Variance Arithmetic*

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

Abstract

A new deterministic uncertainty-bearing floating-point arithmetic called *variance arithmetic* is developed to track the uncertainty for arithmetic calculations statistically. It uses a novel rounding scheme to avoid the excessive rounding error propagation in 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 based on a truncated Gaussian distribution. The variance arithmetic is found to be superior to interval arithmetic in both uncertainty-tracking and uncertainty-bounding for normal usages.

Let δx be the uncertainty distribution deviation for a value x. Define define $\delta x/|x|$ as precision, which represents the information content of the value with uncertainty. The variance arithmetic has precision requirements on the inputs. When the precision requirements are satisfied:

- The output values are identical to the result if there were no uncertainties in the inputs.
- The precision is preserved for unitary operations such as inversion, square and square-root.
- The variance arithmetic provides ideal approach when the analytic solution is available, to avoid the dependency problem completely.
- For a round-trip calculation without the dependency problem such as after forward and backward FFT, the original input uncertainties can be recovered.

When the precision requirements are not satisfied, the bias of the outputs can be expressed in terms of input precisions.

Keywords: computer arithmetic, error analysis, interval arithmetic, uncertainty, 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 Uncertainty

Except for the simplest counting, scientific and engineering measurements never give completely precise results [1][2]. In scientific and engineering measurements, the uncertainty of a measurement x usually is characterized by either the sample deviation δx or the uncertainty range Δx [1][2].

- If $\delta x = 0$ or $\Delta x = 0$, x is a *precise value*.
- Otherwise, x is an *imprecise value*.

$P \equiv \delta x/|x|$ is defined as the *statistical precision* (or simply precision in this paper) of the measurement, in which x is the value, and δx is the uncertainty deviation. A larger precision means a coarser measurement while a smaller precision mean finer measurement. 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].

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:

$$64919121 \times 205117922 - 159018721 \times 83739041 = \\ 13316075197586562 - 13316075197586561 = 1; \quad (1.1)$$

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

$$64919121 \times 205117922 - 159018721 \times 83739041 = \\ 64919121.000000000 \times 205117922.000000000 - 159018721.000000000 \times 83739041.000000000 = \\ 13316075197586562. - 13316075197586560. = 2. = 2.0000000000000000; \quad (1.2)$$

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 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[1] [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 arithmetic formulas[2] [20]:

$$[x] + [y] = [\underline{x} + \underline{y}, \bar{x} + \bar{y}]\,; \tag{1.3}$$

$$[x] - [y] = [\underline{x} - \bar{y}, \bar{x} - \underline{y}]\,; \tag{1.4}$$

$$[x] \times [y] = [\min(\underline{xy}, \underline{x}\bar{y}, \bar{x}\underline{y}, \bar{x}\bar{y}), \max(\underline{xy}, \underline{x}\bar{y}, \bar{x}\underline{y}, \bar{x}\bar{y})]\,; \tag{1.5}$$

$$0 \notin [y] :\ [x] \,/\, [y] = [x] \times [1/\bar{y}, 1/\underline{y}]\,; \tag{1.6}$$

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

[1] 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).

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

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 [3].

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 problem of interval arithmetic [21]. There is no known generic way to apply Taylor expansion in interval arithmetic, so that the dependence problem is an intrinsic problem for interval arithmetic.

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

$$f(x) = x^2 - x; \tag{1.8}$$

$$f(x) = (x - 1)x; \tag{1.9}$$

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, which is achieved in the variance arithmetic, as show later in this paper.

In this paper, an interval is presented as $x \pm \Delta x$, in which x is the middle of the range, and Δx is the half range.

[3] 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.

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 imprecise values:

$$(x \pm \delta x) + (y \pm \delta y) = (x + y) \qquad \pm\sqrt{\delta x^2 + \delta y^2 + 2\delta x \delta y \gamma}; \tag{1.10}$$

$$(x \pm \delta x) - (y \pm \delta y) = (x - y) \qquad \pm\sqrt{\delta x^2 + \delta y^2 - 2\delta x \delta y \gamma}; \tag{1.11}$$

$$(x \pm \delta x) \times (y \pm \delta y) = (x \times y) \qquad \pm|x \times y|\sqrt{{P_x}^2 + P(y)^2 + 2P_x P(y)\gamma}; \tag{1.12}$$

$$(x \pm \delta x)/(y \pm \delta y) = (x/y) \qquad \pm|x/y|\sqrt{{P_x}^2 + P(y)^2 - 2P_x P(y)\gamma}; \tag{1.13}$$

Tracking uncertainty propagation statistically seems a better 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. As show later in this paper, the variance arithmetic is based on a statistical assumption much more lenient than knowing the correlation between imprecise values.

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[4] [30][29], or calculating the mean and variance of a Taylor expansion [31]. Perhaps, it is reasonable to assume the uncertainties of experimental measurements to be independent of each other, while is it not reasonable to assume the uncertainties of calculations to be independent, which is essentially the approach of variance arithmetic.

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 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,

[4] 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].

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.

As show later in this paper, the variance arithmetic achieved significance arithmetic, allowing the significance bit count to be real numbers.

1.6 An Overview of This Paper

In this paper, a new floating-point arithmetic called *variance arithmetic* 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. Variance arithmetic is compared with other uncertainty-bearing arithmetics in Section 5 to Section 9. A brief discussion is provided in Section 11.

Variance arithmetic is an improvement of the previously published precision arithmetic by the same author [37]. This paper is comparable to its processor [37]:

- Section 1 remains almost unchanged.
- The basic assumptions in Section 2 are identical. The variance arithmetic improves over its processor in the following areas:
  - Unlike its processor, the variance arithmetic no longer has the dependency problem. Instead, the variance arithmetic can trace dependency between uncertainties through the identities of variables.
  - The digital representation gives a defined behavior of bits calculated inside uncertainty.
  - The result variance using Taylor expansion was statistically inaccurate previously. The new approach no longer calculate variance around the value if there were no uncertainty.
- Section 3 remains largely unchanged, except it contains the last subsection from its processor.
- Section 4 is new, to provide a statistical test of variance arithmetic in using the common math library functions.
- Section 5 is redone using the new implementation of variance arithmetic. As a linear calculation with each input datum used only once, FFT provides an ideal test case.

