/3 \pm 0.001 + 2/3 \pm 0.001 &= 341 + 6.29\ 2^{-10} + 683 - 6.29\ 2^{-10} \\ &= 1024?12.6\ 2^{-10} = 1 \pm 0.001\sqrt{2}; \end{aligned} \tag{2.58}$$

$$\delta x = \sqrt{R/6 \cdot 2^{-\chi}} 2^{E+\chi}; \tag{2.59}$$

Table 3 shows examples of precision arithmetic with different χ for $\hat{R} = 16$, e.g., with $\chi = 2$, precision arithmetic represents the expected value of 1.000 ± 0.001 as $2^{12}/2^{12}$. χ will be set to 4 empirically later in this paper. The value of $\delta x/2^{\chi}$ is defined as *resolution* for the corresponding precision arithmetic.

χ	0	1	2
$0.5 \pm 0.001 =$	$512?6.29\ 2^{-10}$	$1024?12.58\ 2^{-11}$	$2048?25.16\ 2^{-12}$
$1 \pm 0.001 =$	$1024?6.29\ 2^{-10}$	$2048?12.58\ 2^{-11}$	$4096?25.16\ 2^{-12}$
$1 \pm 0.002 =$	$512?6.29\ 2^{-9}$	$1024?12.58\ 2^{-10}$	$2048?25.16\ 2^{-10}$

Table 3: Examples of precision arithmetic with different χ for $\hat{R} = 16$, in which χ stands for bits calculated inside uncertainty.

The limited calculation inside uncertainty does not necessarily mean that precision arithmetic has a larger calculation error. In Formula eqn: addition with 0-bit inside uncertainty, the mathematically expected value for the result is precisely 1, even though the mathematically expected values of the two operands for addition are not precisely 1/3 and 2/3 after the uncertainty initiation, respectively.

2.16 Implementation

The conventional 64-bit floating-point standard IEEE-754 [9][10] has:

- 11 bits for storing exponent E;

- 53 bits for storing significand S (with a hidden MSB).
- 1 bit for storing sign;

To be a super set of the conventional 64-bit floating-point standard, an 80-bit implementation of precision arithmetic has:

- 11 bits for storing exponent E;
- 53 bits for storing significand S (without using the hidden MSB);
- 1 bit for storing sign;
- 2 bits for storing carry $\sim$;
- 13 bits to store the bounding range R as a fixed-point value.

Precision arithmetic is implemented in C++. With heavy additional codes to count for statistics and to detect implementation errors, it runs about seven times slower than the implementation of interval arithmetic using Formula (1.3), (1.4), (1.5) and (1.6). It is probably faster in speed than the implementation of interval arithmetic without the dependence problem [21]. With code weight trimming and optimization, its speed is expected to be improved at least 3-fold. Unlike conventional floating-point arithmetic, it only calculates a limited number of significand bits, e.g., 12 bits instead of all the 53 bits when the precision is 10^{-3} and χ is 2. Its slowest but very frequent operation is to find the position of the highest non-zero significand digit, which can be found instantly with a decoder [5]. Thus, future hardware optimization can also improve the speed of precision arithmetic by another estimated 10-fold.

2.17 Alternative Form of Precision Arithmetic

Because the uncorrelated uncertainty assumption can lead directly to 1) the Gaussian distribution as the underlying distribution for rounding errors, and 2) Formula (2.47) and its multi-dimension extensions such as Formula (2.52) for generic Taylor expansion, an *alternative form of precision arithmetic* is to represent each uncertainty-bearing value as $x \pm \delta x$ in Formula (2.16). The bounding range is then calculated from δx as the confidence interval [4] for any required upper limit on bounding leakage, e.g., if the required bounding leakage is 10^{-9} or less, the bounding interval is $[x - 6\delta x, x + 6\delta x]$. This alternative form of precision arithmetic is *not* adopted in this paper for the following reasons:

- For the actual numerical calculation, if conventional floating-point arithmetic is used separately for x and δx, then x and δx will be contaminated by unspecified amount of rounding errors. Because the calculation for δx is more complex than that for x, δx probably contains more rounding error than x. Thus, the current form of precision arithmetic defines its own floating-point representation for $x \pm \delta x$ as $S{\sim}R\ 2^E$.
- Another effect of using conventional floating-point arithmetic for x and δx is to calculate many bits inside uncertainty, whose validity is not clear at this moment. In contrast, as demonstrated by Table 3, the current form of precision arithmetic controls the number of bits calculated inside uncertainty.

However, the alternative form could be valuable in theoretical discussions of precision arithmetic.

2.18 Types of Uncertainties Included in Precision Arithmetic

There are four sources of result uncertainty after a calculation [1][12]:

- input uncertainties
- rounding errors
- truncation errors
- modelling errors

As described previously, both input uncertainties and rounding errors are included in the uncertainty specification of precision arithmetic.

In many cases, because a numerical algorithm approaches its analytic counterpart only after infinitive execution, a good numerical algorithm should have an estimator of the *truncation error* toward its analytic counterpart, such as the Cauchy reminder estimator for Taylor expansion [12], or the residual error for numerical integration [12]. Using conventional floating-point arithmetic, a subjective upper limit is chosen for the truncation error, to stop the numerical algorithm at limited execution [12]. However, such arbitrary upper limit may not be achievable with the amount of rounding errors accumulated during calculation, so that such upper limit may actually give a falsely small result precision. Because precision arithmetic tracks rounding errors of a calculation efficiently, it can be used to search for the optimal execution termination point for the numerical algorithm when the truncation error is no longer significant, which is named as the *truncation rule* in this paper. In other words, using precision arithmetic, the result precision of a calculation is determined by the inputs and the calculation itself. Section 7 and 8 will provide such cases of applying truncation rule to Taylor expansion and numerical integration, respectively.

Modelling errors arise when an approximate analytic solution is used, or when a real problem is simplified to obtain the solution. For example, Section 4 demonstrates that the discrete Fourier transformation is only an approximation for the mathematically defined Fourier transformation. Conceptually, modelling errors originate from mathematics, so they are outside the domain for precision arithmetic.

3 Standards and Methods for Comparing Uncertainty-Bearing Arithmetic

3.1 Comparing Standards and Methods

Algorithms each with a known analytic result are used to characterize uncertainty-bearing arithmetic. The difference between the arithmetic result and the analytic result is defined as the *value error*. The question is whether the uncertainty bounding range or the uncertainty deviation is enough to cover the value error with an increased amount of calculation for any input. Corresponding to two different goals for uncertainty-bearing, there are actually two different sets of measurements to characterize an uncertainty-bearing arithmetic:

- The ratio of the absolute value error to the uncertainty deviation is defined as the *tracking ratio* for each output value. An ideal uncertainty-tracking arithmetic should have an average tracking ratio close to 1.
- The ratio of the absolute value error to the uncertainty bounding range is defined as the *bounding ratio* for each output value. An ideal uncertainty-bounding arithmetic should have a maximal bounding ratio either 1 or less than but close to 1. If the maximal bounding ratio is larger than 1, *bounding leakage* measures the probability for errors to be outside uncertainty bounding range.

In both cases, all measurements should be stable for an algorithm so that they should not change significantly for different input deviation, input data, or the amount of calculation. For example, if different branches of conditional executions contain very different amounts of calculations, such stability is crucial for obtaining a valid estimation of result precision.

Without dependency problem, the tracking ratios for precision arithmetic is expected to be normal distributed, and the average tracking ratio is a constant $\frac{2}{\sqrt{2\pi}} \simeq 0.7979$ when the bounding leakage is ignored.

3.2 Comparing Uncertainty-Bearing Arithmetics

Precision arithmetic tracks both the uncertainty bounding range and the uncertainty deviation, so it can be evaluated for both goals. Independence arithmetic has no uncertainty bounding range, while interval arithmetic has no uncertainty deviation. To be able to compare all the three arithmetics, $[x - 6\delta x, x + 6\delta x]$ is used artificially as the bounding range for an average value x with deviation δx for independence arithmetic, and vice versa for interval arithmetic.

As stated previously:

- Independence arithmetic assumes that any two operands are independent of each other, which may not be true in most cases.
- Precision arithmetic assumes that the uncertainties of any two operands are independent of each other, but allows the two operands themselves to be correlated.
- Interval arithmetic has the worst-case assumption because it needs to have zero bounding leakage unconditionally.

The statistical assumption of precision arithmetic is weaker than that of independence arithmetic but stronger than that of interval arithmetic, so after executing the same algorithm on the same input data, the output deviation and the bounding range of precision arithmetic are expected to be larger than those of independence arithmetic but smaller than those of interval arithmetic.

- According to Formula (2.5) and Formula (2.8), the result deviation of addition and subtraction by precision arithmetic propagates in the same way as that of independence arithmetic, while the result bounding range propagates in the same way as that of interval arithmetic. Hence addition and subtraction cannot differentiate the three arithmetics.
- According to Formula (2.54) and Formula (2.56), the result precision of multiplication and division by precision arithmetic is always larger than that by independence arithmetic. However, if both operands have precisions much less than 1, the result precision of multiplication and division is very close to that of independence arithmetic. Thus, the result of precision arithmetic should be much closer to that of independence arithmetic.
- The uncertainty distribution of precision arithmetic is a truncated Gaussian distribution according to Formula (2.15). When an imprecise value is multiplied by a constant, because its uncertainty bounding range and its uncertainty distribution deviation cannot be scaled linearly simultaneously according to Formula (2.12) and Formula (2.13), precision arithmetic chooses to preserve the distribution deviation rather than the bounding range, thus introducing bounding leakages. Figure 5 suggests that the bounding range of precision arithmetic should be much narrower than that of interval arithmetic, while the shape of Gaussian distribution suggests that such introduced bounding leakage should be small when the truncation range is much larger than the distribution deviation, e.g., less than 10^{-6} for the chosen normalization method whose truncation range is about ± 5 deviations.
- Formula (2.47) and its multi-dimensional expansions such as Formula (2.52) are mathematically strict so that precision arithmetic has no dependence problem on expression differences. In contrast, there seems no similar solution for generic Taylor expansion using interval arithmetic, because there seems no general analytic solution to find maxima and minima for generic polynomial at any range [12]. In this respect, precision arithmetic is mathematically simpler than interval arithmetic.

3.3 Comparing Algorithms for Tests

Algorithms of completely different nature with each representative for its category are needed to test the generic applicability of uncertainty-bearing arithmetic.

An algorithm can be categorized by comparing the amount of its input and output data:

- *Transforming*: A transforming algorithm has about equal amounts of input and output data. The information contained in the data remains about the same after transforming. The Discrete Fourier Transformation is a typical transforming algorithm, which contains exactly the same amount of input and output data, and its output data can be transformed back to the input data using essentially the same algorithm. Matrix inversion is another such reversible algorithm. For reversible transformations, a unique requirement for uncertainty-bearing arithmetic is to introduce the least amount of additional uncertainty after forward and reverse transformation, which provides an objective testing standard for a uncertainty-bearing arithmetic. A test of uncertainty-bearing arithmetic using FFT algorithms is provided in Section 4, and a test of matrix inversion is provided in Section 5.

- *Generating*: A generating algorithm has much more output data than input data. Solving differential equations numerically and generating a numerical table of a specific function are two typical generating algorithms. The generating algorithm codes mathematical knowledge into data, so there is an increase of information in the output data. From the perspective of encoding information into data, Taylor expansion is also a generating algorithm. In generating algorithms, input uncertainty should also be considered when deciding if the result is good enough so that the calculation can stop. Some generating algorithms are theoretical calculations which involve no imprecise input so that all result uncertainty is due to rounding errors. Section 6 demonstrates such an algorithm, which calculates a table of the sine function using trigonometric relations and two precise input data, $sin(0) = 0$ and $sin(\pi/2) = 1$. In some other generating algorithms, the accumulation of rounding errors and input uncertainty should stop the algorithm at an optimal termination point using the truncation rule, which is demonstrated in Section 7.
- *Reducing*: A reducing algorithm has much less output data than input data such as numerical integration and statistical characterization of a data set. Some information of the data is lost while other information is extracted during reducing. Conventional wisdom is that a reducing algorithm generally benefits from a larger input data set [4]. Such a notion needs to be re-evaluated when uncertainty accumulates during calculation. A test of uncertainty-bearing arithmetic using numerical integration is provided in Section 8.

Other relations between the input and output can also be used to categorize an algorithm.

- In an *expressive* algorithm, each output is implemented as an analytic mathematical expression of inputs. Formula (2.47) and Formula (2.52) of precision arithmetic are powerful tools to solve expressive algorithms using precision arithmetic.
- In a *progressive* algorithm, each output is based on partial inputs and previously generated outputs. If an output depends on the state which is defined by previous inputs and outputs, the algorithm is also progressive. Most practical algorithms are progressive. Even if there may be an expected analytic mathematical expression between its input and output, an algorithm may not be expressive due to its progressive implementation. The dependence problem usually exists in a progressive algorithm. Section 9 discusses the behaviours of all the three arithmetic for a progressive algorithm.

3.4 Input Data to Use

To test input data of any precision, a precise input value can be cast to any specific input deviation using precision representation. There exist two ways of implementing such a casting:

- A *clean* signal is obtained by directly casting a perfect signal to a specific precision. Such casting may contain systematic rounding errors. For instance, if a perfect sine signal repeats 2^n times in 2^{n+2} samples, the signal contains only values 0, ± 1 and $\pm 1/\sqrt{2}$, with each value repeated multiple times in the signal. The symmetry of the arithmetic may be tested by the output symmetry of clean input signals, e.g., in the discrete Fourier transformation, the frequency space should be conjugately symmetrical [12] for a clean signal in signal space.

- A *noisy* signal is obtained by adding Gaussian noise of the same deviation as the input deviation to a perfect signal before casting. It represents a realistic signal, and it should be used in validating arithmetic on uncertainty propagation.

4 Comparison Using FFT

4.1 Frequency Response of DFT (Discrete Fourier Transformation)

Each testing algorithm needs to come under careful scrutiny. One important issue here is whether the digital implementation of the algorithm is faithful for the original analytic algorithm. For example, the DFT is only faithful for continuous Fourier transformation at certain frequencies, and it has a different degree of faithfulness for other frequencies. This is called the frequency response of the DFT in this paper.

For each signal sequence $h[k], k = 0, 1 \dots N-1$, in which N is a positive integer, the DFT $H[n], n = 0, 1 \dots N-1$ and its reverse transformation is given by Formula (4.1) [12], in which k is the *index frequency* for the DFT:

$$H[n] = \sum_{k=0}^{N-1} h[k]\, e^{i2\pi \frac{k}{N} n}; \qquad h[k] = \frac{1}{N} \sum_{n=0}^{N-1} H[n]\, e^{-i2\pi \frac{n}{N} k}; \tag{4.1}$$

The $H[n]$ of a pure sine signal $h[k] = \sin(2\pi f k/N)$ is calculated by Formula (4.2), in which f is the frequency of the sine wave. When f is an index frequency for $H[n]$, Formula (4.2) becomes Formula (4.3). Otherwise, the general solution for Formula (4.2) is Formula (4.4), which approaches (4.3) when f approaches its closest integer F, or Formula (4.5) when f approaches $F \pm 1/2$.

$$H[n] = \frac{\sum_{k=0}^{N-1} e^{i2\pi(n+f)\frac{k}{N}} - \sum_{k=0}^{N-1} e^{i2\pi(n-f)\frac{k}{N}}}{2i}; \tag{4.2}$$

$$H[n] = i\delta_{n,F} N/2; \tag{4.3}$$

$$H[n] = \frac{1}{2} \frac{\sin(2\pi f - 2\pi\frac{f}{N}) + \sin(2\pi\frac{f}{N}) - \sin(2\pi f) e^{-i2\pi\frac{n}{N}}}{\cos(2\pi\frac{n}{N}) - \cos(2\pi\frac{f}{N})}; \tag{4.4}$$

$$H[n] = N/\pi; \tag{4.5}$$

The DFT $H[n]$ of the signal $h[k]$ is the digital implementation of the continuous Fourier transformation $H(s)$ of the signal $h(t)$ [12], in which $H(s) = i\delta(s - f)$ for $h[k] = \sin(2\pi f)$. From Formula (4.4), when the signal frequency of the original signal falls between two index frequencies of the transformation, the peak is lower and wider with a wrong phase, depending on the fractional frequency $|f - F|$. Thus, the DFT is only faithful for signal components with exactly one of the index frequencies of the transform, and it suppresses and widens unfaithful signal components, each of which has a phase different from its closest faithful representation, with the phase of a sine wave distorted toward that of a cosine wave, and vise visa. Examples of unfaithful representations of fractional frequency by the DFT are shown in Figure 8.

Due to its width, a frequency component in an unfaithful transformation may interact with other frequency components of the Discrete Fourier spectrum, thus sabotaging the whole idea of using the Fourier Transformation to decompose a signal into independent frequency components. Because the reverse DFT mathematically restores the original $\{h[k]\}$ for any $\{H[n]\}$, it exaggerates and narrows all unfaithful signal components correspondingly. This means that the common method of signal processing in the Fourier space [12][15][17] may generate artefacts due to its uniform treatment of faithful and unfaithful signal components, which probably coexist in reality. Unlike aliasing [5][12][17], unfaithful representation of the DFT has an equal presence in the whole frequency range so that it cannot be avoided by sampling the original signal differently.

An unfaithful representation arises from the implied assumption of the DFT. The continuous Fourier transformation has an infinitive signal range so that:

$$h(t) \Leftrightarrow H(s): \quad h(t-\tau) \Leftrightarrow H(s)e^{i2\pi s\tau}; \tag{4.6}$$

As an analog, the DFT $G[n]$ of the signal $h[k], k = 1 \dots N$ can be calculated mathematically from the DFT $H[n]$ of $h[k], k = 0 \dots N-1$:

$$G[n] = (H[n] + h[N] - h[0])e^{i2\pi n/N}; \tag{4.7}$$

Applying Formula (4.6) to Formula (4.7) results in Formula (4.8).

$$h[N] = h[0]; \tag{4.8}$$

Thus, the DFT has an implied assumption that the signal $h[k]$ repeats itself outside the region of $[0, N-1]$ [41]. For an unfaithful frequency, $h[N-1]$ and $h[N]$ are discontinuous in regard to signal periodicity, resulting in larger peak width, lower peak height, and the wrong phase.

The most convenient signals to test uncertainty-bearing arithmetic are perfect sine or cosine signals with index frequencies. A linear signal with the slope $\lambda, h[k] = \lambda k$, provides a generic test for input frequencies other than index frequencies, whose Fourier spectrum is:

$$H[n] = -\lambda\frac{N}{2}\left(1 + \frac{i}{\tan(\pi n/N)}\right); \tag{4.9}$$

4.2 FFT (Fast Fourier Transformation)

When $N = 2^L$, in which L is a positive integer, the generalized Danielson-Lanczos lemma [12] can be applied to the DFT as FFT [12], in which $m = L, L-1, \dots 1, 0$ indicates progress of the transformation, and j is the bit-reverse of n:

$$m = L: \quad H[n, \frac{k}{2^m}] = h[j], k, n = 0, 1 \dots N-1; \tag{4.10}$$

$$m = L-1 \dots 0: \quad H[n, \frac{k}{2^m}] = H[n, \frac{k}{2^{m+1}}] + H[n, \frac{k}{2^{m+1}}] \; \exp(+i2\pi\frac{n}{2^{L-m}}); \tag{4.11}$$

$$m = 0: \quad H[n] = H[n, \frac{k}{2^m}]; \tag{4.12}$$

Thus, each output value is obtained after applying Formula (4.11) L times. L is called FFT order in this paper.

The calculation of the term $\exp(i2\pi\frac{n}{2^{L-m}})$ in Formula (4.11) can be simplified. Let $<<$ denote a bit left-shift operation and let & denote a bitwise AND operation:

$$\varphi[n] \equiv \exp(i2\pi\frac{n}{2^L}): \quad \exp(i2\pi\frac{n}{2^{L-m}}) = \varphi[(n << m)\&((1 << L) - 1)]; \tag{4.13}$$

It is important to have an accurate phase factor array $\varphi[n]$ when tracking the FFT calculation error. The accuracy of $\varphi[n]$ can be checked rigidly within itself by trigonometric relations so that no significant error is introduced from trigonometric functions.

Formula (4.11) always sums up two mutually independent operands, so the error propagation in a FFT algorithm is precisely tracked by independence arithmetic, and the dependency problem should not be a concern for interval arithmetic and precision arithmetic.

FFT is one of the most widely used algorithms [12]. By providing a balanced usage of addition, subtraction and multiplication involving trigonometric functions, it services as one of the most important benchmarks in testing processors for overall mathematical performance [42]. Since all three uncertainty-bearing arithmetics are generic in nature without special optimization for FFT algorithms, the testing result using FFT algorithms should be generic for expressive algorithms. FFT algorithms provide a good linear platform to test any uncertainty-bearing arithmetic, with 1) a clearly defined value measuring the amount of calculation, 2) a known error propagation mechanism, 3) no conditional execution in the algorithm, and 4) using only basic arithmetic operations without the dependence problem.