2 The Variance Arithmetic

2.1 Foundation for Variance Arithmetic

The variance arithmetic calculates the uncertainty variance around each output value [37]. It is based on:

- the round up rule,
- the precision scaling principle, and
- the uncorrelated uncertainty assumption.

The variance arithmetic uses *the round up rule* [37] to avoid the rounding error accumulation in the conventional floating-point representation.

The statistical precision $P(x) = \delta x/|x|$ represents the reliable information content of a measurement statistically, with finer or smaller statistical precision means higher reliability and better reproducibility of the measurement [1][2]. $P(x)$ has an upper bound of 1. When $1 \leq P(x)$:

- Either δx is too coarse to determine the actual value of x,
- Or $x \pm \delta x$ is a measurement of 0.

Variance arithmetic is based on the *precision scaling principle* [37]: When a imprecise is multiplied by a precise value, the result precision equals the precision of the imprecise value.

The inputs to variance arithmetic should obeys the *uncorrelated uncertainty assumption* [37], which means that the uncertainties of any two different inputs can be regarded as uncorrelated of each other. This assumption is consistent with the common methods in processing experimental data [1][2], such as the common knowledge or belief that precision improves with the count n as $1/\sqrt{n}$ during averaging. It is much weaker than requiring any two different inputs to be independent of each other. It can be turned into a realistic and simple statistical requirement or test between two inputs.

The uncorrelated uncertainty assumption only applies to imprecise inputs. When the inputs obeys the *uncorrelated uncertainty assumption*, in the intermediate steps, variance arithmetic uses statistics to account for correlation between uncertainties through their associated variables, such as $x \pm \delta x$ and $y \pm \delta y$. Variance arithmetic has no dependency problem for analytic questions, which is different from the claim of its predecessor [37].

2.2 The Uncorrelated Uncertainty Assumption [37]

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 precision of two measurements.

The correlated parts common to different measurements are regarded as signals, which can either be desired or unwanted. Let X, Y, and Z denote three mutually independent random variables [4] with variance $V(X)$, $V(Y)$ and $V(Z)$, respectively.

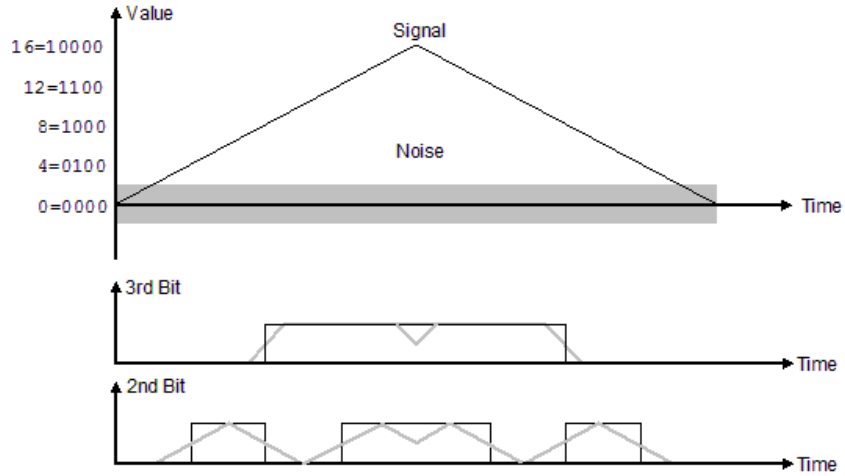

Figure 1: Effect of noise on bit values of a measured value. The triangular wave signal and the added white noise are shown at top using the thin black line and the grey area, respectively. The values are measured by a theoretical 4-bit Digital-to-Analog Converter in ideal condition, assuming LSB is 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.

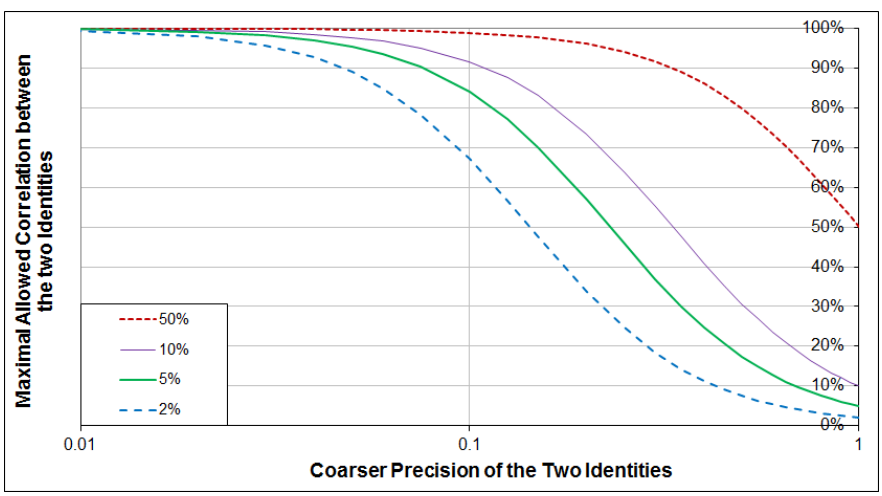

Figure 2: Allowed maximal correlation between two values vs. input precisions and independence standard (as shown in legend) for the independence uncertainty assumption of variance arithmetic to be true.

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:

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

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

Formula (2.4) gives the correlation γ between two random variables, each of which contains a completely uncorrelated part and a completely correlated part X, with η being the average ratio between these two parts. Formula (2.4) can also be interpreted reversely: if two random variables are correlated by γ, each of them can be viewed hypothetically 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 (2.4) 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 (2.4):

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. Quantitatively, let P denote the precision of an imprecise value, 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 (2.4), 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. Thus for each independence standard γ_P, there is a maximal allowed correlation between two values below which the uncorrelated uncertainty assumption of variance 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. 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 them satisfy the uncertainties uncertainty assumption. 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, which defines the upper bound for γ_P. [5]

2.3 The Round Up Rule [37]

The variance arithmetic uses *the round up rule* to avoid the rounding error accumulation in the conventional floating-point representation. Let the content of a floating-point number be denoted as $S\ 2^E$, in which S is the significand [6] and E is the exponent of 2 of the floating-point number. In addition, the variance representation contains a *carry sign* $\sim$ to indicate the sign of its rounding error. When E needs to be increased by 1, S is *round up* once, which proceeds according to the following round up rule, to limit the rounding error to $(-1/2, +1/2)$ of LSB (Least Significant Bit) of the significand:

- 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 $S+2^{E+1}$.
- If $\sim$ is unknown, it is initiated as $-$, so that 1 is rounded up to 0. Even if the initial $\sim$ is wrong, the rounding error will be either reduced or corrected by the round up process.

The carry sign $\sim$ bit in variance arithmetic has the same functionality as the sticky bit or guard bits in conventional floating-point calculations [9] to reduce rounding errors. Numerically, using the sticky bit and guard bits, the accuracy of the conventional floating-point calculation on a modern computer is pretty high, e.g., for $\sqrt{x^2}$ and $(\sqrt{x})^2$, the error is limited to the least significant 2 bits. It is the scenario demonstrated by Formula (1.2) that is the major problem, when the rounding errors are saved as part of the data. Thus, the carry sign $\sim$ bit is part of the variance arithmetic representation.

[5] 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 precision of their measurements..

[6] 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.

2.4 Probability Distribution of Rounding Errors [37]

An ideal floating-point calculation is carried out conceptually to infinitesimal precision before it is rounded up to representation precision [9]. Thus, rounding up should be a process independent of any calculation, and it should be evaluated separately. Empirically, the rounding error are uniformly distributed within $(-1/2, +1/2)$ of LSB of the significand, in which $1/2$ is the *bounding range* R of the rounding error. Let $P_R(x)$ be the distribution density function for the rounding error with a bounding range of R.

$$P_{\frac{1}{2}}(x) \equiv 1, \quad -1/2 \leq x \leq +1/2; \tag{2.5}$$

The variance for $P_{1/2}(x)$ is $1/12$.

2.5 Result Uncertainty For Addition and Subtraction [37]

The uncorrelated uncertainty assumption suggests that the result bounding distribution density function of addition or subtraction is the convolution of the input density functions [4].

$$P_{\frac{n}{2}}(x) \equiv \int_{-\infty}^{+\infty} P_{\frac{1}{2}}(y) P_{\frac{n-1}{2}}(x-y)dy = \int_{-1/2}^{+1/2} P_{\frac{n-1}{2}}(x-y)dy, \quad n = 2, 3, 4 \ldots; \tag{2.6}$$

Figure 3 shows the probability distribution $P_{\frac{n}{2}}(x)$ of Formula (2.6). $P_2(x)$ is already close to the corresponding Gaussian distribution with the same mean and deviation, which is plotted in dash line of the same color.

The Lyapunov form of the central limit theorem [4] states that if X_i is a random variable with mean μ_i and variance V_i 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 $\sum_i^n V_i$. Applying the central limit theorem to the addition and subtraction:

$$V = n/12 = R/6; \tag{2.7}$$

- $P_{n/2}(x)$ converges in distribution to a Gaussian distribution of mean 0 and variance V as in Formula (2.7).
- Figure 3 shows that such convergence to Gaussian distribution is very quick.
- The stable rounding error distribution is independent of any initial rounding error distribution.

Also due to the center-limit theorem, uncertainty in general is expected to be Gaussian distributed [1] [4]. The rounding error distribution is extended to describe uncertainty distribution in general.

For simplicity, define operator $\delta^2 f(x) \equiv (\delta f(x))^2$. Formula (2.8) give the result variance of addition and subtraction surrounding $x \pm y$. Formula (2.9) gives the probability density function for $x \pm \delta x$ in general, in which $N()$ is the density function of the normal distribution [4]. Formula (2.9) can be normalized as Formula (2.10).

$$\delta^2(x \pm y) = \delta^2 x + \delta^2 y; \tag{2.8}$$

$$\rho(\tilde{x}, x, \delta x) = \frac{1}{\delta x} N(\frac{\tilde{x} - x}{\delta x}); \tag{2.9}$$

$$\tilde{z} \equiv \frac{\tilde{x} - x}{\delta x} : \quad \rho(\tilde{x}, x, \delta x) d\tilde{x} = N(\tilde{z}) d\tilde{z}; \tag{2.10}$$

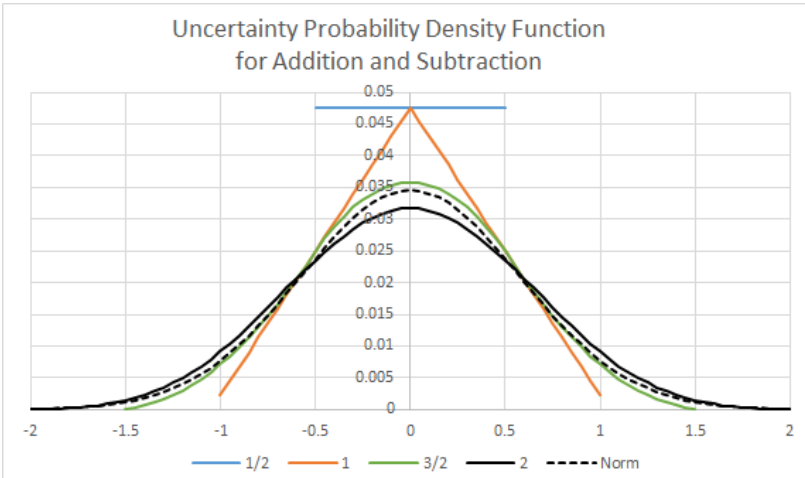

Figure 3: The probability distribution $P_{\frac{n}{2}}(x)$ of Formula (2.6), assuming $P_{\frac{1}{2}}(x)$ is uniformly distributed between $(-1/2, +1/2)$. The legend shows n. $P_2(x)$ is already close to the corresponding Gaussian distribution with the same mean and deviation, which is plotted in dash line of the same color.