4.3 Evaluating Calculation Inside Uncertainty

Figure 9 shows the output deviations and value errors for a noisy sine signal after forward FFT. It shows that the output deviations using precision arithmetic are slightly larger than the output deviations using independence arithmetic, but much less than those using interval arithmetic. For a fixed input deviation, the output deviation using independence arithmetic is a constant for each FFT. Because the value and uncertainty interact with each other through normalization in precision arithmetic, output deviations of Formula (4.11) are no longer a constant. One interesting consequence is that only in precision arithmetic the output deviations for a noisy input signal are larger than those for a corresponding clean input signal.

Figure 9 shows that the value errors calculated using precision arithmetic are comparable to those using conventional floating-point arithmetic, and they are both comparable to the output deviations using either precision arithmetic or independence arithmetic. In other words, the result of calculating 2-bit or 53-bit into uncertainty are quite comparable so that the limited calculation inside uncertainty is reasonable.

Figure 10 compares the output value errors of precision arithmetic calculating different bits inside uncertainty. With no calculation inside uncertainty, the output value errors exist only on four levels. Such quantum distribution is reduced noticeably by the 2-bit calculation inside uncertainty, and is further reduced by the 4-bit calculation inside uncertainty. Compared with Figure 9, Figure 10 shows that the result using precision arithmetic with the 4-bit calculation inside uncertainty approaches that using independence arithmetic so that the 4-bit calculation inside uncertainty seems sufficient. Precision arithmetic with the 4-bit calculation inside uncertainty is used for further tests.

4.4 Evaluating Uncertainty Distribution

Each output value error is normalized with the corresponding output uncertainty deviation before it is counted for histogram. If output value errors are Gaussian-distributed with the deviation given precisely by the corresponding output uncertainty deviation, then the normalized histogram should be normal-distributed. Figure 11 and Figure 12 show that such histograms using wither independence arithmetic or precision arithmetic with 4-bit calculated inside uncertainty are both best fit by Gaussian distribution with the deviation of 0.98 and the mean of 0.06. Due to limited bits calculated inside uncertainty and normalization, the population at where the value errors are zero is expected to be larger for the precision arithmetic, e.g., during normalization, all values which is less than 4-fold of resolution becomes 0. This phenomenon is confirmed by Figure 12.

Precision arithmetic tracks all increases of rounding errors, but it cannot track decreases of the rounding error due to mutual cancellations during arithmetic operations. Hence the uncertainty distribution provided by precision arithmetic serves as the bounding distribution for value errors, and the actual distribution could be narrower than the bounding distribution. FFT provides a good test for such probability bounding. Its forward and reverse algorithms are identical except for a constant so that they result in exactly the same bounding probability distributions. On the other hand, the forward FFT condenses a sine signal into only two non-zero imaginary values by mutual cancellation of signal components, while the reverse FFT spreads only two non-zero imaginary values to construct a sine signal. Thus, the forward FFT is more sensitive to calculation errors than the reverse FFT, and should have a broader actual uncertainty distribution. Indeed in Figure 12, the histogram for the reverse algorithm for a sine signal with added noise is narrow than that of the forward algorithm, with that for the round-trip algorithm in the middle of the two.

4.5 Evaluating Uncertainty-Tracking

Figure 13 shows that for the same input deviation, the output deviations of the forward FFT increase exponentially with the FFT order using all three arithmetics. Figure 14 shows that for the same FFT order, the output deviations of the forward FFT increase linearly with the input deviation using all three arithmetics. The output deviation does not change with input frequency so that all data of the same input deviation and the same FFT order but with different input frequencies can be pooled together during analysis. The trends in Figure 13 and Figure 14 are modeled by Formula (4.14), in which L is the FFT order, δx is the input deviation, δy is the average output deviation, and α and β are empirical fitting constants:

$$\delta y = \alpha \beta^L \delta x; \tag{4.14}$$

β measures the propagation speed of the deviation with an increased amount of calculation in Formula (4.14). It is called *propagation base rate.* Unless β is close to 1, β dominates α in fitting, thus determining characteristics of Formula (4.14).

It turns out that Formula (4.14) is a very good fit for both average output deviations and value errors for all three arithmetics, such as demonstrated in Figure 15. Because uncertainty-tracking is a competition between error propagation and uncertainty propagation, the average output tracking ratio for the forward FFT is expected to fit Formula (4.15) and Formula (4.16), in which z is the average output tracking ratio, L is the FFT order, $(\alpha_{dev}, \beta_{dev})$ and $(\alpha_{err}, \beta_{err})$ are fitting parameters of Formula (4.14) for average output deviations and value errors, respectively:

$$z = \alpha \beta^L; \tag{4.15}$$

$$\alpha = \alpha_{err}/\alpha_{dev}; \quad \beta = \beta_{err}/\beta_{dev}; \tag{4.16}$$

The estimated average output tracking ratio can then be compared with the measured ones to evaluate the predictability of the uncertainty-tracking mechanism. One example of measured average output tracking ratios is shown in Figure 16, which shows that the average output tracking ratios using precision arithmetic are a constant despite that both average output uncertainty deviations and value errors increase linearly with the input deviation and exponentially with the FFT order. Formula (4.14) and

Formula (4.15) are found empirically to be a good fit for any FFT algorithm with any input signal using any arithmetic.

The Reverse FFT algorithm is identical to the Forward FFT algorithm, except when:

- The Reverse FFT algorithm uses constant (-i) instead of (+i) in Formula (4.11).
- The Reverse FFT algorithm divides the result further by 2^L.

Thus, the average output deviations and value errors of the reverse FFT algorithm are expected to obey Formula (4.14) and Formula (4.17) , in which $(\alpha_{for}, \beta_{for})$ are corresponding fitting parameters of Formula (4.14) for the forward FFT, while the average output tracking ratios are expected to obey Formula (4.16) with the same α and β as those of the forward FFT.

$$\alpha = \alpha_{for}; \quad \beta = \beta_{for}/2; \tag{4.17}$$

The Round-trip FFT is the forward FFT followed by the reverse FFT, with the output of the forward FFT as input to the reverse FFT. Thus, both its average output deviations and value errors are expected to fit Formula (4.14) and Formula (4.18), in which $(\alpha_{for}, \beta_{for})$ and $(\alpha_{rev}, \beta_{rev})$ are corresponding fitting parameters of Formula (4.14) for the forward FFT and the reverse FFT, respectively. Its tracking ratios are expected to fit Formula (4.15) and Formula (4.18), in which $(\alpha_{for}, \beta_{for})$ and $(\alpha_{rev}, \beta_{rev})$ are corresponding fitting parameters of Formula (4.15) for the forward FFT and the reverse FFT, respectively.

$$\alpha = \alpha_{for}\alpha_{rev}; \quad \beta = \beta_{for}\beta_{rev}; \tag{4.18}$$

Figure 17, Figure 18 and Figure 19 show the fitting of β for independent, precision and interval arithmetic for all the three algorithms, respectively. These three figures show that all measured β make no distinction between input signals for any algorithms using any arithmetic, e.g., there is no difference between the real part and the imaginary part for a sine signal. The estimated β for average tracking ratios is obtained from Formula (4.16). The estimated β for average uncertainty deviations and value errors for the reverse FFT and the roundtrip FFT are obtained from Formula (4.17) and Formula (4.18), respectively. The estimated β for average uncertainty deviations for the forward FFT is $\sqrt{2}$, which will be demonstrated later. The measured β and the estimated β agree well with each other in all cases. This confirms that uncertainty-tracking is a simple competition between the error propagation and uncertainty propagation:

- Figure 17 confirms that independence arithmetic is ideal for uncertainty-tracking for FFT algorithms: 1) β for tracking ratios is a constant 1; and 2) β for both the average output deviations and value errors is both 1 for the round-trip FFT because the result signal after the round-trip FFT should be restored as the original signal. Thus, theoretical β for the forward FFT and the reverse FFT are $\sqrt{2}$ and $1/\sqrt{2}$, respectively.
- Precision arithmetic has β for average output deviations slightly larger than those of value errors, resulting in β for average output tracking ratios to be a constant slightly less than 1. Its β for average output deviations is slightly larger than the corresponding β of independence arithmetic, so its average output deviations propagate slightly faster with an increased FFT order than those of

independent arithmetic. Such slightly faster increase with the amount calculation is anticipated by the difference between Formula (2.54) and Formula (1.12) with $\gamma = 0$.

- The β for average output deviations using interval arithmetic is always much larger than β for average output value errors, resulting in β for average output tracking ratios of about 0.62 for the forward and reverse FFT, and about $0.39 \cong 0.62^2$ for the roundtrip FFT. Consequently, using interval arithmetic, the average output deviations propagate much faster with the amount of calculations than the value error does. Such fast propagation of uncertainty ranges is intrinsic to interval arithmetic due to its worst-case assumption.

Figure 20 shows that for the forward FFT, the measured average output tracking ratios using either precision arithmetic or independence arithmetic are approximately constant of 0.8 in both cases, regardless of the FFT order. In contrast, Figure 20 shows that using interval arithmetic the measured average output tracking ratios decrease exponentially with the FFT order L. Such trends of average tracking ratios hold for all three FFT algorithms and all input signals. Thus, in this case, the direct uncertainty tracking provided by precision arithmetic is better than the indirect uncertainty tracking provided by interval arithmetic.

Figure 21 shows that using precision arithmetic, each average output uncertainty deviation equals the corresponding input uncertainty deviation for all FFT orders after a round-trip operation. Thus, after each round-trip operation, precision arithmetic restores the original signal and the corresponding uncertainty for FFT. Such behavior seems ideal for a reversible algorithm. In contrast, Figure 22 shows that using interval arithmetic, the average output uncertainty deviations increase exponentially with FFT orders, which means the undesirable broadening of uncertainty in the restored signal after a round-trip operation.

4.6 Evaluating Uncertainty-Bounding

While uncertainty tracking is the result of the propagation competition between average output deviations and average values errors with increased amount of calculations, uncertainty bounding is the result of the propagation competition between output bounding ranges and maximal value errors, both of which still fit Formula (4.14) well using any arithmetic experimentally. Formula (4.15) and Formula (4.16) can be used to estimate the maximal bounding ratio as well. For example, Figure 23 shows that the maximal output bounding ratios using precision arithmetic fit Formula (4.15) well. Unlike average output tracking ratios in Figure 20, the maximal output bounding ratios increase slowly with the FFT order using either precision arithmetic or independent arithmetic. In contrast, interval arithmetic has its maximal bounding ratios decreasing exponentially with the increased FFT order for all algorithms while keeping its bounding leakages at constant 0. Detailed analysis shows that in interval arithmetic, β for the maximal uncertainty bounding ranges exceeds β for the maximal value error, suggesting the source of over-estimating uncertainty range with the increased amount of calculations. Defining empirical *deviation leakage* as the frequency of the value errors to be outside the range of mean $\pm$ deviation, Figure 24 shows that the deviation leakages is roughly a constant using precision arithmetic, suggesting the statistical nature of uncertainty bounding using precision arithmetic. Whether precision arithmetic is better than interval arithmetic in uncertainty bounding depends on the statistical requirements for the uncertainty bounding:

- In the situation when absolute bounding is required, interval arithmetic is the only choice.
- In the range estimation [1] involving low-resolution measurements whose sources of uncertainty are unclear, interval arithmetic is a better choice because the independence uncertainty assumption of precision arithmetic may not be satisfied.
- Otherwise, precision arithmetic should be more suitable for normal usages.

5 Comparison Using Matrix Inversion

5.1 Uncertainty Propagation in Matrix Determinant

Let vector $[p_1, p_2 \dots p_n]_n$ denote a permutation of the vector $(1, 2 \dots n)$ [43]. Let $\$[p_1, p_2 \dots p_n]_n$ denote the permutation sign of $[p_1, p_2 \dots p_n]_n$ [43]. For a n-by-n square matrix M with the element $x_{i,j}, i, j = 1, 2 \dots n$, let its determinant be defined as Formula (5.1) [12] and let the sub-determinant at index (i, j) be defined as Formula (5.2) [43]:

$$|M| \equiv \sum_{[p_1 \cdots p_n]_n} \$[p_1 \dots p_n]_n \prod_{k} x_{k,p_k}; \tag{5.1}$$

$$|M|_{i,j} \equiv \sum_{[p_1 \cdots p_n]_n}^{p_i = j} \$[p_1 \dots p_n]_n \prod_{k}^{k \neq i} x_{k,p_k}; \tag{5.2}$$

$(-1)^{i+j}|M_{(i,j)}|$ is the determinant of the $(n-1)$-by-$(n-1)$ matrix that results from deleting the row i and column j of M [12]. Formula (5.3) holds for the arbitrary row index i or the arbitrary column index j [12]:

$$|M| = \sum_{j=1}^{n} |M_{i,j}| x_{i,j} = \sum_{i=1}^{n} |M_{i,j}| x_{i,j}; \tag{5.3}$$

Assuming $p_1, p_2 \in \{1, 2...n\}$, let $[p_1, p_2]_n$ denote the length-2 unordered permutation which satisfies $p_1 \neq p_2$, and let $< p_1, p_2 >_n$ denote the length-2 ordered permutation which satisfies $p_1 < p_2$. Letting $< i_1, i_2 >_n$ be an arbitrary ordered permutation, Formula (5.3) can be applied to $M_{i,j}$, as:

$$|M_{<i_1,i_2>_n [j_1,j_2]_n}| \equiv \sum_{[p_1 \cdots p_n]_n}^{p_{i_1} = j_1, p_{i_2} = j_2} \$[p_1 \dots p_n]_n \prod_{k}^{k \neq i_1, k \neq i_2} x_{k,p_k}; \tag{5.4}$$

$$|M| = \sum_{j_1} x_{i_1,j_1} |M_{i_1,j_1}| = \sum_{j_1} \sum_{j_2}^{i_2 \neq i_1, j_2 \neq j_1} x_{i_1,j_1} x_{i_2,j_2} |M_{<i_1,i_2>_n [j_1,j_2]_n}|; \tag{5.5}$$

Because $|M_{<i_1,i_2>_n [j_1,j_2]_n}|$ relates to the determinant of the $(n-2)$-by-$(n-2)$ matrix that results from deleting the row i_1 and i_2, and the column j_1 and j_2 of M. This leads to Formula (5.6).

$$||M_{<i_1,i_2>_n [j_1,j_2]_n}|| = ||M|_{<i_1,i_2>_n [j_2,j_1]_n}||; \tag{5.6}$$

The definition of a sub-determinant can be extended to Formula (5.7), in which $m \in \{1, 2...n\}$. Formula (5.5) can be generalized as Formula (5.8), in which $m \in \{1, 2...n\}$ and $< i_1 \dots i_m >_n$ is an arbitrary ordered permutation. Formula (5.8) can be viewed as the extension for both Formula (5.3) and Formula (5.1).

$$|M_{<i_1 \dots i_m>_n [j_1 \cdots j_m]_n}| \equiv \sum_{[p_1 \cdots p_n]_n}^{p_{i_k} = j_k, k \in \{1 \dots m\}} \$[p_1 \dots p_n]_n \prod_{k \in \{1 \dots n\}}^{k \notin \{i_1 \dots i_m\}} x_{k,p_k}; \tag{5.7}$$

$$|M| = \sum_{[j_1 \cdots j_m]_n} |M_{<i_1 \dots i_m>_n [j_1 \cdots j_m]_n}| \prod_{k=1}^{m} x_{i_k,j_k}; \tag{5.8}$$

According to the basic assumption of precision arithmetic, the uncertainty of each element $x_{i,j}$ is independently and symmetrically distributed. Let $\widetilde{y}_{i,j}$ denote a random variable at the index (i,j) symmetrically distributed with the deviation $\delta x_{i,j}$. Let $\widetilde{|M|}$ denote the determinant of the matrix $\widetilde{M}$ whose element is $(x_{i,j}+\widetilde{y}_{i,j})$. Applying Taylor expansion to Formula (5.8) results in Formula (5.9), which results in Formula (5.10) after applying Formula (2.44):

$$|\widetilde{M}| - |M| = \sum_{m=1}^{n} \sum_{<i_1 \dots i_m>_n} \sum_{[j_1 \dots j_m]_n} |M_{<i_1 \dots i_m>_n [j_1 \dots j_m]_n}| \prod_{k=1}^{m} \widetilde{y}_{i_k,j_k}; \tag{5.9}$$

$$\delta|M|^2 = \sum_{m=1}^{n} \sum_{<i_1 \dots i_m>_n} \sum_{[j_1 \dots j_m]_n} |M_{<i_1 \dots i_m>_n [j_1 \dots j_m]_n}|^2 \prod_{k=1}^{m} \delta x_{i_k,j_k}^2; \tag{5.10}$$

Defining $|M_{<>_n<>_n}| \equiv |M|$, Formula (5.11) is an recursive form of Formula (5.10):

$$\delta|M_{<p_1 \dots p_k>_n <q_1 \dots q_k>_n}|^2 = \sum_{p_i} \sum_{q_j} \delta x_{p_i,q_j}^2$$
$$(|M_{<p_1 \dots p_i \dots p_k>_n <q_1 \dots q_j \dots q_k>_n}|^2 + \delta|M_{<p_1 \dots p_i \dots p_k>_n <q_1 \dots q_j \dots q_k>_n}|^2); \tag{5.11}$$

When using Formula (5.3) to calculate determinant in conventional floating-point arithmetic:

- The input uncertainty can not be accounted for.
- One path is chosen out of many possible paths, such as selecting a different sub-determinant to start with.
- Because of the rounding error, each path may result in a different result even if all elements of the determinant are precise, and the spread of all results is expected to be inversely proportional to the stability of the matrix [44].

In another word, using conventional floating-point arithmetic, the calculation of determinant is one leap of faith. Instead, Formula (5.11) shows that the result uncertainty is the aggregation of uncertainties from all possible path of Formula (5.3). To accounts for all such uncertainties, Formula (5.11) starts from all 1x1 sub-determinants, and constructs all sub-determinants whose size is 1 larger, until reaches the determinant itself. Thus, uncertainty-bearing calculation should be order-of-magnitude more complex and time-consuming than the correspond calculation using conventional floating-point arithmetic.

The element $z_{i,j}$ at the index (i,j) of the inverted matrix M^{-1} is calculated as [43]:

$$z_{i,j} = \frac{|M_{j,i}|}{|M|}; \tag{5.12}$$

Formula (5.12) shows that the uncertainty of the matrix determinant $|M|$ propagates to every element of the inverted matrix M^{-1}. Instead, the matrix which consists of the element $|M_{j,i}|$ at the index (i,j) is defined as the adjugate matrix M^A [43], whose elements are not directly affected by M^{-1}. M^A is recommended to replace M^{-1} whenever the application allows [12].

5.2 Matrix Testing Algorithm

A matrix $\widehat{M}$ is constructed using random integers between [-16384, + 16384]. Its adjugate matrix $\widehat{M}^A$ and its determinant $|\widehat{M}|$ are calculated precisely using integer arithmetic. $\widehat{M}$, $|\widehat{M}|$ and $\widehat{M}^A$ are all scaled proportionally as M, $|M|$ and M^A so that the elements of M are 2's fractional numbers randomly distributed between [-1, +1]. The scaled matrix M is called a clean testing matrix. M^{-1} is calculated from $|M|$ and M^A using Formula (5.12). Floating-point arithmetic is used to calculate M^A and M^{-1} from M, and the results are compared with the corresponding precise results for value errors. Gaussian noises corresponding to different deviations between 10^{-17} and 10^{-1} may be added to each clean testing matrix, to result in noisy testing matrix. Each combination of matrix size and input deviation is tested by 32 different noisy matrices.

5.3 Testing Matrix Stability

Each matrix has a different stability [44], which means how stable the inverted matrix is in regard to small value changes of the original matrix elements. It is well known that more mutual cancellations in Formula (5.1) mean less stability of the matrix [11][12], with the Hilbert matrix [45] being the most famous unstable matrix. The condition number has been defined to quantify the stability of a matrix [44]. Even though the definition of the condition number excludes the effects of rounding errors, in reality most calculations are done numerically using conventional floating-point arithmetic so that the combination effect of rounding errors and matrix instability cannot be avoided in practice. When a matrix is unstable, the result is more error prone due to rounding errors of conventional floating-point arithmetic [11]. Consequently, there are no general means to avoid the mysterious and nasty "numerical instability" in numerical applications due to rounding errors [11]. For example, the numerical value of the calculated condition number of a matrix may have already been a victim of "numerical instability", and there is no sure way to judge this suspicion, so this value may not be very useful in judging the stability of the matrix in practice. On the other hand, the rounding errors of conventional floating-point arithmetic can be used to test the stability of a matrix. Rounding errors effectively change the item values of a matrix, so they produce a larger effect on a less stable matrix. If the inverted matrix and the adjugate matrix are calculated using conventional floating-point arithmetic, larger value errors indicate that the matrix is less stable.