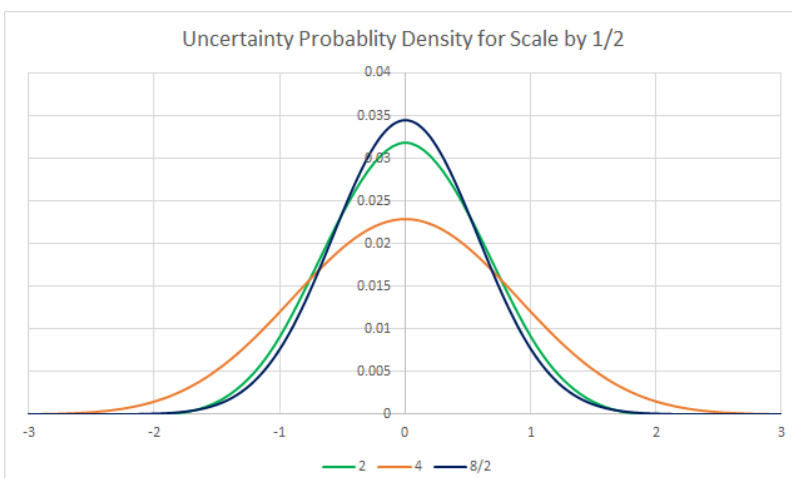

Figure 4: The result uncertainty distribution density function $P_{8/2}$ (which is P_8 scaled by 1/2 in the x-direction) is compared with P_2 and the P_4, with the distribution range shown in then legend. When $P_{8/2}$ is replaced by P_2, the measured bounding leakage is 0.06%.

Formula (2.8) is different from Formula (1.10) and (1.11), because in variance arithmetic, the correlation between x and y does not contribute to the result distribution as far as the uncorrelated uncertainty assumption is satisfied. It may show that Formula (1.10) and (1.11) are wrong statistically for the result uncertainty.

2.6 The Precision Scaling Principle [37]

Similar to how the conventional floating-point representation [10] saves each value as $S \times 2^E$ commonly in 32-bits or 64-bits, a value and its uncertainty variances are saved using *variance representation* which is another floating-point representation. To fit into their bit confinements, S and V need to be rounded up once for each increment of E. According to the precision scaling principle:

- During each round up, V is reduced to 1/4-fold.
- During each round down, V is increased to 4-fold.

Using the precision scaling principle, the variance representation represents $x \pm \delta x$ as:

$$x = S\ 2^E; \tag{2.11}$$

$$(\delta x)^2 = V\ 2^{2E}; \tag{2.12}$$

However, such round-up or round-down of V conflicts with the accuracy of R according to Formula (2.7). When P_8 is rounded up, Figure 4 shows that $P_{8/2}$ is much closer to P_2 than P_4 in distribution, with the former follows the logic of variance arithmetic, while the latter follows the logic of interval arithmetic. The precision scaling principle chooses to scarify R in favor of V. This choice introduces the probability for the actual value to be outside the range of $(x - R, x + R)$, which is defined as the *bounding leakage*, e.g., the bounding leakage is 6 10^{-4} When $P_{8/2}$ is replaced by P_2. To reduce the representation bounding leakage to virtually 0, V is maximized in its bit confinement in the *normalization* process of the variance representation.

2.7 Comparison

Two imprecise values can be compared statistically using their difference [1]. For example, the difference between 1.002 ± 0.001 and 1.000 ± 0.002 is 0.002 ± 0.00224, so that the probability for them to be not equal is $\xi(\frac{0.002}{0.00224}) = 81.4\%$, in which $\xi(z)$ is the cumulative density function for normal distribution [4]. If the threshold probability of not equality is 80%, $1.000 \pm 0.002 < 1.002 \pm 0.001$.

2.8 Multiplication

To obey the precision scaling principle, the result of multiplying $x \pm \delta x$ by a precise value y is $y^2\delta^2 x$. The result of multiplying $0 \pm \delta x$ by $0 \pm \delta y$ is a normal production distribution [4], which is centered at 0 with variance $(\delta x)^2(\delta y)^2$. The general multiplication can be decomposed as Formula (2.13), which leads to Formula (2.14) [37] and (2.15) [37] for the result variance and precision surrounding the result xy, respectively.

$$(x \pm \delta x) \times (y \pm \delta y) = (x + (0 \pm \delta x)) \times (y + (0 \pm \delta y)); \tag{2.13}$$

$$\delta^2(xy) = x^2(\delta^2 y) + y^2(\delta^2 x) + (\delta^2 x)(\delta^2 y); \tag{2.14}$$

$$xy \neq 0 : P(xy)^2 = P(x)^2 + P(y)^2 + P(x)^2 P(y)^2; \tag{2.15}$$

Formula (2.14) is identical to Formula (1.12) for statistical propagation of uncertainty except their cross term, representing difference in their statistical requirements, respectively. Formula (2.14) is simpler and more elegant than Formula (1.5) for the interval arithmetic multiplication. The result of Formula (1.2) calculated by variance arithmetic is $2 \pm 2\sqrt{5}$. It is 2 ± 9 for interval arithmetic [19].