Precision arithmetic accounts for all rounding error with stable characterization of result uncertainties. More mutual cancellations in Formula (5.1) will result in a smaller absolute value related to the uncertainty deviation of the determinant. Thus, the precision of the determinant $|M|$ of a matrix M calculated using precision arithmetic measures the amount of mutual cancellations, and it may measure the stability of a matrix. Particularly, if $|M|$ is of coarser precision, then each element of M^{-1} should tend to have a larger value error, according to Formula (5.12). This hypothesis is confirmed by Figure 25, which shows a good linear relation between the precision of $|M|$ and the average value error of its inverted matrix M^{-1}, regardless of the matrix size. The maximal output values errors are related to the precision of $|M|$ in the same fashion. In contrast, Figure 26 shows that the value errors of the adjugate matrix M^A do not depend noticeably on the precision of $|M|$. Thus, the precision of the denominator in Formula (5.12) determines the overall stability in matrix inversion, confirming the validity of common advice to avoid matrix inversion operations in general [12].

Such a linear relation between the precision and the value error also extends to the calculation of the adjugate matrix. Let the relative value error be defined as the ratio of the value error divided by the expected value. The relative error is expected to correspond to the result precision linearly. Figure 27 compares each precision of the sub-matrix determinant $|M_{j,i}|$ with the corresponding relative error of the element at the index (i,j) of the adjugate matrix M^A of the clean matrix of different sizes. It shows that larger relative errors of adjugate matrix elements indeed correspond to coarser precisions of the sub-matrix determinant.

While each condition number [44] only gives the result sensitivity to one matrix element, Formula (5.10) contains the result sensitivity to any matrix element, any combination of matrix elements, as well as the aggregated result uncertainty deviation. Therefore, Formula (5.10) and Formula (5.11) may be better than the condition numbers for describing matrix stability.

5.4 Testing Uncertainty Propagation in Adjugate Matrix

When the adjugate matrix is calculated using precision arithmetic, Figure 28 shows that the average output deviations for the adjugate matrix increase linearly with the input deviation, which is in good agreement with Formula (4.14). Such relation is also true for maximal and average output values errors. Formula (4.14) is expected to describe the general value error propagation for linear algorithms in which L is the amount of calculations [14]. The question is what value L should be when calculating the adjugate matrix of a square matrix of size N. Figure 28 suggests that L increases with N^2 for the average output precision and average output error[10].

Figure 29 shows that the average output tracking ratio of the adjugate matrix using precision arithmetic is approximately a constant of 0.8. Figure 29 is very similar to Figure 16. Similar to the maximal output bounding ratios of FFT algorithms, the maximal output bounding ratios for the adjugate matrix using precision also obey Formula (4.15) well, with β of 1.005, meaning a slow increase with the matrix size. Added to the similarity is the normalized uncertainty distribution shown in Figure 30, which is very similar to Figure 12. Even though FFT and the calculating adjugate matrix are two very different sets of linear transformational algorithms, their uncertainty propagation characteristics are remarkably similar even in quantitative details. This similarity indicates that precision arithmetic is a generic arithmetic for linear algorithms.

5.5 Calibration

Because $|M_{j,i}|$ and $|M|$ are not independent of each other, M^{-1} calculated by Formula (5.12) contains the dependency problem. Figure 30 shows that the tracking ratios for the adjugate matrix and the inverted matrix are both standard distributed, while they are exponentially distributed when the inverted matrix is inverted again. Because the inverted matrix has the same tracking ratio distribution as that of the adjugated matrix, which has no dependency problem, the inverted matrix contains hardly any dependency problem. In contrast, Figure 30 shows that the double inverted matrix is severely affected by the dependency problem, such that its tracking ratio increases with matrix size as shown in Figure 31. Figure 32 shows that average tracking ratios

[10]The amount of calculation L does not mean the calculation complexity using the Big O notation [46]. It is just a measurement of how output uncertainty increases with a dimension of calculation according to (4.14) [14]. For example, any sorting algorithm will not change the uncertainty distribution, so that L is always 0 regardless the calculation complexity for the sorting algorithm. The measured calculation time suggests calculation complexity of $O(2^N)$ for using Formula (5.11) to calculate the matrix determinant.

for different matrix sizes follows a same exponential distribution, but with different extend, e.g., the distribution for matrix size 4 has yet reaches stable distribution beyond 2.5, which causes the increase of the average tracking ratio with the matrix size as shown in Figure 31.

Applying the same algorithms twice results in so much differences, which shows that the dependency problem has been embedded in the data, and which shows the importance of calibration.

6 Comparison Using Recursive Calculation of Sine Values

Starting from Formula (6.1), Formula (6.2) and Formula (6.3) can be used recursively to calculate the phase array $\varphi[n]$ in Formula (4.13).

$$\sin(0) = \cos(\frac{\pi}{2}) = 0; \qquad \sin(\frac{\pi}{2}) = \cos(0) = 1; \tag{6.1}$$

$$\sin\left(\frac{\alpha+\beta}{2}\right) = \sqrt{\frac{1-\cos(\alpha+\beta)}{2}} = \sqrt{\frac{1-\cos(\alpha)\cos(\beta)+\sin(\alpha)\sin(\beta)}{2}}; \tag{6.2}$$

$$\cos\left(\frac{\alpha+\beta}{2}\right) = \sqrt{\frac{1+\cos(\alpha+\beta)}{2}} = \sqrt{\frac{1+\cos(\alpha)\cos(\beta)-\sin(\alpha)\sin(\beta)}{2}}; \tag{6.3}$$

This algorithm is very different from both FFT and matrix inversion in nature because Formula (6.2) and Formula (6.3) are no longer linear, and the test presents a pure theoretical calculation without input uncertainty. The recursion iteration count L is a good measurement for the amount of calculations. Each repeated use of Formula (6.2) and Formula (6.3) accumulates calculation errors to the next usage so that both value errors and uncertainty are expected to increase with L. Each recursion iteration L corresponds to 2^{L-2} outputs, which enables statistical analysis for large L.

Figure 33 shows that both average output value errors and the corresponding average output deviation increase exponentially with the recursion count for all three arithmetics, and Figure 34 shows that in response to the increased amount of calculations:

- The average tracking ratio for precision arithmetic is a constant about 0.25;
- The maximal output bounding ratio for precision arithmetic increases slowly;
- The average tracking ratio for interval arithmetic decreases exponentially; and
- The maximal output bounding ratio for interval arithmetic remains roughly a constant.

Unlike FFT algorithms, the initial precise sine values participate in every stage of the recursion, which results in few small output deviations at each recursion. Detailed inspection shows that the maximal output bounding ratios for interval arithmetic are all obtained from small output deviations, and bounding ratios using interval arithmetic in general decrease exponentially with the amount of calculations. Thus, the result uncertainty propagation characteristics of the regressive calculation of sine values are very similar to those of both FFT and the calculating adjugate matrix; even though all these algorithms are quite different in nature. This may indicate again that the stability of precision arithmetic is generic, regardless of the algorithms used.

7 Validation Using Taylor Expansion

When a Taylor expansion is implemented using conventional floating-point arithmetic, the rounding errors are ignored, so that the result of a higher order of expansion is assumed to be more precise, because the Cauchy estimator of the expansion, which gives an upper bound for the remainder of the expansion, decreases with the order of the expansion for analytic expressions. A subjective upper limit is chosen for the Cauchy estimator, to stop the expansion at limited order [12]. However, such arbitrary upper limit may not be achievable with the amount of rounding errors accumulated during calculation, so that such upper limit may actually gives a false expansion precision.

Using precision arithmetic, the rounding errors as well as the input uncertainties are all accounted for, so that the maximal expansion order when applying a Taylor expansion of Formula (2.46) or Formula (2.51) is no longer subjective. Formula (2.45) is decomposed into the contribution of each successive term for Tylor expansion, as Formula (7.1):

$$(\delta \sum_{j=0}^{J+1} a_j x^j)^2 = \int (\sum_{j=0}^{J} a_j (x+\widetilde{y})^j - \sum_{j=0}^{J} a_j x^j + a_{J+1}(x+\widetilde{y})^{J+1} - a_{J+1}x^{J+1})^2 \rho(\widetilde{y}) d\widetilde{y}$$

$$= \int (\sum_{j=0}^{J} a_j (x+\widetilde{y})^j - \sum_{j=0}^{J} a_j x^j)^2 \rho(\widetilde{y}) d\widetilde{y} + a_{J+1}^2 \int (\sum_{k=1}^{J+1} C_{J+1}^k \widetilde{y}^k x^{J+1-k})^2 \rho(\widetilde{y}) d\widetilde{y}$$

$$+ 2 \int (\sum_{j=0}^{J} a_j \sum_{k=1}^{j} C_j^k \widetilde{y}^k x^{j-k})(a_{J+1} \sum_{k=1}^{J+1} C_{J+1}^k \widetilde{y}^k x^{J+1-k}) \rho(\widetilde{y}) d\widetilde{y}$$

$$(\delta \sum_{j=0}^{J+1} a_j x^j)^2 - (\delta \sum_{j=0}^{J} a_j x^j)^2 = \sum_{k_1=1}^{J+1} \sum_{k_2=1}^{J+1} a_{J+1}^2 C_{J+1}^{k_1} C_{J+1}^{k_2} M(k_1+k_2)(\delta x)^{k_1+k_2} x^{2J+2-k_1-k_2}$$

$$+ 2 \sum_{k_1=1}^{J+1} \sum_{j=0}^{J} \sum_{k_2=1}^{j} a_j a_{J+1} C_{J+1}^{k_1} C_j^{k_2} M(k_1+k_2)(\delta x)^{k_1+k_2} x^{J+1+j-k_1-k_2}; \quad (7.1)$$

Applying Formula (7.1) to Taylor expansion:

1. Formula (7.1) provides the deviation at n-th expansion order, which becomes stabilized when the *delta deviation* at n-th expansion order (which is the contribution of the n-th expansion order to the deviation) is much less than the deviation at n-th expansion order.
2. The *resolution* of precision arithmetic is the deviation divided by 2^χ, in which χ is the constant bits calculated inside uncertainty.
3. The maximal expansion order of a Taylor expansion is reached when the Cauchy estimator is less than the resolution of precision arithmetic, after which the changes in Cauchy estimator is no longer detectable. Ideally, the Taylor expansion reminder should also become zero when the expansion order is larger than the maximal expansion order.

Formula (7.1) also shows that the deviation of Taylor expansion may decrease at certain expansion order. For example, at $x = 1 \pm \delta x$, $1 - 2x + x^2$ is equivalent to y^2 at $y = 0 \pm \delta x$, thus it has smaller result variance than $1 - 2x$ at $x = 1 \pm \delta x$.

Formula (7.2) provides an example test in Taylor expansion, in which n is a positive integer.

$$f_n(x) = \sum_{j=0}^{n} (-x)^j; \quad \lim_{n \to \infty} f_n(x) = 1/(1+x); \tag{7.2}$$

In Formula (7.2), the absolute value of $(n+1)$th term in the expansion is the Cauchy remainder estimator of the nth order expansion. Formula (7.2) is analytic when $|x|$ is less than 1, and a smaller value $|x|$ means faster convergence to the correct value $1/(1+x)$.

Using Formula (7.2) as a test case, Figure 35 confirms the above Taylor expansion process using precision arithmetic with 0-bit calculated inside uncertainty and with input uncertainty at 10^{-3}. For smaller $|x|$, in addition to faster decrease of both reminder and Cauchy estimator, delta deviation also decreases faster, thus deviation reaches its stable values faster. Once the maximal expansion order is reached, the reminder also becomes to zero. Figure 35 repeats the above process with 4-bit calculated inside uncertainty, which only differs from Figure 35 by having resolution smaller than deviation and larger maximal expansion order.

When input has larger uncertainty, deviation reaches to its stable value much slower, which is show in Figure 37 for 0-bit calculated inside uncertainty:

- When $x = 0.75$, deviation barely reaches its stable value when the Cauchy estimator reaches resolution.
- When $x = 0.875$, deviation has not reaches its stable value when the Cauchy estimator reaches resolution, and reminder does not become zero at the maximal expansion order but a few orders beyond.
- When $x = 0.9375$, deviation has no stable value and becomes imaginative eventually. Nevertheless, reminder becomes zero beyond the maximal expansion order.

In contrast, with 4-bit calculated inside uncertainty as shown in Figure 38:

- When $x = 0.75$, the maximal expansion order is reached later when the resolution is stabilized.
- When $x = 0.875$, the maximal expansion order is reached later when the resolution is stabilized, however reminder still does not become zero at the maximal expansion order but a few orders beyond.
- When $x = 0.9375$, resolution has no stable value and becomes negative eventually, after which the precision representation becomes undefined. Because Cauchy estimator never reaches resolution, the maximal expansion order is not defined either.

Judged from the above simple cases of Taylor expansion, calculating inside uncertainty brings no clear-cut benefit.

8 Validation of Precision Arithmetic Using Numerical Integration

In numerical integration over the variable x using conventional floating-point arithmetic, a finer sampling of the function to be integrated $f(x)$ is associated with a better result [12], and it is assumed that $f(x)$ can be sampled at infinitive fine intervals of x. In reality, floating-point arithmetic has limited significant bits, so that rounding errors will increase with finer sampling of $f(x)$. However, such limitation of numerical integration due to rounding errors is seldom studied seriously. In this paper:

1. The function to be integrated is treated as a black-box function.
2. The numerical integration is carried out using the rectangular rule [12].
3. The residual error is estimated locally as the difference between using the rectangular rule and using the trapezoidal rule [12].
4. The sampling is localized using simplest depth-first binary-tree search algorithm.
5. The sampling stops when the residual error is no longer significant.

Specifically, for each integration interval $[x_{start}, x_{end}]$, define:

$$x_{mid} \equiv (x_{start} + x_{end})/2; \tag{8.1}$$

$$f_{err} \equiv (f(x_{start}) + f(x_{end}))/2 - f(x_{mid}); \tag{8.2}$$

$$f_{\triangle} \equiv f(x_{mid})(x_{end} - x_{start}); \tag{8.3}$$

If f_{err} becomes insignificant, the interval $[x_{start}, x_{end}]$ is considered to be fine enough, and $f_{\triangle}$ is added to the total integration. Otherwise, the search continues on the intervals $[x_{start}, x_{mid}]$ and $[x_{mid}, x_{end}]$, which is the next depth for searching. This searching algorithm is very adaptive, with the local search depth depending only on how $f(x)$ changes locally. However, such adaptation to the local change of $f(x)$ brings one weakness to this searching algorithm: when $f(0) = f'(0) = 0$, the algorithm spends the majority of the execution time around $x = 0$, searching in tiny intervals of great depth, and adding tiny significant values to the result each time. This weakness is called zero trap here. It cannot be removed by simply offsetting $f(x)$ by a constant because doing so will change the precision of each sampling of $f(x)$, and increase the output uncertainty deviation. For a proof-of-principle demonstration, zero trap is avoided in this paper.

Formula (8.4) provides an example test for the above simple algorithm, in which n is a positive integer.

$$\frac{4^{n+1} - 10^{-6(n+1)}}{n+1} = \int_{10^{-6}}^{4} x^n dx; \tag{8.4}$$

Table 4 shows that the result of numerical integration is very comparable to the expected value. It shows that the above integration algorithm introduces no broadening of result uncertainty, so the above algorithm always selects optimal integration intervals when calculating the best possible result for a numerical integration. Tests of integration using different polynomials with different integration ranges all confirm the above result.

One thing worth noticing in Table 4 is that even though Formula (8.3) consistently underestimates integration for each integration interval $[x_{start}, x_{end}]$, the final underestimation is quite small and comparable to the uncertainty deviation. This example shows that the bias inside the uncertainty range has insignificant contribution to the final result using precision arithmetic.

Power n	Search Depth	$\delta\left(\int_{10^{-6}}^{4} x^n dx\right)$	$\int_{10^{-6}}^{4} x^n dx - \frac{4^{n+1}-10^{-6(n+1)}}{n+1}$
2	[25, 47]	1.32x10^{-14}	-0.705x10^{-14}
3	[25, 47]	2.52x10^{-14}	-1.42x10^{-14}
4	[26, 47]	1.16x10^{-13}	-1.13x10^{-13}
5	[26, 48]	5.08x10^{-13}	-6.82x10^{-13}
6	[26, 48]	1.92x10^{-12}	-2.72x10^{-12}

Table 4: Uncertainty deviation and value error of numerical integration vs. expected results using precision arithmetic for different power function. The search range is deepest near 10^{-6}.

9 Comparison Using Progressive Moving-Window Linear Regression

9.1 Progressive Moving-Window Linear Regression Algorithm

Formula (9.1) gives the result of the least-square line-fit of $Y = \alpha + \beta X$ between two set of data Y_j and X_j, in which j is an integer index to identify (X, Y) pairs in the sets [12].

$$\begin{aligned} \alpha &= \frac{\sum_j Y_j}{\sum_j 1}; \\ \beta &= \frac{\sum_j X_j Y_j \ \sum_j 1 - \sum_j X_j \ \sum_j Y_j}{\sum_j X_j X_j \ \sum_j 1 - \sum_j X_j \ \sum_j X_j}; \end{aligned} \tag{9.1}$$

In many applications data set Y_j is an input data stream collected with fixed rate in time, such as a data stream collected by an ADC (Analogue-to-Digital Converter) [5]. Y_j is called a time-series input, in which j indicates time. A moving window algorithm [12] is performed in a small time-window around each j. For each window of calculation, X_j can be chosen to be integers in the range of $[-H, +H]$ in which H is an integer constant specifying windows half width so that $\sum_j X_j = 0$, to reduce (9.1) into (9.2):

$$\begin{aligned} \alpha_j &= \alpha\ 2H = \sum_{X=-H+1}^{H} Y_{j-H+X}; \\ \beta_j &= \beta\ \frac{H(H+1)(2H+1)}{3} = \sum_{X=-H}^{H} X Y_{j-H+X}; \end{aligned} \tag{9.2}$$

According to Figure 39, in which H takes an example value of 4, the calculation of (α_j, β_j) can be obtained from the previous values of $(\alpha_{j-1}, \beta_{j-1})$, to reduce the calculation of (9.2) into a progressive moving-window calculation of (9.3):

$$\begin{aligned} \beta_j &= \beta_{j-1} - \alpha_{j-1} + H(Y_{j-2H-1} + Y_j); \\ \alpha_j &= \alpha_{j-1} - Y_{j-2H-1} + Y_j; \end{aligned} \tag{9.3}$$

9.2 Dependency Problem in a Progressive Algorithm

(9.3) uses each input multiple times, so it will have dependency problem for all the three uncertainty-bearing arithmetic. The question is how the overestimation of uncertainty evolves with time.

The moving-window linear regression is done on a straight line with a constant slope of exactly 1/1024 for each advance of time, with a full window width of 9 data points, or $H = 4$. Both average output value errors and deviations of all three arithmetic increases linearly with input deviations, and increase monotonically with time. Thus both the average output tracking ratio and the maximal output bounding ratio are largely independent of input precisions, e.g., Figure 40 shows such trend for the average output tracking ratio using precision arithmetic. Such independence to input precision is expected for linear algorithms in general [12]. Therefore, only results with the input deviation of 10^{-3} are shown for the remaining discussions unless otherwise specified. Figure 41 shows the output deviation and the value errors vs. time while Figure 42 shows the output average tracking ratios and the maximal bounding ratios vs. time for all three arithmetics.

For interval arithmetic and independence arithmetic, the output value errors remain on a constant level, while the output deviations increase with time, so that both output average tracking ratios and maximal bounding ratios decrease with time. The stable linear increase of output deviation with time using either interval arithmetic or independence arithmetic in Figure 41 suggests that the progressive linear regression calculation has accumulated every input uncertainty, which results in the monotonic decrease of both the maximal bounding ratios and the average output tracking ratios with time using both arithmetics in Figure 42.

In contrast, while precision arithmetic has slightly larger output deviations than those of independence arithmetic, its output value errors follows its output deviations, so that both its tracking ratios and bounding ratios remain between 0.1 and 0.9. The reason for such increase of output value errors with time is due to the fact that precision arithmetic calculates only limited bits inside uncertainty, and uses larger granularity of values in calculation for larger uncertainty deviation. Such granularity of calculation is evident when comparing 2-bit or 4-bit calculation inside uncertainty using precision arithmetic in Figure 41. This mechanism of error tracking in precision arithmetic is also demonstrated in Figure 43 and Figure 44. Figure 43 shows that for fewer bits calculated inside uncertainty, the output value errors follow the output deviation closer in time, but such usage of larger granularity of values in calculation causes the result to become insignificant sooner, while for more bits calculated inside uncertainty, the average tracking ratios initially follow the result using independence arithmetic longer, and then follow the output deviation for longer duration. The similarity in patterns of the average tracking ratios for different bits calculated inside uncertainty using precision arithmetic in Figure 43 suggests that they are all driven by a same mechanism but on different time scale, which is expected when smaller granularity of error needs more time to accumulate to a same level. From the definition of tracking ratio, the granularity of error is actually measured in term of granularity of precision, e.g., Figure 44 shows that for same bits calculated inside uncertainty, smaller input uncertainty deviations results in longer tracking of the output value errors to the output deviations. The similar pattern of average tracking ratios is repeated on slower time scale for smaller input uncertainty deviations in Figure 44, revealing similar underline error-tracking mechanism in both cases. Figure 44 also shows that for the same bits calculated inside uncertainty, the average tracking ratios deviate from independence at exactly the same time. Figure 43 and Figure 44 thus demonstrate a uniform and granular error tracking mechanism of the precision arithmetic for different bits calculated inside uncertainty.

Is such increase of the value errors with the increase of uncertainty deviation using precision arithmetic desired? First, in real calculations the correct answer is not known, and the reliability of a result depends statistically on the uncertainty of the result, so that there is no reason to assume that calculating more bits inside uncertainty is any better. Conceptually, when the uncertainty of a calculation increases, the value error of the calculation is also expected to increase, which agrees with the trend shown by precision arithmetic. Second, the stability of the average output tracking ratios and the maximal bounding ratios of precision arithmetic is quite valuable in interpretation results. For example, even the output deviation may have unexpectedly changed, as in this case if dependency problem were not known and expected, such stability still gives a good estimation of the value errors in the result using precision arithmetic. Third, such stability ensures that the result of algorithm at each window does not depend strongly on the usage history of the algorithm, which makes precision arithmetic the only practically usable uncertainty-bearing arithmetic for this progressive algorithm. To test the effect of usage history on each uncertainty-bearing arithmetic, noise is increased by 10-fold at the middle 1/3 duration of the straight line, to result in additional two test cases:

- *Changed*: In Figure 45 and 47, the input deviation is also increased by 10-fold to simulate an increase in measured uncertainty.
- *Defective*: In Figure 46 and 48, the input deviation remains the same to simulate the defect in obtaining the uncertainty deviations.

Accordingly, the original case of linear regression on a line with fixed slope is named as *Simple*.

The question is how each uncertainty-bearing arithmetic responses to this change of data in the last 1/3 duration of calculation. Using either independence or interval arithmetic, both the average output tracking ratios and the maximal output bounding ratios are decrease by about 10-fold in Figure 47 while they are not affected at all in Figure 48. They show extreme sensitivity to the usage history. Because the real input data are neither controllable nor predictable, the result uncertainty for this progressive algorithm using either interval arithmetic or independence arithmetic may no longer be interpretable. In contrast, using precision arithmetic, both the average output tracking ratios and the maximal output bounding ratios are relatively stable, while the output deviations and value errors are sensitivity to usage history, so that the result using precision arithmetic is still interpretable.

9.3 Choosing a Better Algorithm for Imprecise Inputs

Formula (9.3) has much less calculations than Formula (9.2), and it seems a highly efficient and optimized algorithm according to conventional criterion [12]. However, from the perspective of uncertainty-bearing arithmetic, Formula (9.3) is progressive while Formula (9.2) is expressive, so that Formula (9.2) should be better. Figure 45 and Figure 49 respectively show the output deviations and the value errors vs. time for using either Formula (9.3) or Formula (9.2) of a straight line with 10-fold increase of input uncertainty in the middle 1/3 duration. They show that while the progressive algorithm carries all the historical calculation uncertainty into future, the expressive algorithm is clean from any previous results. For example, at the last 1/3 duration when the moving window is already out of the area for the larger input uncertainty, the progressive algorithm still gives large result uncertainty, while the expressive algorithm gives output result only relevant to the input uncertainty within the moving window. So instead of Formula (9.3), Formula (9.2) is confirmed to be a better solution for this linear regression problem.

9.4 Modelling Dependency Problem

However, the majority algorithms used today are progressive. Most practical problems are not even mathematical and analytical in nature, so that they may have no expressive solutions. Expressive algorithms are simply just not always avoidable in practice.

With known expressive counterpart, the progressive moving-window linear regression algorithm can serve as a model for studying progressive algorithms. For example:

- The progressive moving-window linear regression shows that the dependency problem of independence and interval arithmetic can manifest as dependency on the usage history of an algorithm. Because of its stability, precision arithmetic should be used generally in progressive algorithms.
- Figure 51 shows that the result tracking ratios of the progressive linear regression is exponentially distributed, while Figure 52 shows that the result tracking ratios of the expressive linear regression is Gaussian distributed only when the uncertainty deviation is characterized correctly, e.g., the result is Gaussian distributed for the "Changed" case but not for the "noisier" case. Thus, the exponentially distributed tracking ratios does not necessarily imply dependency problem.

10 Conclusion and Discussion

10.1 Summary

The starting point of precision arithmetic is the uncorrelated uncertainty assumption, which requires input data to have decent precision for each or small overall correlation among them, as shown in Figure 2, which quantifies the statistical requirements for input data to precision arithmetic. In addition, it requires that the systematic errors is not the major source of uncertainty, and all of its input data do not have confused identities.

Due to the uncorrelated uncertainty assumption and central limit theorem, the rounding errors of precision arithmetic are shown to be bounded by a Gaussian distribution with a truncated range. The rounding error distribution is extended to describe the uncertainty distribution in general, with the uncertainty deviation of a single precision value given by Formula (2.15), and the result uncertainty deviation of a function given by Formula (2.47) and its multi-dimension extensions such as Formula (2.52).

Formula (4.14) is shown to describe the general uncertainty deviation propagation in precision arithmetic. The average tracking ratios and the maximal bounding ratio using precision arithmetic are shown to be independent of input precision, and stable for the amount of calculations for a few very different applications. In contrast, both average tracking ratios and the maximal bounding ratio using interval arithmetic are shown to decrease exponentially with the amount of calculations in all tests. Such stability is the major reason why precision arithmetic is better than interval arithmetic in all tests done so far.

The statistical nature of precision arithmetic provides not only quantitative explanation for the dependency problem, but also solutions to the dependency problem, which is in form of either Taylor expansion or calibration. The treatment of dependency problem is another major advantage of precision arithmetic over interval arithmetic.

Statistical precision has a central role in precision arithmetic:

- Precision is regarded as information content of a uncertainty-bearing value, which is in par with information entropy in information theory. Because of this, precision needs to be preserved when the uncertainty-bearing value is multiplied or divided by a constant, which results in the scaling principle.
- Precision arithmetic itself can be deduced from the scaling principle and the uncorrelated uncertainty assumption.
- The convergence property of the result deviation using Taylor expansion method is determined by input precisions, such as for inversions and square roots.

10.2 Efficiency of Precision Arithmetic

Precision arithmetic tries to solve a different mathematical problem from conventional floating-point arithmetic. For example, to calculate the determinant of a matrix:

- Conventional floating-point arithmetic may use a Laplace method [12], namely, to randomly choose a row or a column, and then to sum up the products of each element within the chosen row or the column with the corresponding sub-determinant of the element. Each sub-determinant is calculated in the same fashion. Depending on the choices of the row or the column in each stage, there are many paths to calculate the determinant of a matrix. Because conventional floating-point arithmetic has unbounded rounding errors, each path may give a different result, and the spread of all the results depends on the stability of the matrix and each sub-matrix [44]. In this perspective, by taking a random path and assuming to get the only correct result, conventional floating-point arithmetic can be viewed as a leap-of-faith approach.

- In contrast, precision arithmetic also needs to calculate the spread of the result due to rounding error or input uncertainties, so it effectively has to cover all paths of the calculation. For example, using Formula (5.11), precision arithmetic starts from each elements of the matrix, and treat it as a 1x1 sub-determinant, then grow it to all possible 2x2 sub-determinants containing it, etc, until reach the determinant of the matrix. Thus, precision arithmetic takes order-of-magnitude more time than a single leap-of-faith calculation.

However, it is wrong to conclude that precision arithmetic is less efficient than conventional floating-point arithmetic, because in most cases rounding errors and input uncertainty can not be ignored. Because conventional floating-point arithmetic can not contain uncertainty in its value, it has to use another value to specify uncertainty, such as an interval of $[min, max]$ or a common statistical pair $value \pm deviation$, which may brings the following drawbacks:

- The most common way to calculate result spread using conventional floating-point arithmetic is sampling [15] [12]. Assuming the matrix size is $N \times N$, and a minimal 3-point sampling is performed on each matrix element, then the spread calculation of matrix determinant requires N^6 leap-of-faith calculations, which is still a lot. In contrast, using Formula (5.11), precision arithmetic only need one calculation. Thus, conventional floating-point arithmetic may be less efficient than precision arithmetic in this context.
- During to unbounded rounding errors, a conventional floating-point value losses its precision gradually and silently, so that a interval or a statistical pair itself can become unknowingly invalid. At least, it is not clear at what precision the interval or the statistical pair specifies.

10.3 Choose a better algorithm

Because precision arithmetic tries to solve a different problem than conventional floating-point arithmetic, it has completely different criterion when choosing algorithms or implementations of algorithms. For example, for matrix inversion, because conventional floating-point arithmetic has unbounded rounding errors, it will choose certain flavour of LU-decomposition over Gaussian elimination and determinant division [12]. The result difference of LU-decomposition, Gaussian elimination and determinant division shows that conventional floating-point arithmetic has strong dependency problem, which has been a way of life when using conventional floating-point arithmetic, e.g., different algorithms or different implementation of the same algorithm are expected to give different results, of which a best algorithm or implementation is always chosen for each usage context [12], even though they may be mathematically equivalent. In contrast, rounding errors are bounded in both precision arithmetic and interval arithmetic [19], so they are no longer needed to be considered. When interval arithmetic reformat a numerical question as "Given each input to be an interval, what the output would be?", it effectively states that the results for most numerical questions to be solved should be *unique* to be either one or a few intervals that tightest bounds the results, *regardless* of the algorithm to be used, *unless* dependency problem is introduced in the implementation of an algorithm. Same concept is true for precision arithmetic, which converges all input uncertainty distribution to *ubiquitously Gaussian* at the outputs, and which further *quantifies* the source of the dependency problem. Using precision arithmetic instead of conventional floating-point arithmetic, the focus has shifted from minimizing rounding errors to minimizing dependency problem. Of the three algorithms for matrix inversion, both LU-decomposition and Gaussian elimination are progressive, which means that each input may appear multiple times in different branch at different time, whose dependency problem is difficult to quantified. On the other hand, a determinant of a $N \times N$ matrix can be treated as a N-order polynomial with N^2 variables, to be readily for the Taylor expansion, which results in Formula (5.11), so that the determinant division method is chosen in this paper for matrix inversion. For the same reason, in the moving-window linear regression, the worse method in conventional floating-point arithmetic, Formula (9.2), becomes the better method in precision arithmetic, and vice versa.

Due to the requirement of minimizing dependence problem, precision arithmetic has much less operational freedom than conventional arithmetic and may require extensive symbolic calculations, following practices in affine arithmetic [40]. Also, the comparison relation in conventional arithmetic needs to be re-evaluated in precision arithmetic, which brings about another reason for different algorithm selection.

10.4 Improving Precision Arithmetic

Figure 2 uses a cut-off for the test of the uncorrelated uncertainty assumption among two uncertainty-bearing values. A better approach is to associate the amount of the dependence problem with the amount of correlation between the uncertainties of the two values.

There are actually three different ways to round up $(2S+1)?4R\ 2^E$:

1. always round up $(2S+1)?4R\ 2^E$ to $(S+1) - R\ 2^{E+1}$;
2. always round up $(2S+1)?4R\ 2^E$ to $S + R\ 2^{E+1}$;
3. randomly round up $(2S+1)?4R\ 2^E$ to either $(S+1) - R\ 2^{E+1}$ or $S + R\ 2^{E+1}$.

The first method results in slightly slower loss of significand than the second method, while the third method changes precision arithmetic from deterministic to stochastic. Because no empirical difference has been detected among these three different rounding up methods, the first method is chosen in this paper. Further study is required to distinguish the different rounding up methods.

The objectives of precision arithmetic need to be studied further. For example, Formula (2.44) has rejected the effect of uncertainty on the expected value by incorporating the value shift due to uncertainty as increase of variance, such as in the case of calculating $f(x) = x^2$. The effect of such asymmetrical broadening is unclear.

The number of bits to be calculated inside uncertainty also needs to be studied further. For example, when limited bits are calculated inside uncertainty, adding insignificant higher order term of a Taylor expansion may decrease the value error while increasing the uncertainty deviation, which may call for an optimal bits to be calculated inside uncertainty for the truncation rule.

Because precision arithmetic is based on generic concepts, it is targeted to be a generic arithmetic for both uncertainty-tracking and uncertainty-bounding. However, it seems a worthwhile alternative to interval arithmetic and the de facto independence arithmetic. Before applying it generally, precision arithmetic still needs more groundwork and testing. It should be tested further in other problems such as improper integrations, solutions to linear equations, and solutions to differential equations.

10.5 Acknowledgements

As an independent researcher, the author of this paper feels indebted to encouragements and valuable discussions with Dr. Zhong Zhong from Brookhaven National Laboratory, Prof. Hui Cao from Yale University, Dr. Anthony Begley from *Physics Reviews B*, the organizers of *AMCS 2005*, with Prof. Hamid R. Arabnia from University of Georgia in particular, and the organizers of *NKS Mathematica Forum 2007*, with Dr. Stephen Wolfram in particular. Finally, the author of this paper is very grateful for the editors and reviewers of *Reliable Computing* for their tremendous help in shaping this unusual paper from unusual source, with managing editor, Prof. Rolph Baker Kearfott in particular.

References

[1] Sylvain Ehrenfeld and Sebastian B. Littauer. *Introduction to Statistical Methods.* McGraw-Hill, 1965.

[2] John R. Taylor. *Introduction to Error Analysis: The Study of Output Precisions in Physical Measurements.* University Science Books, 1997.

[3] Jurgen Bortfeldt, editor. *Fundamental Constants in Physics and Chemistry.* Springer, 1992.

[4] Michael J. Evans and Jeffrey S. Rosenthal. *Probability and Statistics: The Science of Uncertainty.* W. H. Freeman, 2003.

[5] Paul Horowitz and Hill Winfield. *Art of Electronics.* Cambridge Univ Press, 1995.

[6] Fixed-point arithmetic. `http://en.wikipedia.org/wiki/Fixed-point_arithmetic`, 2011. wikipedia, the free encyclopedia.

[7] Arbitrary-precision arithmetic. `http://en.wikipedia.org/wiki/Arbitrary-precision_arithmetic`, 2011. wikipedia, the free encyclopedia.

[8] John P Hayes. *Computer Architecture.* McGraw-Hill, 1988.

[9] David Goldberg. What every computer scientist should know about floating-point arithmetic. *ACM Computing Surveys*, March 1991.

[10] Institute of Electrical and Electronics Engineers. *ANSI/IEEE 754-2008 Standard for Binary Floating-Point Arithmetic*, 2008.

[11] U. Kulish and W.M. Miranker. The arithmetic of digital computers: A new approach. *SIAM Rev.*, 28(1), 1986.

[12] William H. Press, Saul A Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes in C.* Cambridge University Press, 1992.

[13] Oliver Aberth. *Precise Numerical Methods Using C++.* Academic Press, 1998.

[14] Gregory L. Baker and Jerry P. Gollub. *Chaotic Dynamics: An Introduction.* Cambridge University Press, 1990.

[15] J. Vignes. A stochastic arithmetic for reliable scientific computation. *Mathematics and Computers in Simulation*, 35:233–261, 1993.

[16] B. Liu and T. Kaneko. Error analysis of digital filters realized with floating-point arithmetic. *Proc. IEEE*, 57:p1735–1747, 1969.

[17] B. D. Rao. Floating-point arithmetic and digital filters. *IEEE, Transations on Signal Processing*, 40:85–95, 1992.

[18] R.E. Moore. *Interval Analysis.* Prentice Hall, 1966.

[19] W. Kramer. A prior worst case error bounds for floating-point computations. *IEEE Trans. Computers*, 47:750–756, 1998.

[20] G. Alefeld and G. Mayer. Interval analysis: Theory and applications. *Journal of Computational and Applied Mathematics*, 121:421–464, 2000.

[21] W. Kramer. Generalized intervals and the dependency problem. *Proceedings in Applied Mathematics and Mechanics*, 6:685–686, 2006.

[22] A. Neumaier S.M. Rump S.P. Shary B. Kearfott, M. T. Nakao and P. Van Hentenryck. Standardized notation in interval analysis. *Computational Technologies*, 15:7–13, 2010.

[23] W. T. Tucker and S. Ferson. *Probability bounds analysis in environmental risk assessments.* Applied Biomathmetics, 100 North Country Road, Setauket, New York 11733, 2003.

[24] J. Stolfi and L. H. de Figueiredo. An introduction to affine arithmetic. *TEMA Tend. Mat. Apl. Comput.*, 4:297–312, 2003.

[25] R. Alt and J.-L. Lamotte. Some experiments on the evaluation of functional ranges using a random interval arithmetic. *Mathematics and Computers in Simulation*, 56:17–34, 2001.

[26] J. Stolfi and L. H. de Figueiredo. *Self-validated numerical methods and applications.* ftp:// ftp.tecgraf.puc-rio.br/pub/lhf/doc/cbm97.ps.gz, 1997.

[27] Propagation of uncertainty. `http://en.wikipedia.org/wiki/Propagation_of_uncertainty`, 2011. wikipedia, the free encyclopedia.

[28] S. Ferson H. M. Regan and D. Berleant. Equivalence of methods for uncertainty propagation of real-valued random variables. *International Journal of Approximate Reasoning*, 36:1–30, 2004.

[29] C. P. Robert. *Monte Carlo Statistical Methods.* Springer, 2001.

[30] Monte carlo method. `http://en.wikipedia.org/wiki/Monte_Carlo_method`, 2011. wikipedia, the free encyclopedia.

[31] C. L. Smith. Uncertainty propagation using taylor series expansion and a spreadsheet. *Journal of the Idaho Academy of Science*, 30-2:93–105, 1994.

[32] Significance arithmetic. `http://en.wikipedia.org/wiki/Significance_arithmetic`, 2011. wikipedia, the free encyclopedia.

[33] M. Goldstein. Significance arithmetic on a digital computer. *Communications of the ACM*, 6:111–117, 1963.

[34] R. L. Ashenhurst and N. Metropolis. Unnormalized floating-point arithmetic. *Journal of the ACM*, 6:415–428, 1959.

[35] G. Spaletta M. Sofroniou. Precise numerical computation. *The Journal of Logic and Algebraic Programming*, 65:113134, 2005.

[36] C. Denis N. S. Scott, F. Jezequel and J. M. Chesneaux. Numerical 'health' check for scientific codes: the cadna approach. *Computer Physics Communications*, 176(8):501–527, 2007.

[37] C. P. Wang. Error estimation of floating-point calculations by a new floating-point type that tracks the errors. In H. R. Arabnia and I. A. Ajwa, editors, *Proceedings of the 2005 International Conference on Algorithmic Mathematics and Computer Science, AMCS 2005*, pages 84–92, 2005.

[38] A. Feldstein and R. Goodman. Convergence estimates for the distribution of trailing digits. *Journal of the ACM*, 23:287–297, 1976.

[39] Double factorial. `http://mathworld.wolfram.com/DoubleFactorial.html`, 2014. Wolfram MathWorld.

[40] C. Pennachin, M. Looks, and Joo A. de Vasconcelos. Robust symbolic regression with affine arithmetic. In *Proceedings of the 12th annual conference on Genetic and evolutionary computation (2010)*, pages 917–924, 2010.

[41] N. Beaudoin and S. S. Beauchemin. A new numerical fourier transform in d-dimensions. *IEEE Transactions on Signal Processing*, 51-5:1422–1430, 2003.

[42] Digital signal processor. `http://en.wikipedia.org/wiki/Digital_signal_processor`, 2011. wikipedia, the free encyclopedia.

[43] J. Hefferon. Linear algebra. `http://joshua.smcvt.edu/linearalgebra/`, 2011.

[44] Condition number. `http://en.wikipedia.org/wiki/Condition_number`, 2011. wikipedia, the free encyclopedia.

[45] Hilbert matrix. `http://en.wikipedia.org/wiki/Hilbert_matrix`, 2011. wikipedia, the free encyclopedia.

[46] Big o notation. `http://en.wikipedia.org/wiki/Big_Oh_notation`, 2011. wikipedia, the free encyclopedia.