2.9 Uncertainty Distribution

Let $\tilde{y} = f(\tilde{x})$ be a strictly monotonic function, so that $\tilde{x} = f^{-1}(\tilde{y})$ exist. In Formula (2.16) [1], the same distribution can be expressed in either $\tilde{x}$ or $\tilde{y}$ or $\tilde{z}$, which are different representations of the same underlying random variable. Using Formula (2.16), Formula (2.17), (2.18), and (2.19) give the $\rho(\tilde{y}, y, \delta y)$ for e^x, $\ln(x)$, and x^c, respectively.

$$N(\tilde{z})d\tilde{z} = \rho(\tilde{x}, x, \delta x)d\tilde{x} = \rho(f^{-1}(\tilde{y}), x, \delta x)\frac{d\tilde{x}}{d\tilde{y}}d\tilde{y} = \rho(\tilde{y}, y, \delta y)d\tilde{y}; \tag{2.16}$$

$$y = e^x : \rho(\tilde{y}, y, \delta y) = \frac{1}{\tilde{y}}\frac{1}{\delta x}N(\frac{\log(\tilde{y}) - x}{\delta x}); \tag{2.17}$$

$$y = \ln(x) : \rho(\tilde{y}, y, \delta y) = e^{\tilde{y}}\frac{1}{\delta x}N(\frac{e^{\tilde{y}} - x}{\delta x}); \tag{2.18}$$

$$y = x^c : \rho(\tilde{y}, y, \delta y) = c\tilde{y}^{\frac{1}{c}-1}\frac{1}{\delta x}N(\frac{\tilde{y}^{\frac{1}{c}} - x}{\delta x}); \tag{2.19}$$

Using Formula (2.16), Formula (2.20) gives the equation for the mode of the result distribution, whose solutions $f^{-1}(\tilde{y}_m)$ for e^x, $\ln(x)$, and x^c are Formula (2.21), (2.22), and (2.23) respectively.

$$\tilde{z} = \frac{f^{-1}(\tilde{y}) - x}{\delta x} : \quad \frac{d^2\tilde{z}}{d\tilde{y}^2} = \frac{d\tilde{z}}{d\tilde{y}}\tilde{z}; \tag{2.20}$$

$$\tilde{y} = e^{\tilde{x}} : \quad \ln(\tilde{y}_m) = x - (\delta x)^2; \tag{2.21}$$

$$\tilde{y} = \ln(\tilde{x}) : \quad e^{\tilde{y}_m} = \frac{x + \sqrt{x^2 + 4(\delta x)^2}}{2}; \tag{2.22}$$

$$\tilde{y} = \tilde{x}^c : \quad (\tilde{y}_m)^{1/c} = \frac{x + \sqrt{x^2 - 4(c-1)(\delta x)^2}}{2}; \tag{2.23}$$

The *Gaussian difference* of $\rho(f^{-1}(\tilde{y}), x, \delta x)$ is calculated as $\int_{-\infty}^{+\infty} |\rho(f^{-1}(\tilde{y}), x, \delta x) - \varrho(\tilde{y})|d\tilde{y}$ in which $\varrho(\tilde{y})$ is the Gaussian distribution of the same mode and the same deviation δx. A Gaussian difference of 0 means a perfect Gaussian.

Viewed in the $f^{-1}(\tilde{y})$ coordinate, $\rho(\tilde{y}, y, \delta y)$ is Gaussian modulated by $\frac{d\tilde{x}}{d\tilde{y}} = 1/f_x^{(1)}$. A *zero* of the uncertainty distribution happens when $f_x^{(1)} = \infty \rightarrow \rho(\tilde{y}, y, \delta y) = 0$, while a *pole* happens when $f_x^{(1)} = 0 \rightarrow \rho(\tilde{y}, y, \delta y) = \infty$. Zeros and poles have different impacts on the properties of of the uncertainty distributions.

If $y = f(x)$ is a linear transform of x, $f_x^{(1)}$ is a constant, and $\rho(\tilde{y}, y, \delta y)$ is known to be Gaussian [4]. From another prospective, a linear transformation generates neither zero nor pole according Formula (2.16).

Figure 5 shows the probability density function for $\sqrt{x \pm 1}$ according to Formula (2.19), which has a zero at $x = 0$. Each probability density function $\rho(\tilde{x})$ in solid line is compared with a normal distribution $\varrho(\tilde{x})$ of the same mode in dash line of the same color.

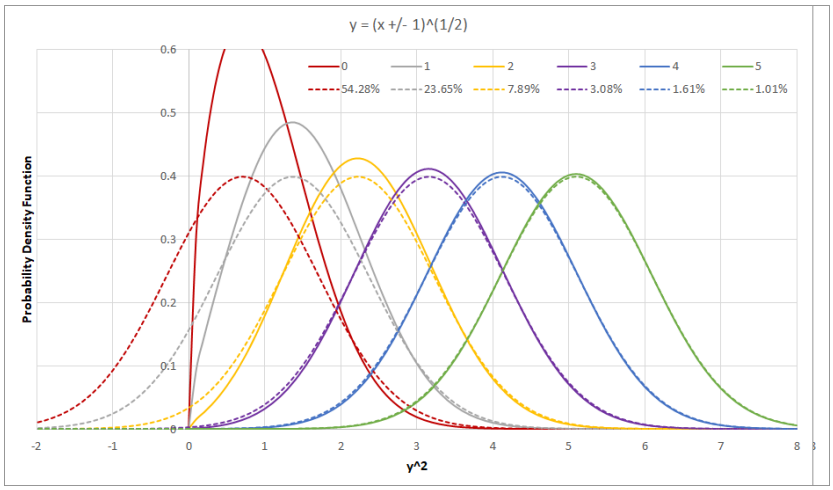

Figure 5: The probability density function for $\tilde{y} = \sqrt{x \pm 1}$, for different x as shown in the legend. The x-axis is scaled as $\tilde{y}^2$. Each probability density function $\rho(\tilde{x})$ in solid line is compared with a Gaussian distribution $\varrho(\tilde{x})$ of the same mode and the same deviation in the dash line of the same color. The legend for the Gaussian distribution shows the value of Gaussian difference in percentage.