11 Figures

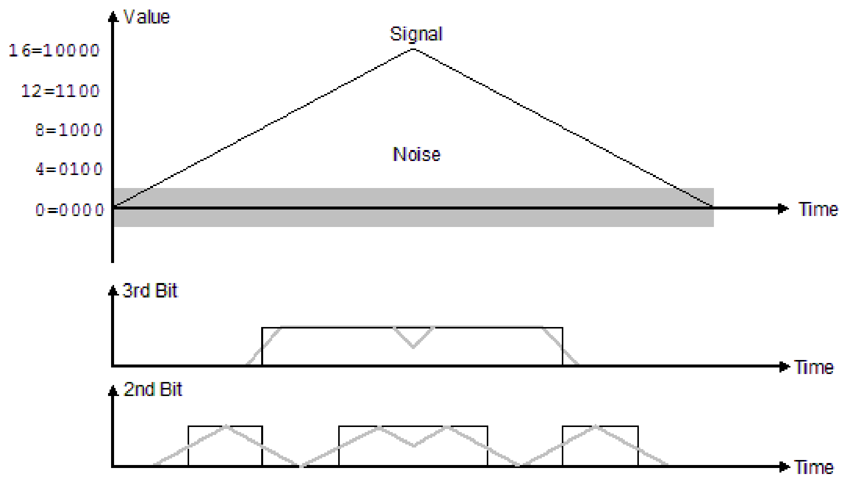

Figure 1: Effect of noise on bit values of a measured value. The triangular wave signal and the added white noise are shown at top using the thin black line and the grey area, respectively. The values are measured by a theoretical 4-bit Digital-to-Analog Converter in ideal condition, assuming LSB is the 0th bit. The measured 3rd and 2nd bits without the added noise are shown using thin black lines, while the mean values of the measured 3rd and 2nd bits with the added noise are shown using thin grey lines.

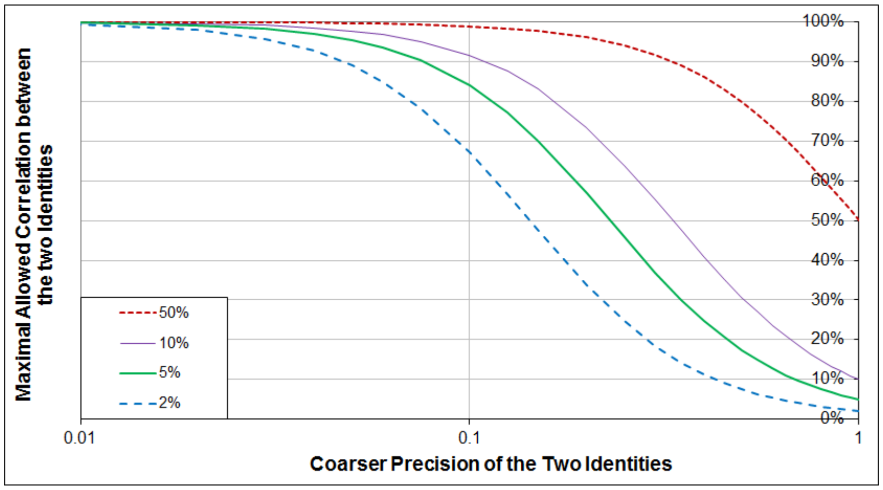

Figure 2: Allowed maximal correlation between two values vs. input precisions and independence standard (as shown in legend) for the independence uncertainty assumption of precision arithmetic to be true.

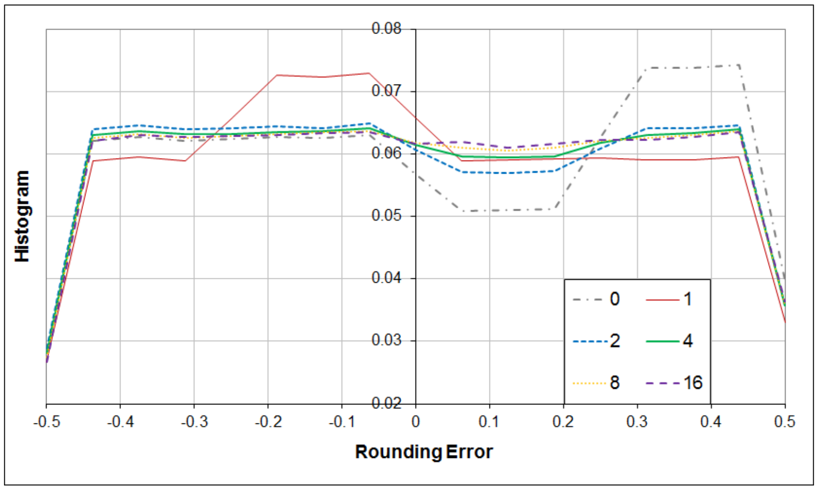

Figure 3: Measured probability distribution of rounding errors of precision round-up rule for the minimal significand thresholds 0, 1, 2, 4, and 8 respectively. Mathematically the probability is usually defined either in range (-1/2, +1/2] or in range [-1/2, +1/2), but not in range [-1/2, +1/2]. Because -1/2 and +1/2 in bounding range have different meaning in precision representation, the probability range is defined as [-1/2, +1/2], which introduces the artificially smaller count of histogram in sections containing either -1/2 or +1/2.

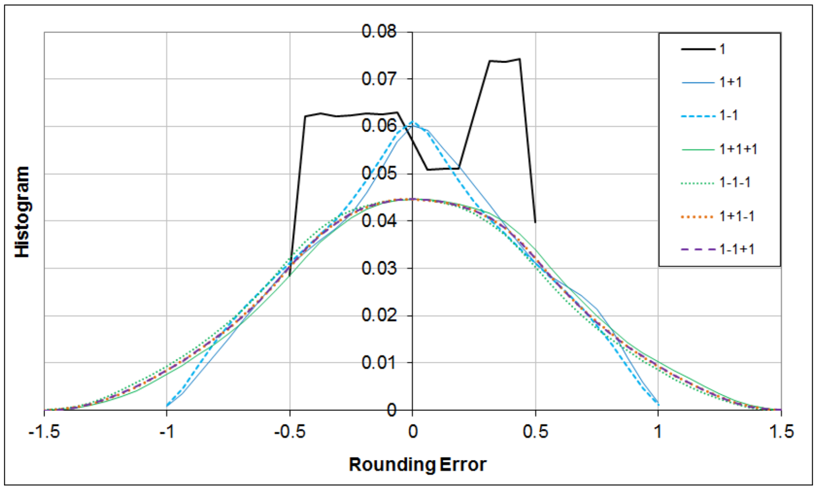

Figure 4: Measured probability distribution of the rounding error after addition and subtraction. In the legend, "1" for measured rounding error distribution for the minimal significand thresholds 0, "1+1" for addition once and "1-1" for subtraction once using the rounding error distribution of "1", while "1+1+1" for addition twice, "1-1-1" for subtraction twice, "1+1-1" for addition once then subtraction once, and "1-1+1" for subtraction once then addition once.

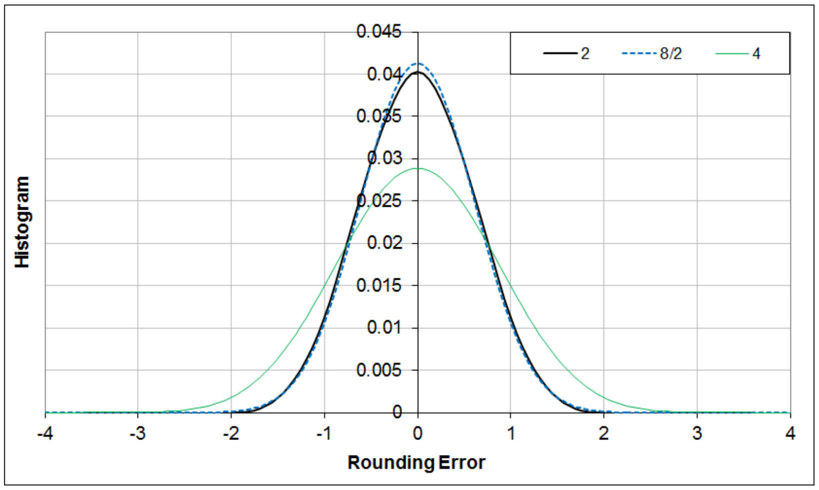

Figure 5: The result rounding error distribution $R = 8/2$ after the original error distribution $R = 8$ is rounded up once. The $R = 8/2$ distribution is compared with the $R = 4$ distribution and the $R = 2$ distribution, which have the same bounding range and deviation, respectively.

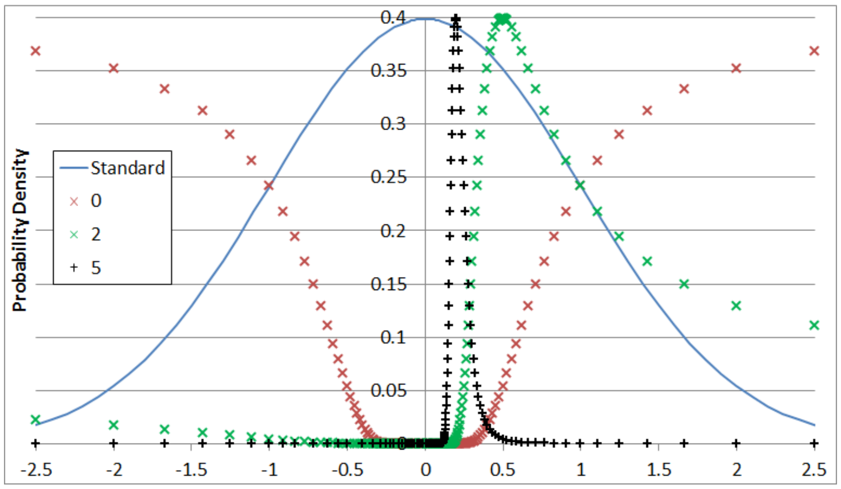

Figure 6: The probability density function for $1/(x + \delta x)$ in which $\delta x = 1$ and $x = 0, 2, 5$ respectively, as shown in the inlet. For comparison, the density function of a standard distribution is shown.

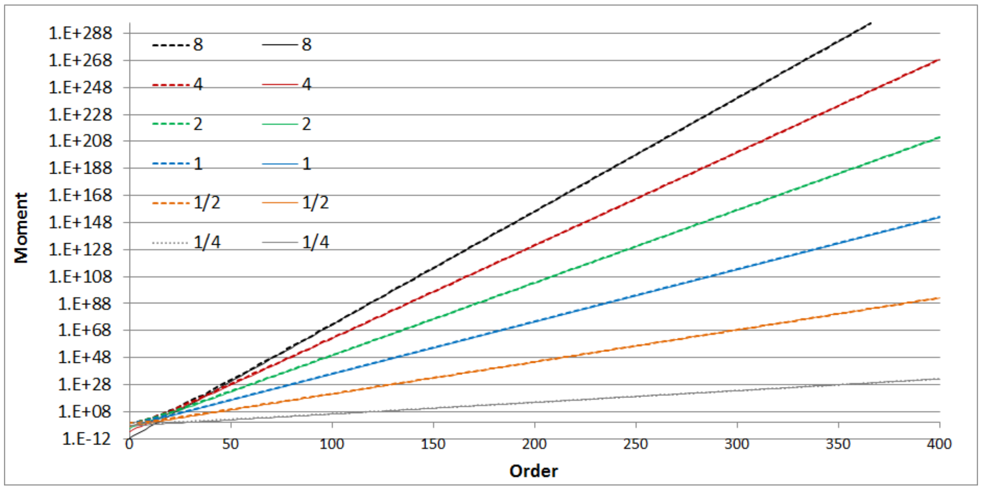

Figure 7: The moments up to 200-order for rounding error distribution for different bounding range $\hat{R}$ of the legend. The moments for each bounding range $\hat{R}$ are drawn in one color using dashed line, and exponentially fitted, whose fitting lines are displayed in the same color using thin solid lines.

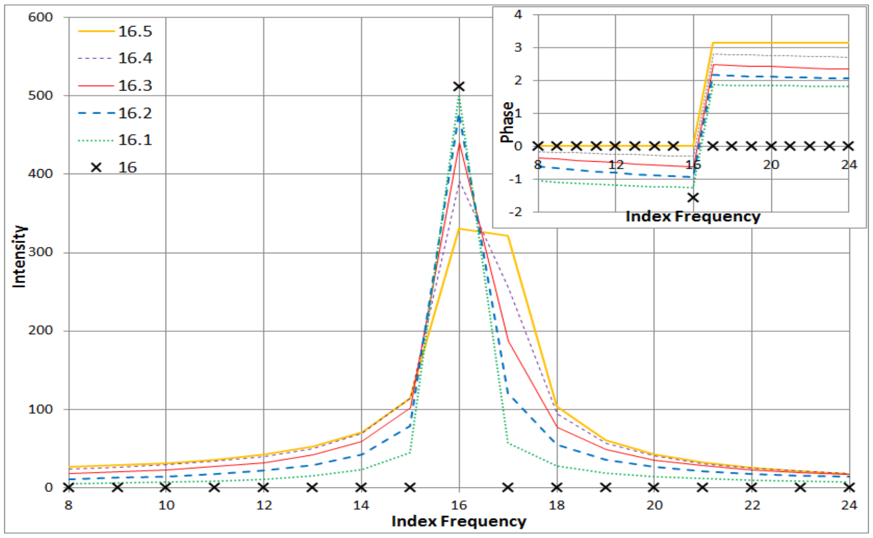

Figure 8: Unfaithful representations of perfect sine signals in the Discrete Fourier Transformation. The calculation is done on 1024 samples using FFT on a series of perfect sine signals having amplitude of 1 and slightly different frequencies as shown in legends. In the drawing, x axis shows frequency, y axis shows either intensity or phase (inlet). A faithful representation is also included for comparison, whose phase is $\pi/2$ at the index frequency, and undetermined at other frequencies.

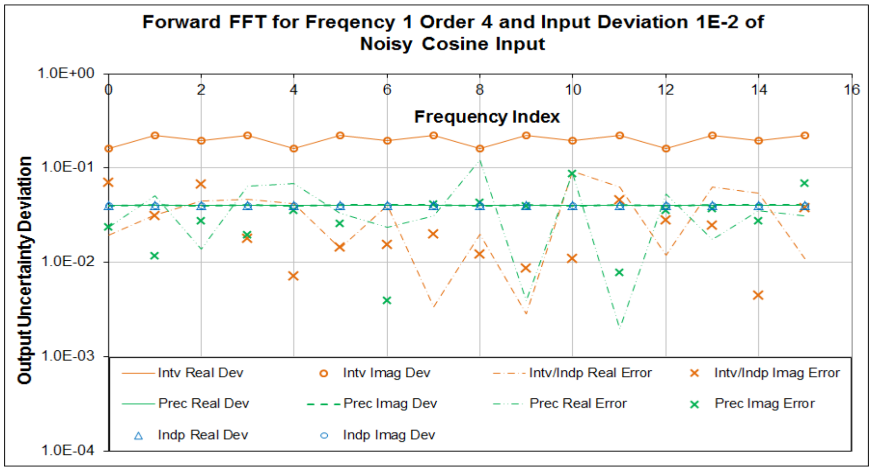

Figure 9: The output deviations and value errors of the forward FFT on a noisy sine signal of FFT order 4, index frequency 1 and input deviation 10^{-2}. In the legend, ”Intv” means interval arithmetic, ”Indp” means independence arithmetic, ”Prec” means precision arithmetic, “Dev” means output uncertainty deviations, “Error” means output value errors, “Real” means real part, and “Imag” means imaginary part. Because both interval arithmetic and independence arithmetic using conventional floating arithmetic for underlying calculations, they have the same value errors.

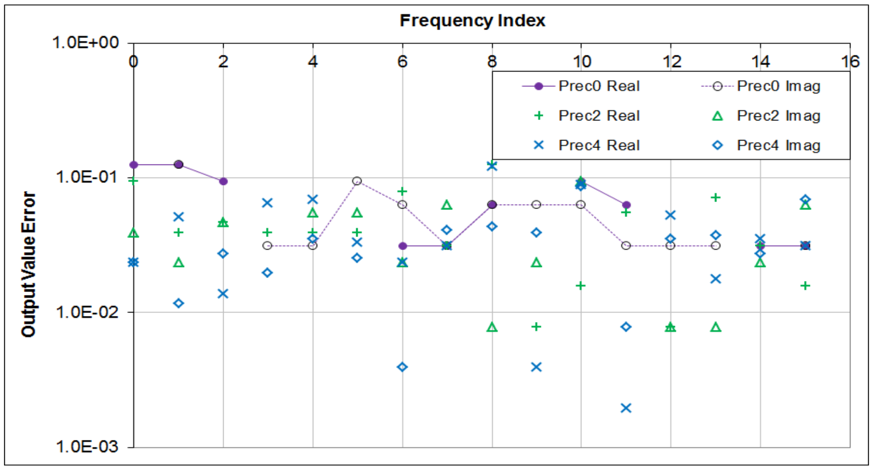

Figure 10: The output value errors of the forward FFT on a noisy sine signal of index frequency 1 and input deviation 10^{-2} using precision arithmetic with different bit inside uncertainty. In the legend, “Prec0” means precision arithmetic with 0-bit calculated inside uncertainty, “Prec2” means precision arithmetic with 2-bit calculated inside uncertainty, and ”Prec4” means precision arithmetic with 4-bit calculated inside uncertainty.

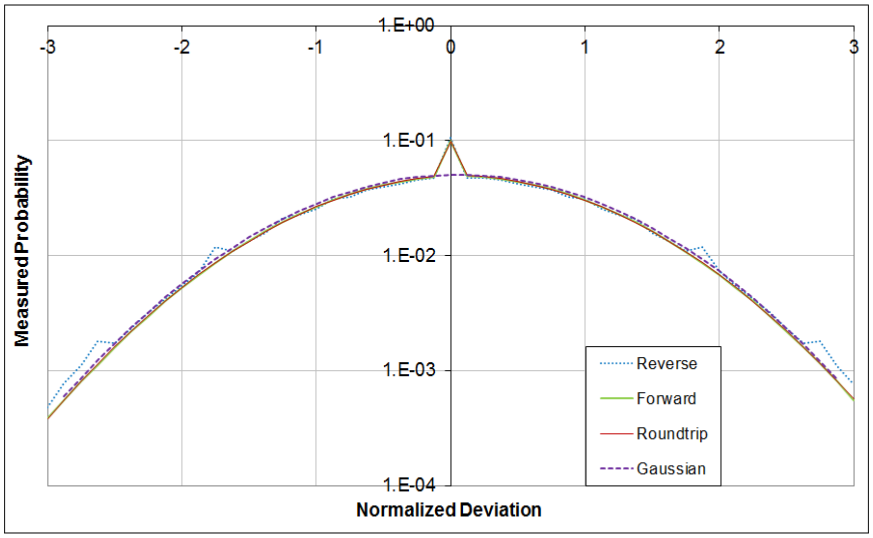

Figure 11: The measured tracking ratio distributions using independence arithmetic for FFT algorithms (as shown in legend). They are best fitted by a Gaussian distribution with the mean of 0.06 and deviation of 0.98.

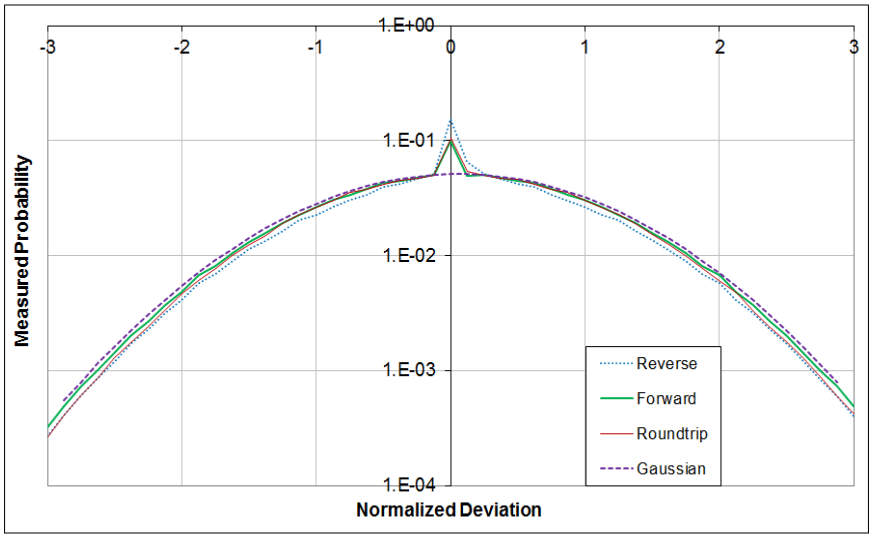

Figure 12: The measured tracking ratio distributions using precision arithmetic for FFT algorithms (as shown in legend). They are best fitted by a Gaussian distribution with the mean of 0.06 and deviation of 0.97.