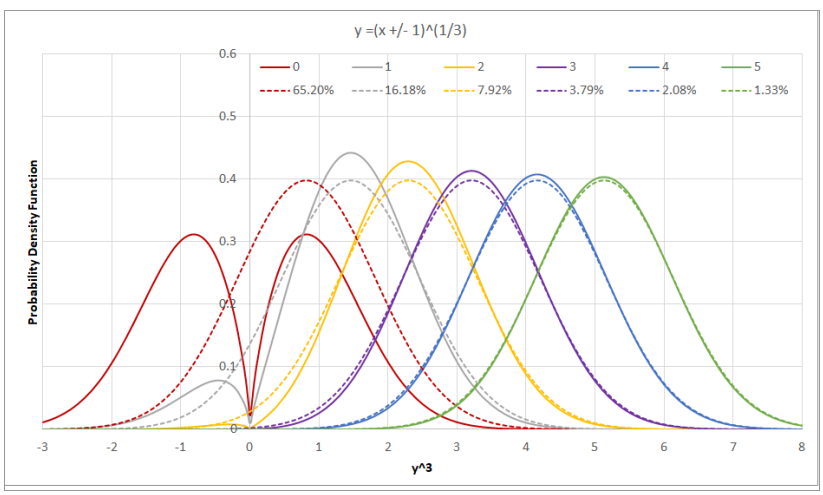

Figure 6: The probability density function for $\sqrt[3]{x \pm 1}$, for different x as shown in the legend. The y-axis is scaled as $\tilde{y}^3$. Each probability density function $\rho(\tilde{x})$ in solid line is compared with a Gaussian distribution $\varrho(\tilde{x})$ of the same mode and the same deviation in the dash line of the same color. The legend for the Gaussian distribution shows the value of Gaussian difference in percentage.

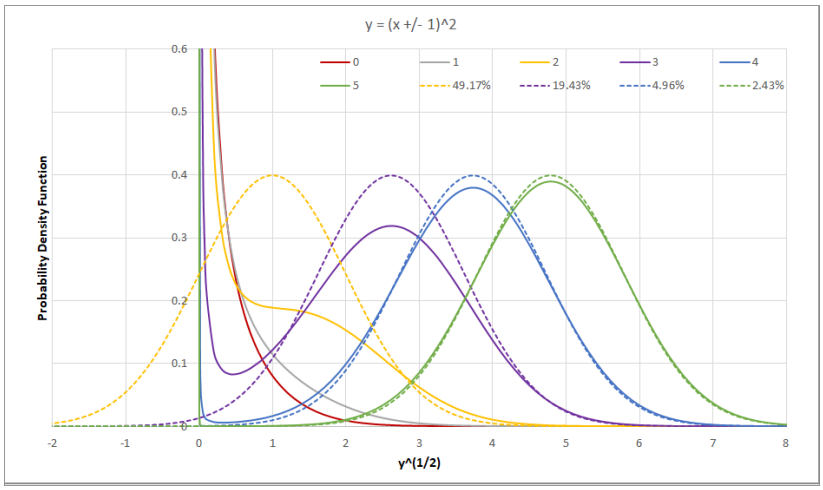

Figure 7: The probability density function for $(x \pm 1)^2$, for different x as shown in the legend. The y-axis is scaled as $\sqrt{\tilde{y}}$. Some probability density function $\rho(\tilde{x})$ in solid line is compared with a Gaussian distribution $\varrho(\tilde{x})$ of the same mode and the same deviation in the dash line of the same color. The legend for the Gaussian distribution shows the value of Gaussian difference in percentage.

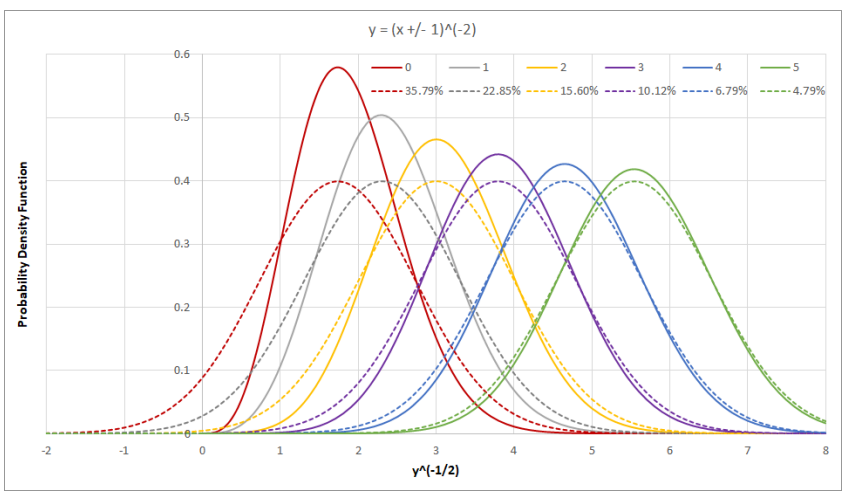

Figure 8: The probability density function for $1/(x \pm 1)^2$, for different x as shown in the legend. The y-axis is scaled as $1/\sqrt{\tilde{y}}$. The probability density function $\rho(\tilde{x})$ in solid line is compared with a Gaussian distribution $\varrho(\tilde{x})$ of the same mode and the same deviation in the dash line of the same color. The legend for the Gaussian distribution shows the value of Gaussian difference in percentage.

Figure 6 shows the probability density function for $\sqrt[3]{x \pm 1}$ according to Formula (2.19), which also has a zero at $x = 0$. Compared with $\sqrt{x \pm 1}$, $\sqrt[3]{x \pm 1}$ distributes on $\tilde{y} < 0$ also, so that the uncertainty distributions for $\sqrt[3]{0 \pm 1}$ has two equal peaks instead of one larger peak on the positive side only for $\sqrt{x \pm 1}$.

Figure 7 shows the probability density function for $(x \pm 1)^2$ according to Formula (2.19), which has a pole at $x = 0$. The uncertainty distributions for $(0 \pm 1)^2$ and $(1 \pm 1)^2$ deviate significantly from Gaussian, while the uncertainty distributions for $(4 \pm 1)^2$ and $(5 \pm 1)^2$ look Gaussian but each still with an infinitive but narrower peak at $\tilde{y} = 0$. According to Formula (2.23), $(0 \pm 1)^2$ and $(1 \pm 1)^2$ has no mode. $(0 \pm 1)^2$ is actually the χ^2 distribution [1].