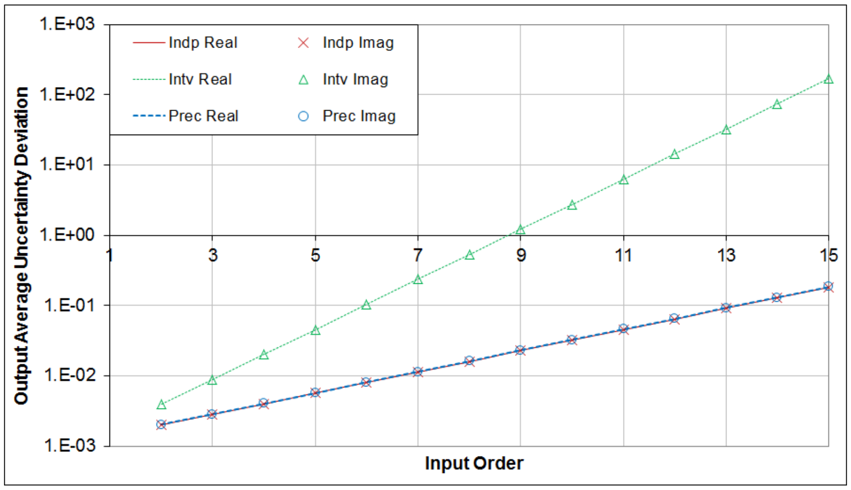

Figure 13: For the same input deviation of 10^{-3}, the empirical average output deviations of the forward FFT increase exponentially with the FFT order for all uncertainty-bearing arithmetics. In the legend, "Intv" means interval arithmetic, "Indp" means independence arithmetic, "Prec" means precision arithmetic, "Real" means real part, and "Imag" means imaginary part.

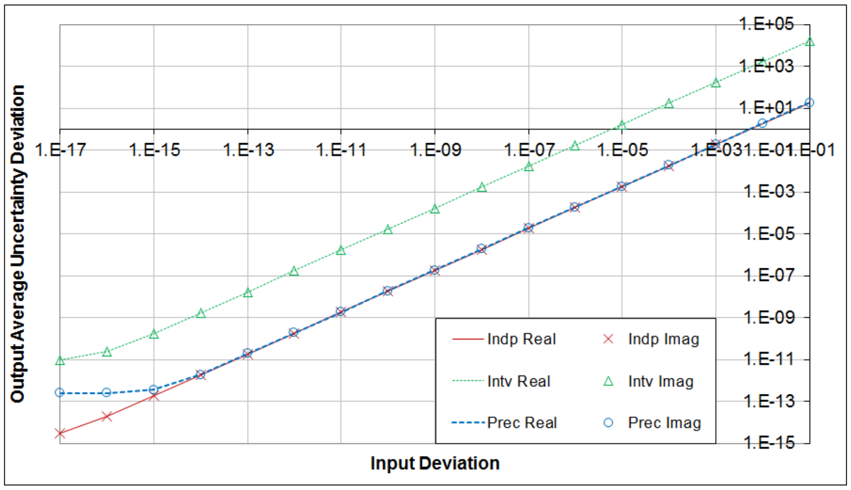

Figure 14: For the same order of the FFT calculation of 15, the empirical average output deviations of the forward FFT increases linearly with the input deviation for all uncertainty-bearing arithmetics. In the legend, "Intv" means interval arithmetic, "Indp" means independence arithmetic, "Prec" means precision arithmetic, "Real" means real part, and "Imag" means imaginary part.

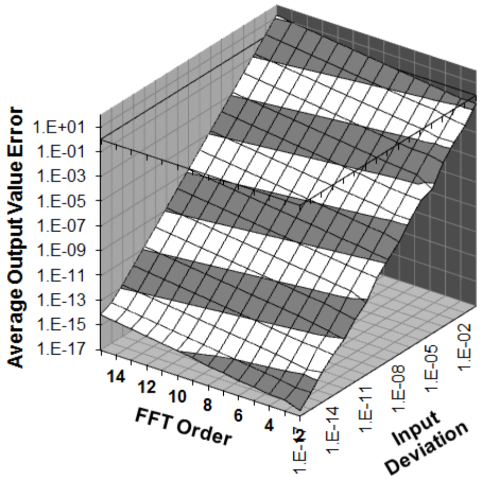

Figure 15: The empirical average output value errors using precision arithmetic increase exponentially with the FFT order and linearly with the input deviation, respectively.

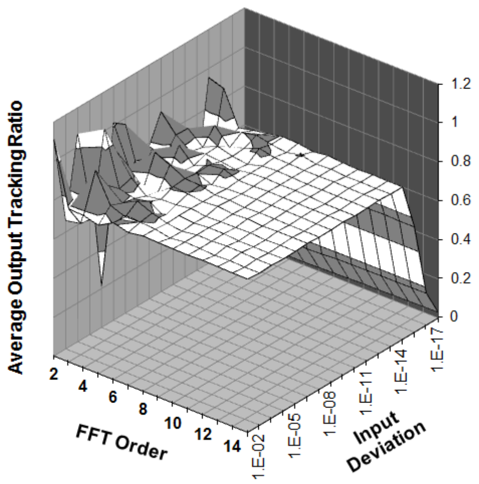

Figure 16: The empirical average output tracking ratios using precision arithmetic is a constant when the input deviation is larger than 10^{-14} and the FFT order is more than 5 for forward FFT algorithms. Because the precision of conventional floating-point representation is at 10^{-16}, adding Gaussian noises with the deviation of 10^{-17} should have little effect on the input data. For the same reason, the output tracking ratios are stable only when the input deviation is more than 10^{-14}. When the FFT order is 2, a FFT calculation actually involves no arithmetic calculation between input data. For the same reason, when the FFT order is less than 5, there is not enough arithmetic calculation for the result tracking ratios to reach equilibrium.

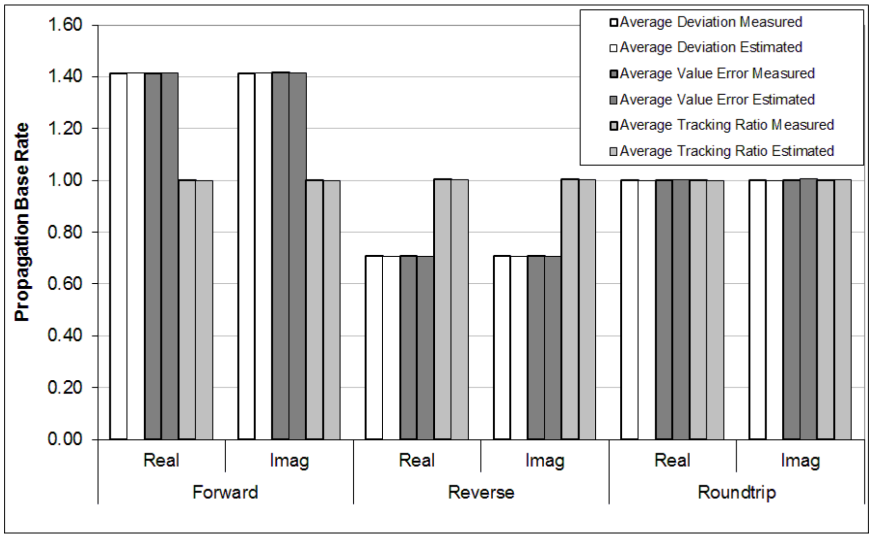

Figure 17: Empirical and theoretical β for fitting average output deviations, value errors and tracking ratios for forward, reverse and roundtrip FFT using independence arithmetic on noisy sine signals. In the chart, “Real” means real part, and “Imag” means imaginary part.

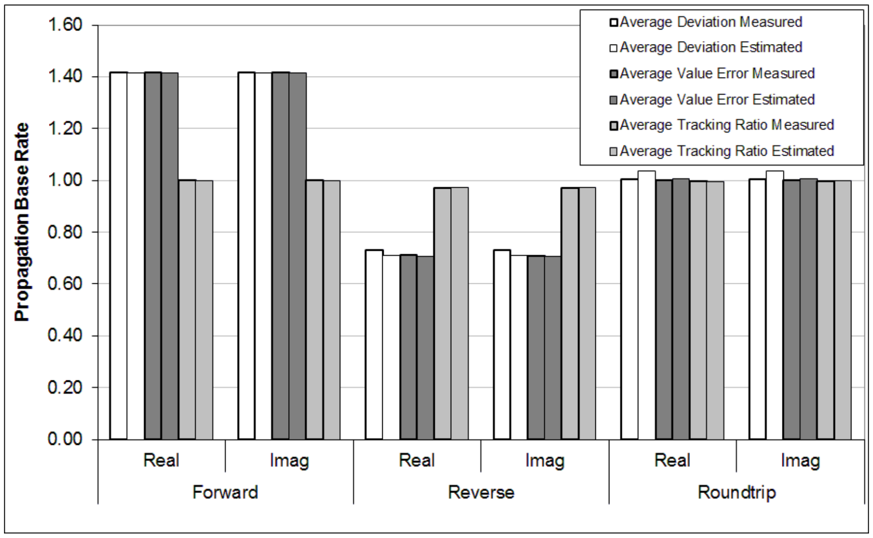

Figure 18: Empirical and theoretical β for fitting average output deviations, value errors and tracking ratios for forward, reverse and roundtrip FFT using precision arithmetic on noisy sine signals. In the chart, “Real” means real part, and “Imag” means imaginary part.

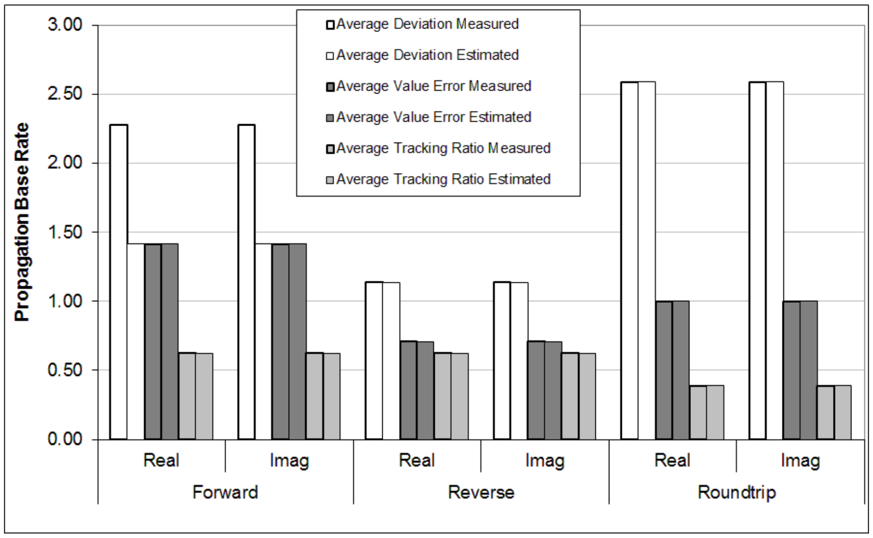

Figure 19: Empirical and theoretical β for fitting average output deviations, value errors and tracking ratios for forward, reverse and roundtrip FFT using interval arithmetic on noisy sine signals. In the chart, “Real” means real part, and “Imag” means imaginary part.

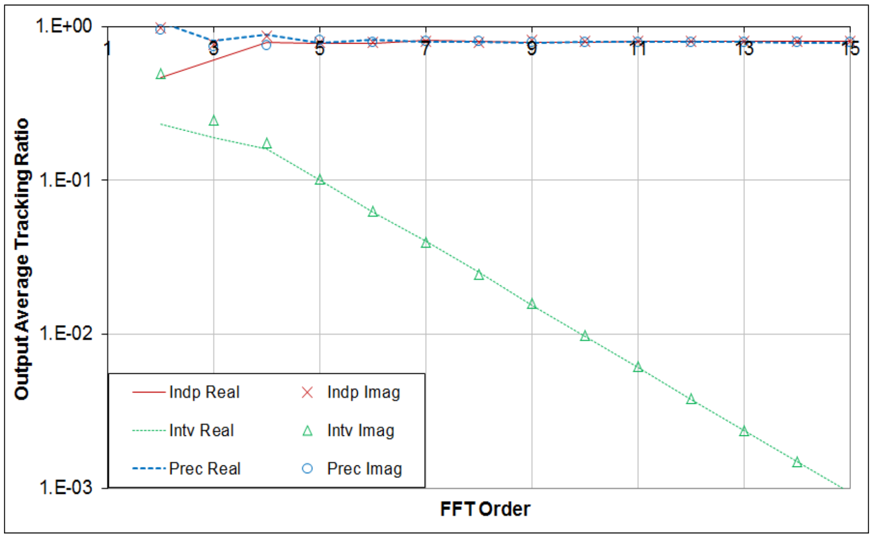

Figure 20: The empirical output average tracking ratios vs. the FFT order of the forward FFT for all three arithmetics when the input uncertainty deviation is 10^{-3}. In the legend, ”Intv” means interval arithmetic, ”Indp” means independence arithmetic, ”Prec” means precision arithmetic, “Real” means real part, and “Imag” means imaginary part.

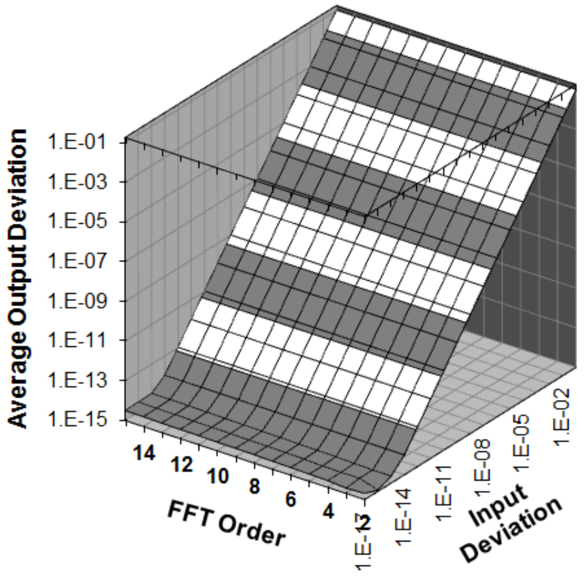

Figure 21: The empirical average output deviations vs. the FFT order and input deviations using precision arithmetic for the round-trip FFT algorithm.

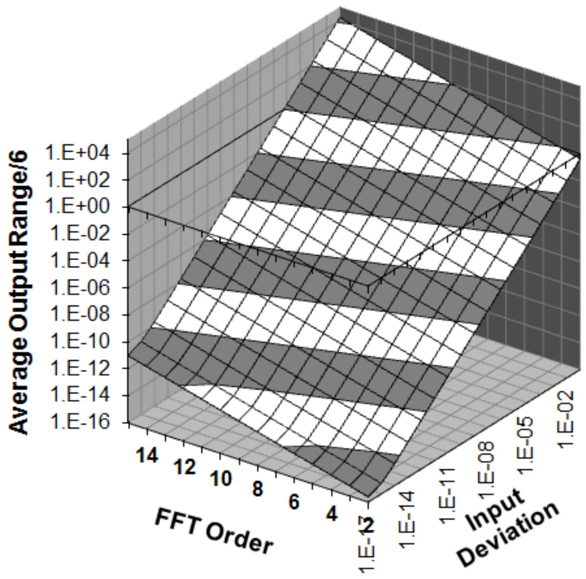

Figure 22: The empirical average output deviations vs. the FFT order and input deviations using interval arithmetic for the round-trip FFT algorithm.

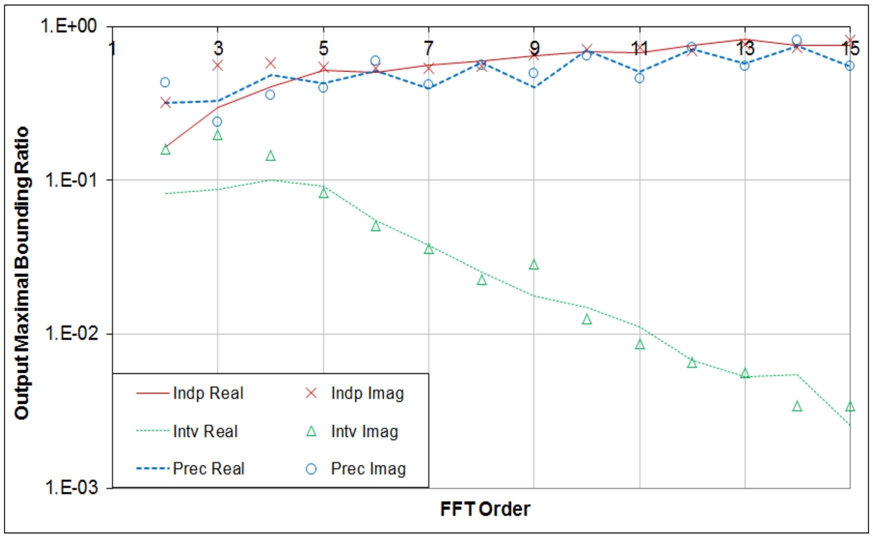

Figure 23: The empirical maximal output bounding ratios vs. the FFT order of the forward FFT for all three arithmetics. In the legend, "Intv" means interval arithmetic, "Indp" means independence arithmetic, "Prec" means precision arithmetic, "Real" means real part, and "Imag" means imaginary part.

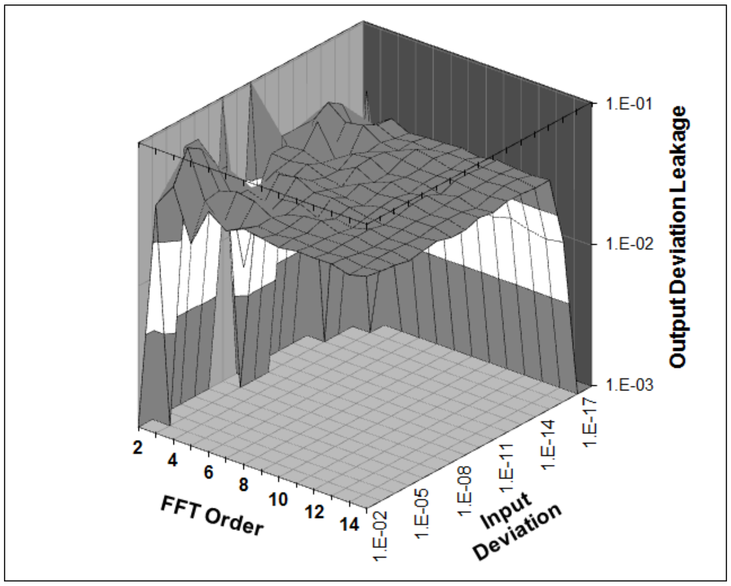

Figure 24: The empirical deviation leakages vs. the FFT order and input deviations using precision arithmetic for the forward FFT algorithm.

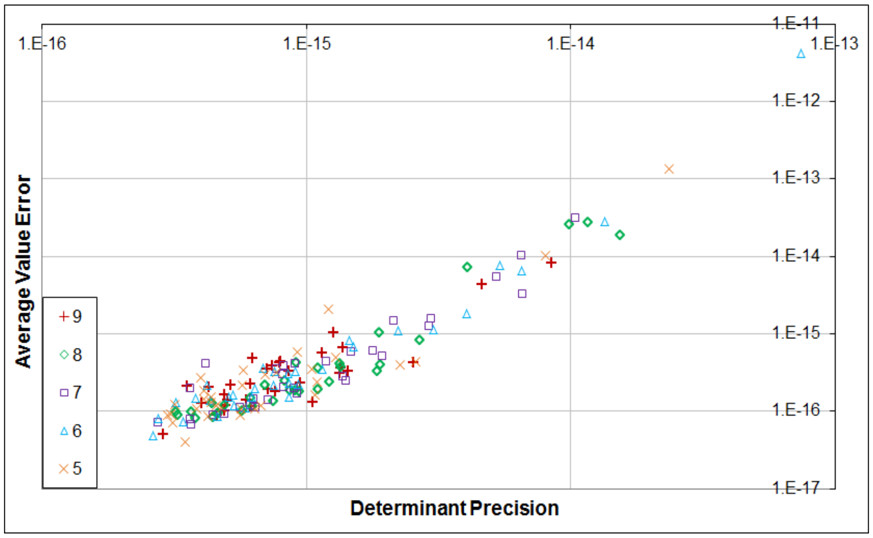

Figure 25: The empirical average value errors of the inverted matrix using conventional floating-point arithmetic vs. matrix determinant precision using precision arithmetic for clean matrices of different sizes (as shown in legend).

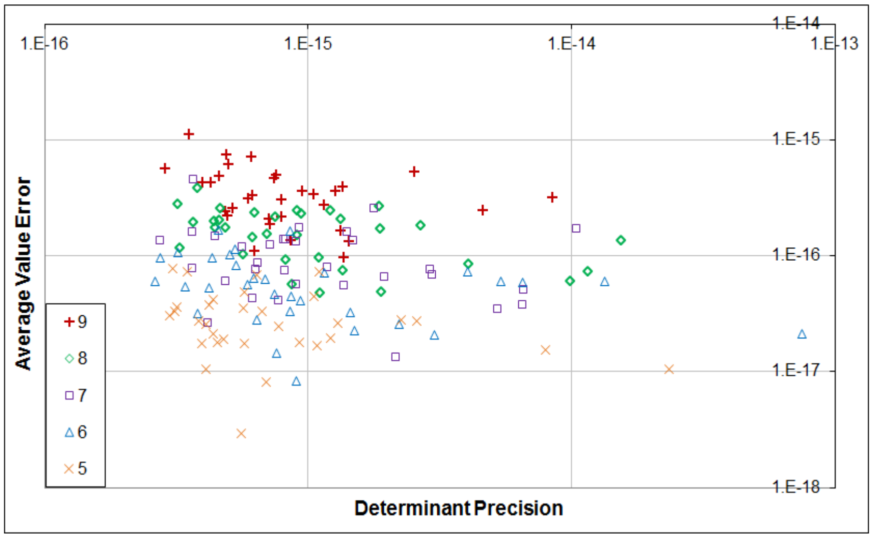

Figure 26: The empirical average value errors of the adjugate matrix using conventional floating-point arithmetic vs. matrix determinant precision using precision arithmetic for clean matrices of different sizes (as shown in legend).

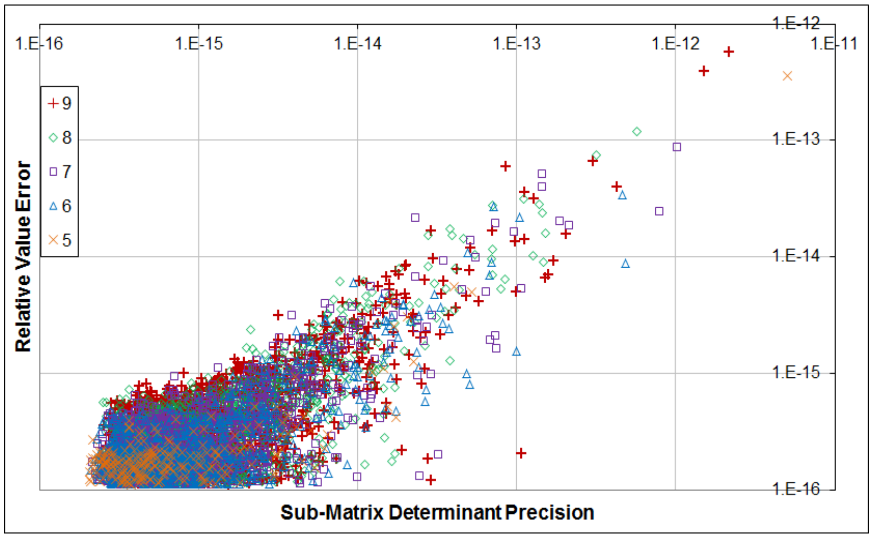

Figure 27: Empirical relative value errors of the adjugate matrix using conventional floating-point arithmetic vs. corresponding sub-matrix determinant precision using precision arithmetic for clean matrices of different sizes (as shown in legend).

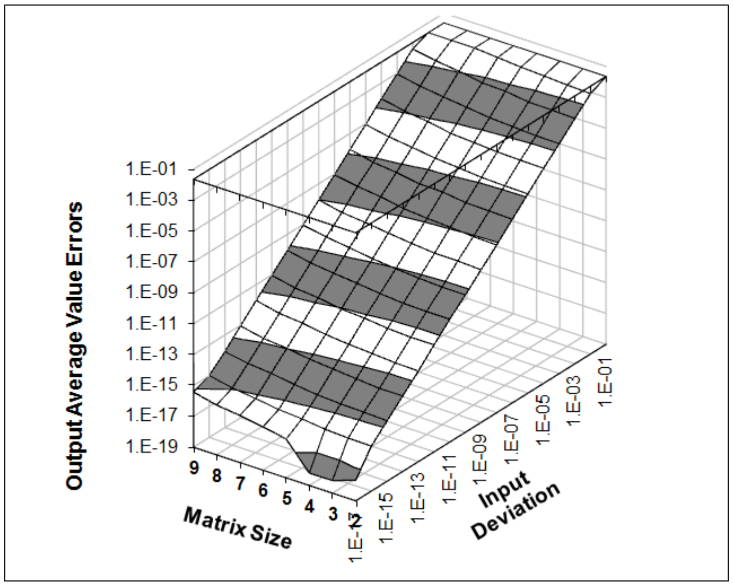

Figure 28: Using precision arithmetic, the average output deviations of the adjugate matrix vs. input precision and the matrix size.

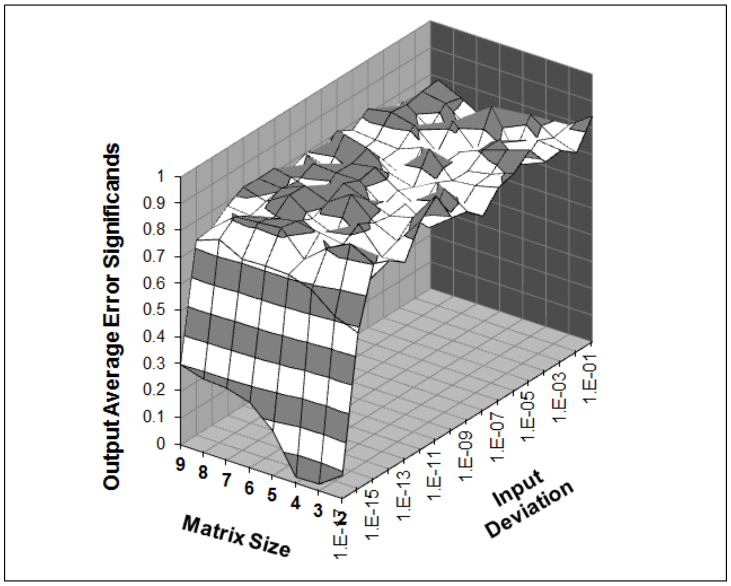

Figure 29: Using precision arithmetic, the average output tracking ratios of the adjugate matrix vs. input precision and the matrix size.

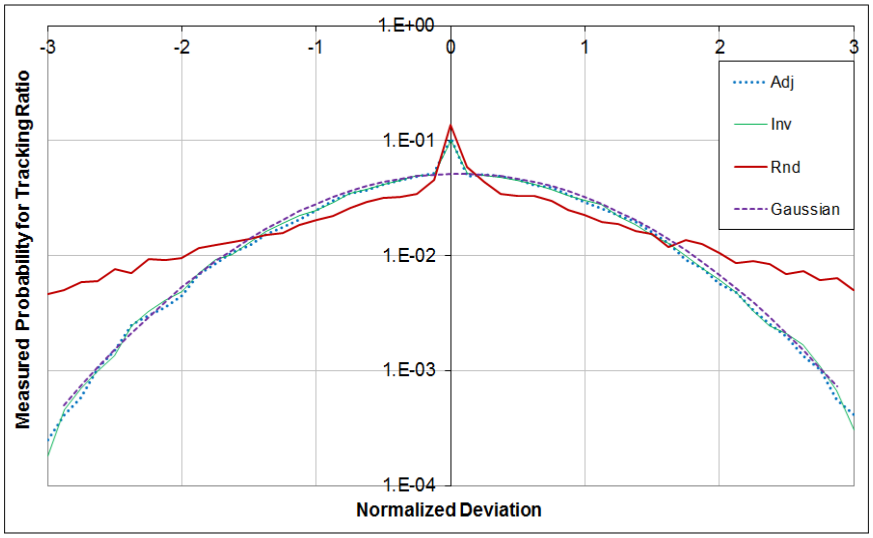

Figure 30: The measured tracking ratio distributions using precision arithmetic for matrix calculations of matrix size 9. They are best fitted by a Gaussian distribution with the mean of 0.06 and deviation of 0.96. In the legend, "Adj" means calculating adjugate M^A, "Inv" means calculating inverted M^{-1}, and "Rnd" means calculating double inverted $(M^{-1})^{-1}$.

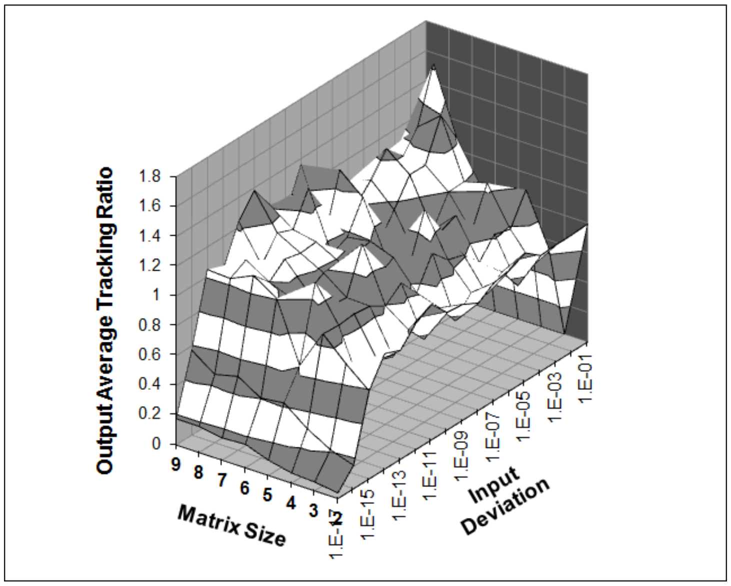

Figure 31: Using precision arithmetic, the average output tracking ratios of the double inverted matrix vs. input precision and the matrix size.

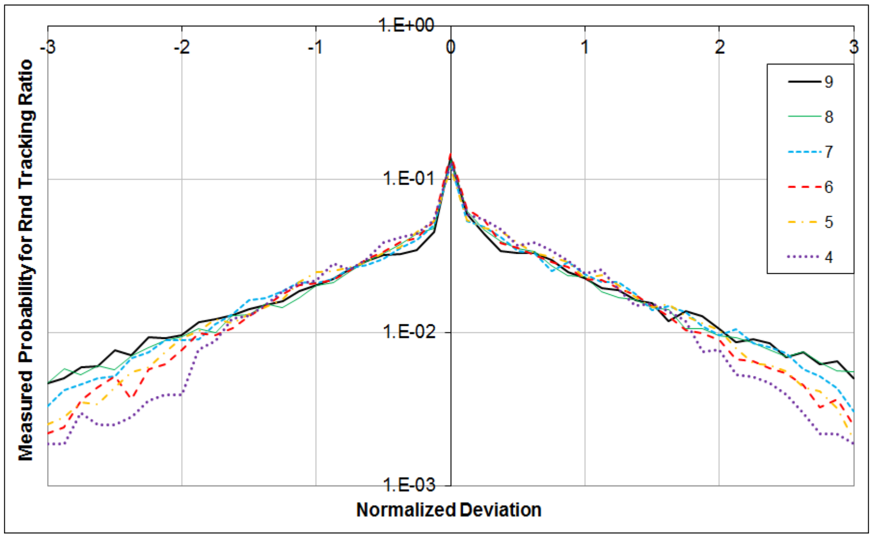

Figure 32: The measured tracking ratio distributions using precision arithmetic for matrix calculations of matrix size 9. They are best fitted by a Gaussian distribution with the mean of 0.06 and deviation of 0.96. In the legend, "Adj" means calculating adjugate M^A, "Inv" means calculating inverted M^{-1}, and "Rnd" means calculating double inverted $(M^{-1})^{-1}$.

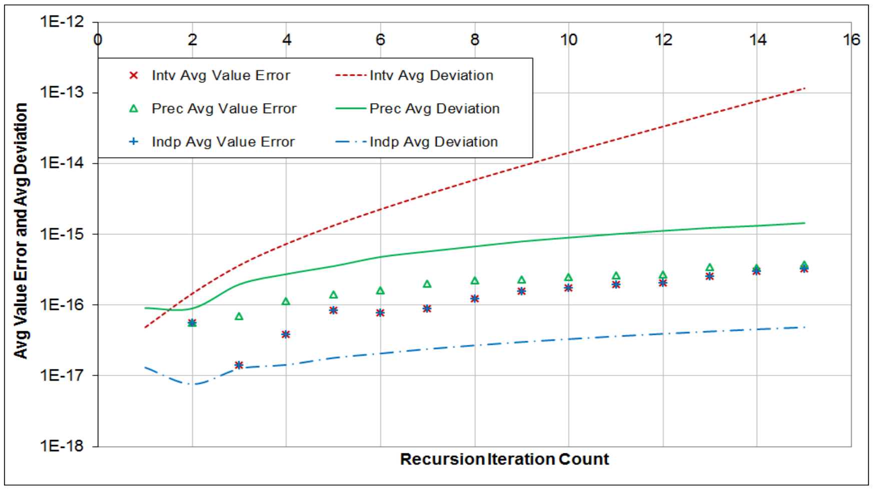

Figure 33: The empirical output average value errors and corresponding average output deviations vs. the recursion iteration count of the regressive calculation of sine values using interval arithmetic, precision arithmetic and independent arithmetic. The x-axis indicates the recursion iteration count L, while the y-axis indicates either the average value errors or average uncertainty deviations. In the legend, "Intv" means interval arithmetic, "Indp" means independence arithmetic, and "Prec" means precision arithmetic.

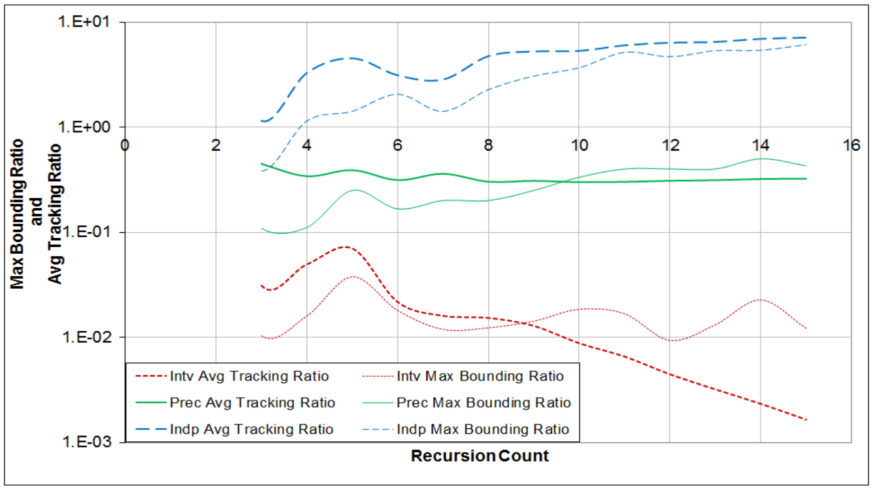

Figure 34: The empirical output maximal bounding ratios and average tracking ratios vs. the recursion iteration count of the regressive calculation of sine values using interval and precision arithmetic. In the legend, "Intv" means interval arithmetic, "Indp" means independence arithmetic, and "Prec" means precision arithmetic.

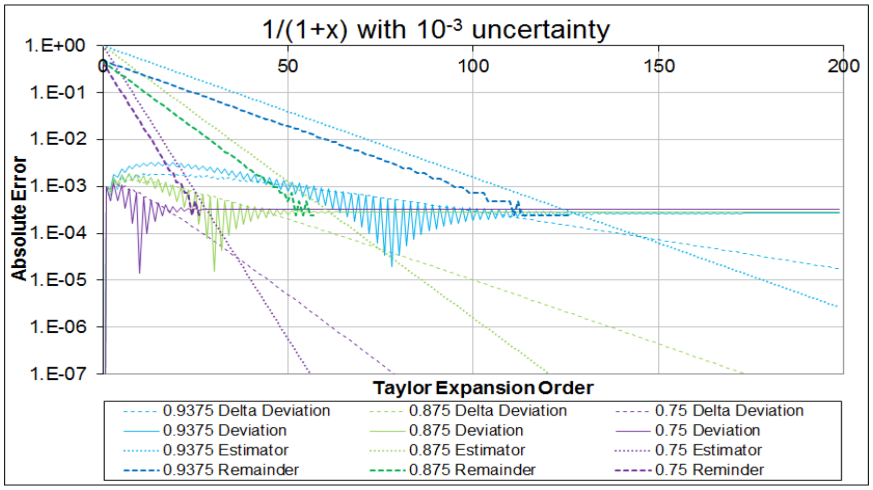

Figure 35: The delta deviation, deviations, Cauchy estimator and remainders of a Taylor expansion vs. the expansion orders for different input value with 10^{-3} input uncertainty using precision arithmetic with 0-bit calculated inside uncertainty. Different inputs are displayed using different color.

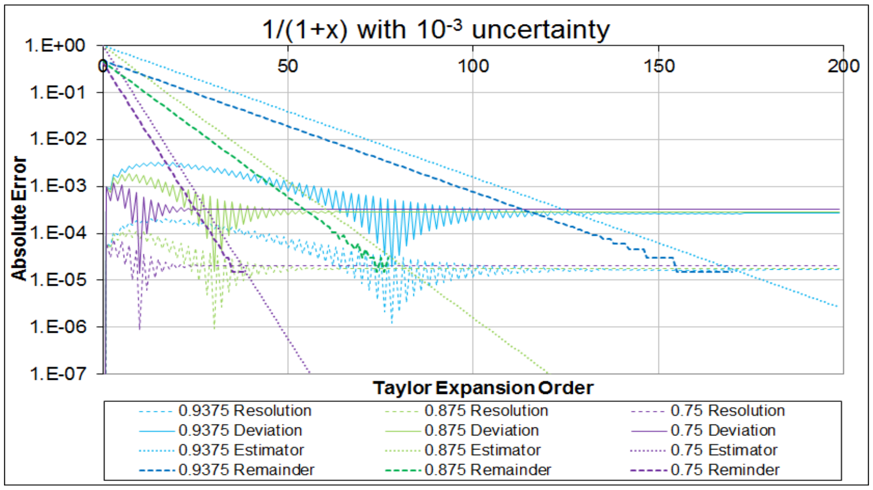

Figure 36: The deviations, resolutions, Cauchy estimator and remainders of a Taylor expansion vs. the expansion orders for different input value with 10^{-3} input uncertainty using precision arithmetic with 4-bit calculated inside uncertainty. Different inputs are displayed using different color.

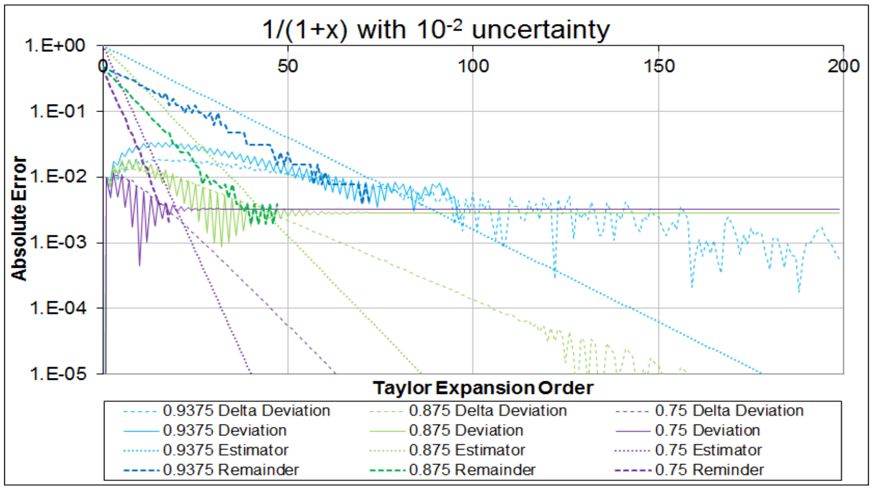

Figure 37: The delta deviation, deviations, Cauchy estimator and remainders of a Taylor expansion vs. the expansion orders for different input value with 10^{-2} input uncertainty using precision arithmetic with 0-bit calculated inside uncertainty. Different inputs are displayed using different color.