Figure 8 shows the probability density function for $1/(x \pm 1)^2$ according to Formula (2.19), which has a zero at $x = 0$. Figure 8 looks more like Figure 5 than Figure 7, showing that pole or zero determines the properties of uncertainty distributions.

In all the figures, $\rho(f^{-1}(\tilde{y}), x, \delta x)$ becomes more and more Gaussian-like with decreasing $p(x)$ when the mode of the distributions is farther away from either zero or pole.

2.10 Analytic Functions

Formula (2.16) gives the uncertainty distribution of an analytic function. However, normal scientific and engineering calculations usually do not care about the result distribution, but just some simple statistics of the result, such as the result deviation [1] [2]. Using Taylor expansion is one way to get the simple statistics when the result uncertainty distribution is nearly Gaussian.

An analytic function $f(x)$ can be accurately given in a range by the Taylor series as shown in Formula (2.24). Using Formula (2.10), Formula (2.26) and (2.27) [37] gives the mean $\overline{f(x)}$ and the variance $\delta^2 f(x)$, respectively, in which $\zeta(n)$ is the *variance momentum* defined by Formula (2.25), and σ is defined as the *bounding factor*.

$$\tilde{y} = f(x + \tilde{x}) = f(x + \tilde{z}\delta x) = \sum_{n=0}^{\infty} \frac{f_x^{(n)}}{n!} \tilde{z}^n (\delta x)^n; \tag{2.24}$$

$$\zeta(n) \equiv \int_{-\sigma}^{+\sigma} \tilde{z}^n N(\tilde{z}) d\tilde{z}; \quad \zeta(2n+1) = 0; \tag{2.25}$$

$$\overline{f(x)} = \int_{-\sigma}^{+\sigma} f(x + \tilde{z}\delta x) N(\tilde{z}) d\tilde{z} = \sum_{n=0}^{\infty} (\delta x)^{2n} \frac{f_x^{(2n)} \zeta(2n)}{(2n)!}; \tag{2.26}$$

$$\begin{aligned} \delta^2 f(x) = \overline{f(x)^2} - \overline{f(x)}^2 &= \sum_{n=0}^{\infty} (\delta x)^{2n} \\ &\left(\zeta(2n) \sum_{j=1}^{2n-1} \frac{f_x^{(j)}}{j!} \frac{f_x^{(2n-j)}}{(2n-j)!} - \sum_{j=1}^{n-1} \frac{f_x^{(2j)} \zeta(2j)}{(2j)!} \frac{f_x^{(2n-2j)} \zeta(2n-2j)}{(2n-2j)!} \right); \end{aligned} \tag{2.27}$$

$$\begin{aligned} &= (\delta x)^2 (f_x^{(1)})^2 + (\delta x)^4 \left(f_x^{(1)} f_x^{(3)} + \frac{1}{2} (f_x^{(2)})^2 \right) \\ &\quad + (\delta x)^6 \left(\frac{1}{4} f_x^{(1)} f_x^{(5)} + \frac{1}{2} f_x^{(2)} f_x^{(4)} + \frac{5}{12} (f_x^{(3)})^2 \right) + o((\delta x)^8); \end{aligned} \tag{2.28}$$

The choice of the bounding factor σ needs careful considerations. If $\sigma = \infty$, $\zeta(2n) = (2n-1)!!$, which may cause Formula (2.27) to diverge in most cases. When σ is limited, Formula (2.25) becomes Formula (2.29):

- For $2n < 10$ when $5 \leq \sigma$, $\zeta(2n)$ can be approximated by $\zeta(2n) = (2n-1)!!$ according to Formula (2.30).
- For large $2n$, Formula (2.29) reduces to Formula (2.31), which shows that $\zeta(2n)$ increases slower than σ^{2n} for increasing $2n$.
- For $20 < 2n \leq 200$, when $3 \leq \sigma \leq 6$, $\zeta(2n)$ is well fitted by λ^{2n} numerically, in which λ is a fitting parameter. λ is always slightly smaller than σ empirically, as expected.

$$\zeta(2n) = 2N(\sigma)\sigma^{2n}\sum_{j=1}^{\infty}\sigma^{2j-1}\frac{(2n-1)!!}{(2n-1+2j)!!}; \tag{2.29}$$

$$= (2n-1)\zeta(\sigma, 2n-2) - 2N(\sigma)\sigma^{2n-1}; \tag{2.30}$$

$$\sigma^2 \ll 2n: \quad \zeta(2n) \simeq 2N(\sigma)\frac{\sigma^{2n+1}}{2n+1}; \tag{2.31}$$

The property of $\zeta(2n)$ when $\sigma^2 \ll 2n$ is not sensitive to the underlying uncertainty distribution. If the normal distribution $N(\tilde{z})$ in Formula (2.25) is replaced by a uniform distribution between $[-\sigma, +\sigma]$, Formula (2.32) is similar to Formula (2.31):

$$\zeta(2n) \equiv \int_{-\sigma}^{+\sigma}\frac{1}{2\sigma}\tilde{z}^{2n}d\tilde{z} = 2\frac{1}{2\sigma}\frac{\sigma^{2n+1}}{2n+1}; \tag{2.32}$$

The limited range of $\tilde{x} \in (-\sigma\delta x, +\sigma\delta x)$ causes a bounding leakage ϵ according to Formula (2.33), in which $\xi(z)$ is the cumulative density function for normal distribution [4]. When $\sigma = 5$, $\epsilon = 2 \times 10^{-7}$, which is small enough for most applications. $\sigma = 5$ is also a golden standard to assert a negative result [2].

$$\epsilon = 2 - 2\xi(\frac{1}{2} + \sigma); \tag{2.33}$$

When $\sigma \geq 5$, $\zeta(2n)$ is almost identical to $(2n-1)!!$ when $2n \leq 16$, which makes Formula (2.27) quite insensitive to the actual value of σ.

Thus, $\sigma = 5$ in variance arithmetic.