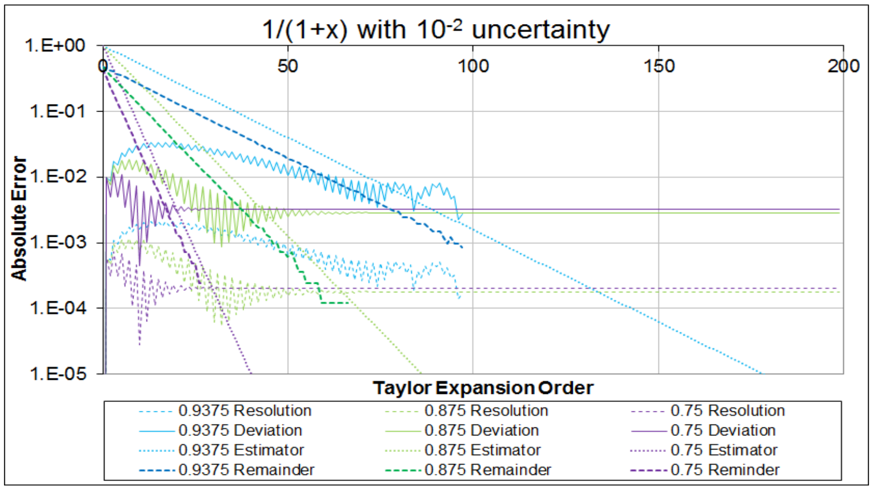

Figure 38: The deviations, resolutions, Cauchy estimator and remainders of a Taylor expansion vs. the expansion orders for different input value with 10^{-2} input uncertainty using precision arithmetic with 4-bit calculated inside uncertainty. Different inputs are displayed using different color.

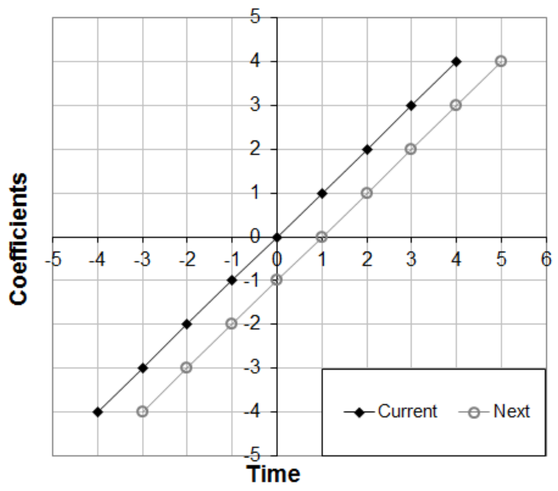

Figure 39: Coefficients of X in (9.2) at current and next position in a time series of the least square linear regression. Except the two end points at $X = -H$ and $X = H + 1$, respectively, the coefficient difference between the current and then next position in a time series are all by 1 in the overlapping region from $X = -H + 1$ to $X = H$, which results in (9.3).

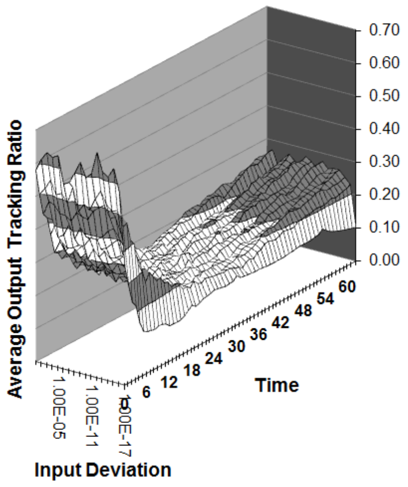

Figure 40: The average tracking ratio vs. time and the input uncertainty deviations for the progressive moving-window linear regression of a straight line using precision arithmetic with 4-bit calculated inside uncertainty.

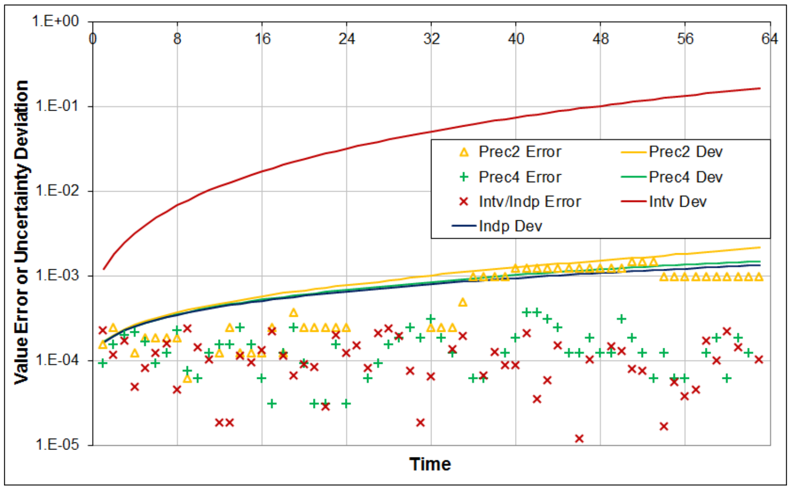

Figure 41: The output uncertainty deviations and the value errors vs. time for the progressive moving-window linear regression of a straight line. In the legend, "Indp" means independent arithmetic, "Intv" means interval arithmetic, "Prec4" and "Prec2" means the precision arithmetic with 4-bit and 2-bit calculated inside uncertainty, respectively.

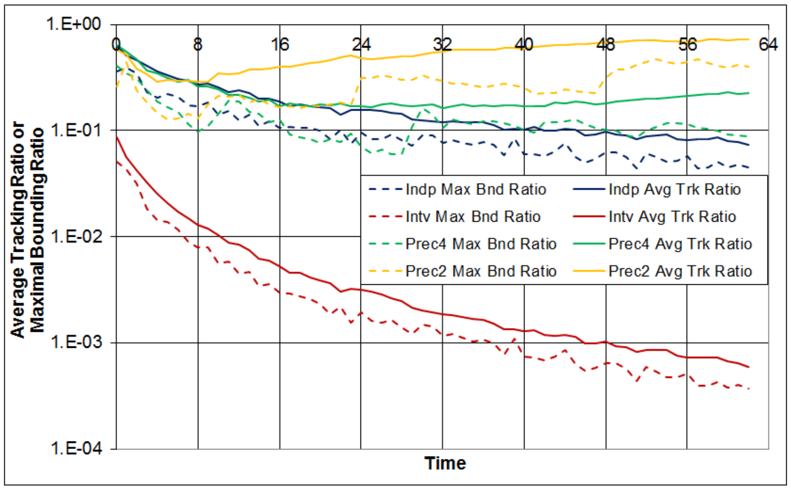

Figure 42: The average tracking ratios and the max bounding ratios vs. time for the progressive moving-window linear regression of a straight line. In the legend, "Indp" means independence arithmetic, "Intv" means interval arithmetic, "Prec4" and "Prec2" means the precision arithmetic with 4-bit and 2-bit calculated inside uncertainty, respectively. "Max Bnd Ratio" is the abbreviation for the maximal bounding ratio, and "Avg Trk Ratio" is the abbreviation for the average tracking ratios.

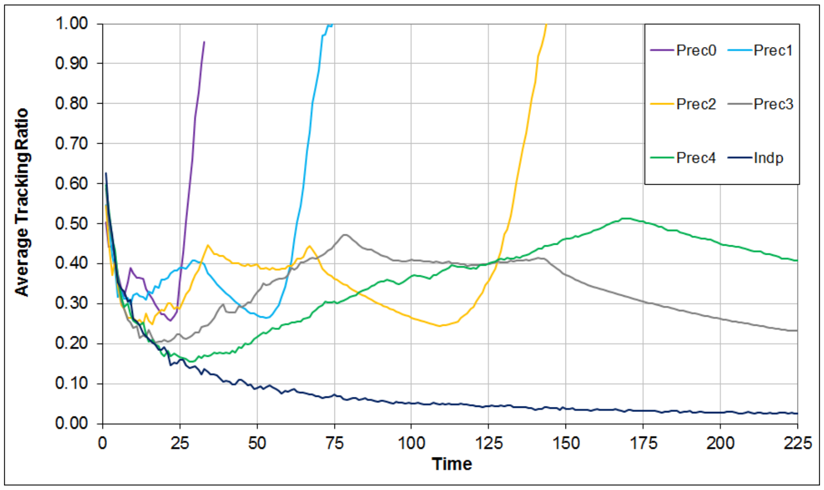

Figure 43: The average tracking ratios vs. time and the bits calculated inside uncertainty using precision arithmetic for the progressive moving-window linear regression of a straight line. In the legend, "Indp" means independence arithmetic, "PrecX" means the precision arithmetic with X-bit calculated inside uncertainty.

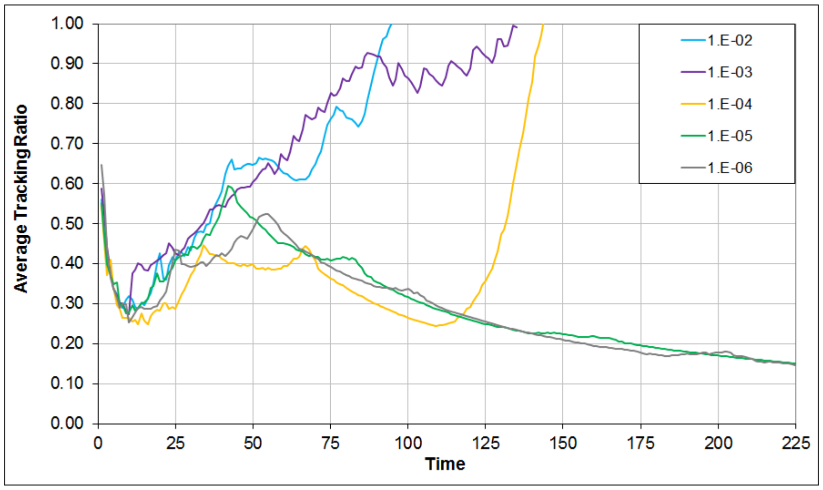

Figure 44: The average tracking ratios vs. time and the input precision using precision arithmetic with 2-bit calculated inside uncertainty for the progressive moving-window linear regression of a straight line for different input uncertainty deviations.

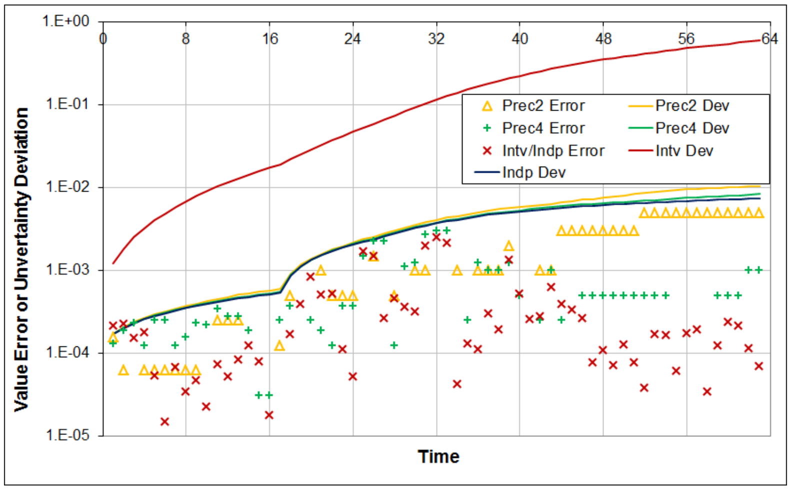

Figure 45: The output deviations and the value errors vs. time for the progressive moving-window linear regression of a straight line with 10-fold increase of both input noise and input uncertainty in the middle.

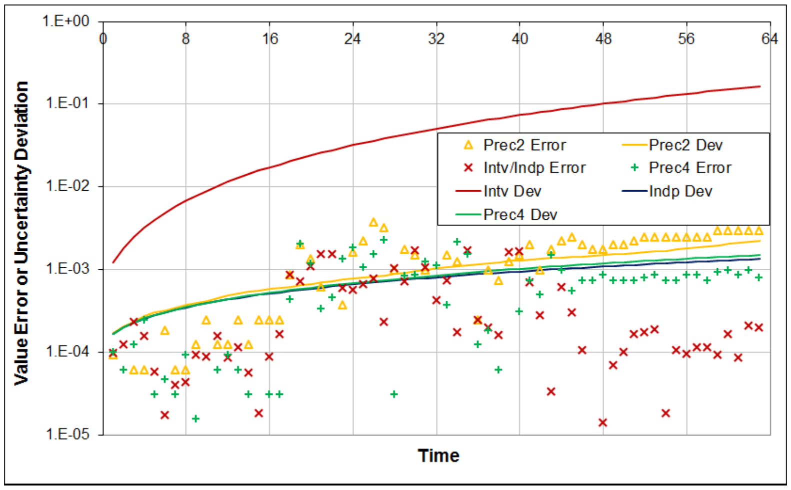

Figure 46: The output deviations and the value errors vs. time for the progressive moving-window linear regression of a straight line with only 10-fold increase of both input noise in the middle.

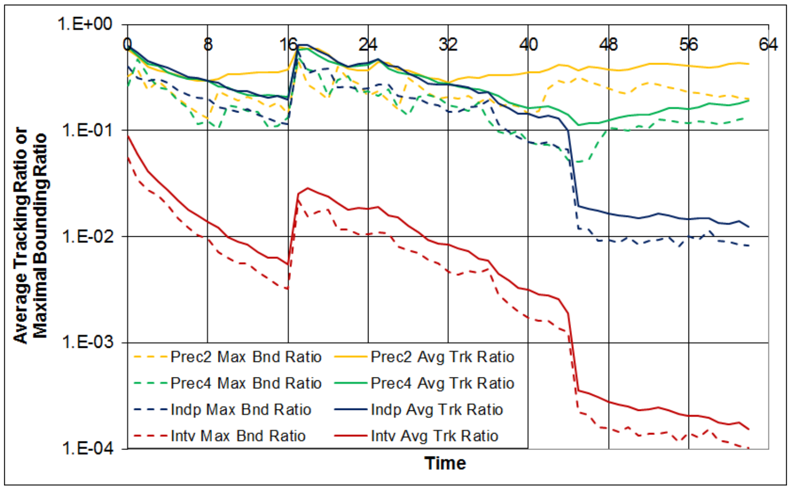

Figure 47: The average tracking ratios and the max bounding ratio vs. time for the progressive moving-window linear regression of a straight line with 10-fold larger input noise and deviation in the middle, to simulate larger noise following the straight line.

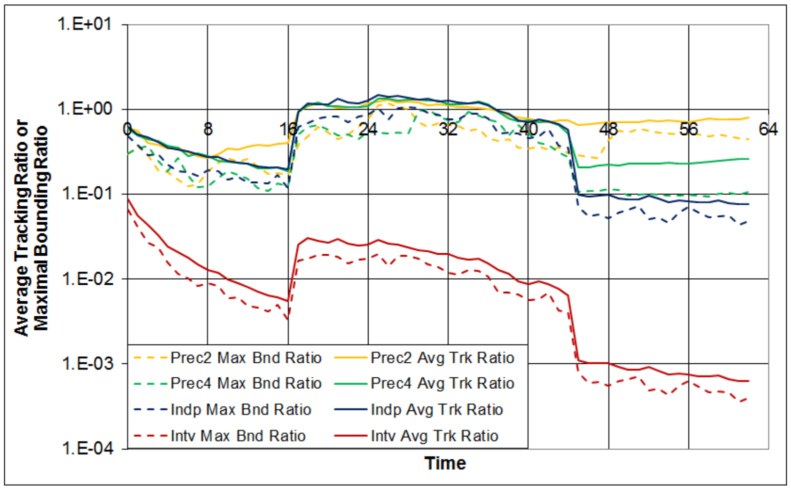

Figure 48: The average tracking ratios and the max bounding ratio vs. time for the progressive moving-window linear regression of a straight line with 10-fold larger input noise but same input deviation in middle, to simulate defects in obtaining the corresponding uncertainty deviations.

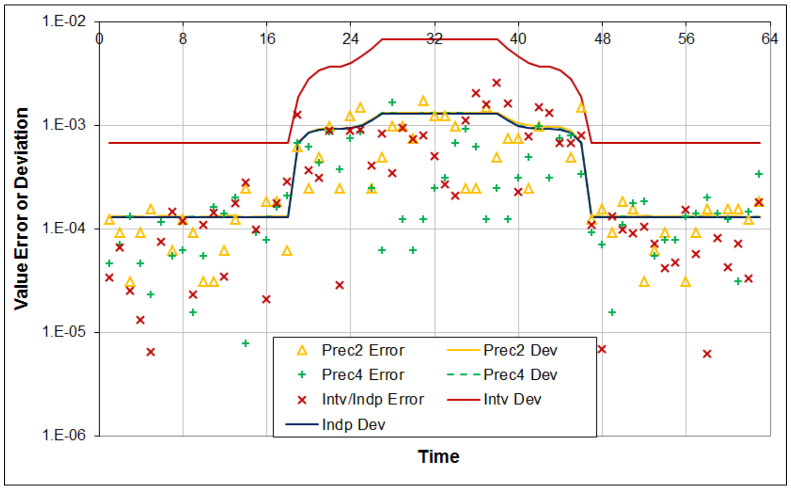

Figure 49: The output deviations and the value errors vs. time for the expressive moving-window linear regression of a straight line with 10-fold increase of both input noise and input uncertainty in the middle using precision arithmetic with 4-bit calculated inside uncertainty.

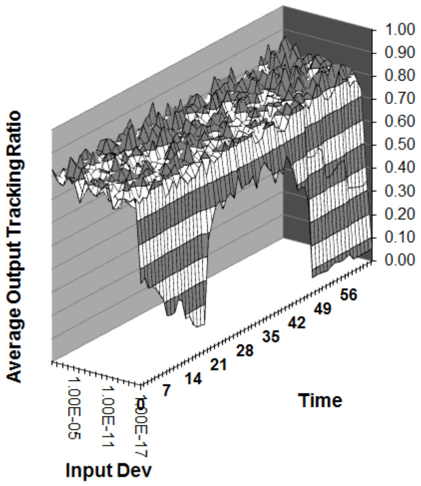

Figure 50: The average tracking ratio vs. time and the input uncertainty deviations for the expressive moving-window linear regression of a straight line with 10-fold increase of both input noise and input uncertainty in the middle using precision arithmetic with 4-bit calculated inside uncertainty.

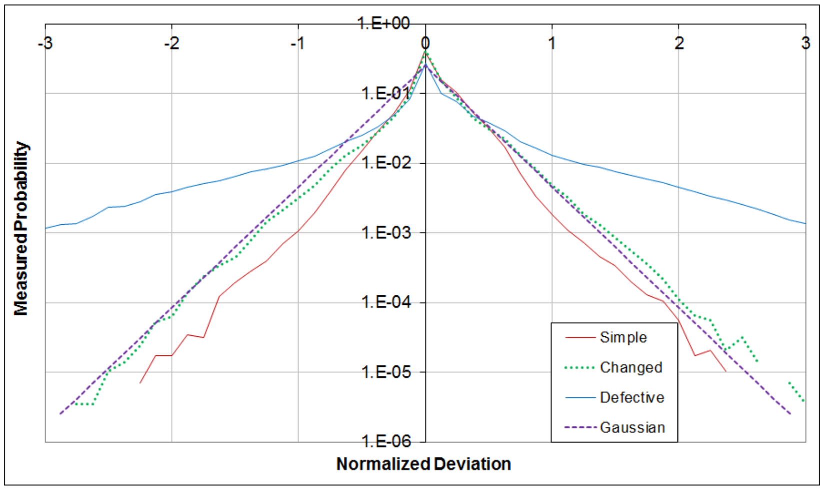

Figure 51: The measured tracking ratio distributions of the progressive moving-window linear regression for different cases (as shown in legend) using precision arithmetic with 4-bit calculated inside uncertainty. The case of "Changed" is best fitted by a exponential distribution with the mean of 0 and deviation of 0.25.

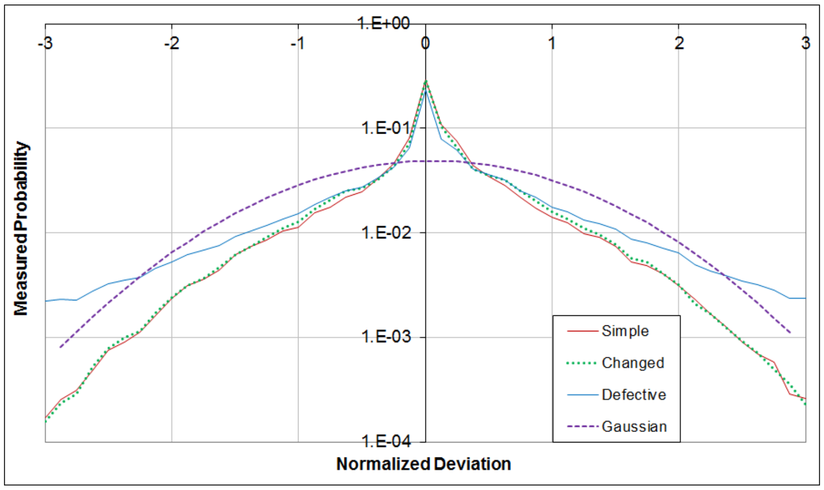

Figure 52: The measured tracking ratio distributions of the expressive moving-window linear regression using Formula (9.3) for different cases (as shown in legend) using precision arithmetic with 4-bit calculated inside uncertainty. The cases of "Simple" and "Changed" are best fitted by a Gaussian distribution with the mean of 0.06 and deviation of 0.97.