Formula (2.35) gives the result variance for e^x, which always converge:

$$e^{x+\tilde{x}} = e^x\sum_{n=0}^{\infty}\frac{\tilde{x}^n}{n!}; \tag{2.34}$$

$$P(e^x)^2 = \frac{\delta^2 e^x}{\overline{e^x}^2} \simeq \frac{\delta^2 e^x}{(e^x)^2} = \sum_{n=1}^{\infty}(\delta x)^{2n}\left(\sum_{j=1}^{2n-1}\frac{\zeta(2n)}{j!\,(2n-j)!} - \sum_{j=1}^{n-1}\frac{\zeta(2j)}{(2j)!}\frac{\zeta(2n-2j)}{(2n-2j)!}\right) \tag{2.35}$$

$$= (\delta x)^2 + \frac{3}{2}(\delta x)^4 + \frac{7}{6}(\delta x)^6 + \frac{5}{8}(\delta x)^8 + o((\delta x)^{10}); \tag{2.36}$$

Formula (2.38) gives the result variance for $\ln(x)$:

$$\ln(x+\tilde{x}) - \ln(x) = \ln(1+\frac{\tilde{x}}{x}) = \sum_{j=1}^{\infty}\frac{(-1)^{j+1}}{j}\frac{\tilde{x}^j}{x^j}; \tag{2.37}$$

$$\delta^2\ln(x) = \sum_{n=1}^{+\infty}P(x)^{2n}\left(\sum_{j=1}^{2n-1}\frac{\zeta(2n)}{j(2n-j)} - \sum_{j=1}^{n-1}\frac{\zeta(2j)}{2j}\frac{\zeta(2n-2j)}{2n-2j}\right) \tag{2.38}$$

$$= P(x)^2 + P(x)^4\frac{9}{8} + P(x)^6\frac{119}{24} + P(x)^8\frac{991}{32} + o(P(x)^{10}); \tag{2.39}$$

Formula (2.41) gives the result variance for $\sin(x)$, which converge when $\delta x < 1$:

$$\sin(x+\tilde{x}) = \sum_{n=0}^{\infty} \sin(x)(-1)^n \frac{\tilde{x}^{2n}}{(2n)!} + \sum_{n=0}^{\infty} \cos(x)(-1)^n \frac{\tilde{x}^{2n+1}}{(2n+1)!}; \tag{2.40}$$

$$\delta^2 \sin(x) = \sum_{n=1}^{\infty} (\delta x)^{2n} (-1)^{n-1} \left(\cos(x)^2 \sum_{j=0}^{n-1} \frac{\zeta(2n)}{(2j+1)!(2n-2j-1)!} - \sin(x)^2 \sum_{j=1}^{n-1} \frac{\zeta(2n)-\zeta(2j)\zeta(2n-2j)}{(2j)!(2n-2j)!} \right) \tag{2.41}$$

$$= (\delta x)^2 \cos(x)^2 - (\delta x)^4 (\cos(x)^2 \frac{3}{2} - \frac{1}{2}) + (\delta x)^6 (\cos(x)^2 \frac{7}{6} - \frac{1}{2}) + o((\delta x)^8); \tag{2.42}$$

Formula (2.44) gives the result variance for x^c, in which $\binom{c}{n} = \frac{\prod_{j=0}^{n-1}(c-j)}{n!}$:

$$(x+\tilde{x})^c = x^c (1+\frac{\tilde{x}}{x})^c = x^c \sum_{n=1}^{\infty} \frac{\tilde{x}^n}{x^n} \binom{c}{n}; \tag{2.43}$$

$$P(x^c)^2 = \frac{\delta^2 x^c}{\overline{x^c}^2} \simeq \frac{\delta^2 x^c}{(x^c)^2} = \sum_{n=1}^{\infty} P(x)^{2n} \left(\sum_{j=1}^{2n-1} \zeta(2n) \binom{c}{j} \binom{c}{2n-j} - \sum_{j=1}^{n-1} \zeta(2j) \binom{c}{2j} \zeta(2n-2j) \binom{c}{2n-2j} \right) \tag{2.44}$$

$$= c^2 P(x)^2 + \frac{3}{2} c^2 (c-1)(c-\frac{5}{3}) P(x)^4 + \frac{7}{6} c^2 (c-1)(c-2)^2 (c-\frac{16}{7}) P(x)^6 + o(P(x)^8); \tag{2.45}$$

As the special cases for Formula (2.44), Formula (2.46) gives the result variance for x^2, Formula (2.47) gives the result variance for $\sqrt{x}$, Formula (2.48) gives the result variance for $1/x$, and Formula (2.49) gives the result variance for $1/x^2$:

$$P(x^2)^2 = 4P(x)^2 + 2P(x)^4; \tag{2.46}$$

$$P(\sqrt{x})^2 = \frac{1}{4} P(x)^2 + \frac{7}{32} P(x)^4 + \frac{75}{128} P(x)^6 + o(P(x)^8); \tag{2.47}$$

$$P(1/x)^2 = P(x)^2 + 8P(x)^4 + 69P(x)^6 + o(P(x)^8); \tag{2.48}$$

$$P(1/x^2)^2 = 4P(x)^2 + 66P(x)^4 + 960P(x)^6 + o(P(x)^8); \tag{2.49}$$

If Formula (2.27) can be expressed as $\sum_{n=1}^{\infty} \upsilon(2n)\zeta(2n)P(x)^{2n}$, such as Formula (2.38) and (2.44), and if $|\upsilon(2n)| \sim \lambda^{2n}$, then Formula (2.27) converges if $P(x) < 1/(\lambda\sigma)$. λ depends on $f(x)$, e.g., it is larger for $1/x$ than for $\sqrt{x}$. The maximal $P(x)$ for Formula (2.27) in the $P(x)$-form to converge is defined as the *applicable precision threshold*, which can be estimated as $1/\sigma$ according to Formula (2.31).

Formula (2.27) breaks down near a zero or pole, when the underlying uncertainty distribution deviates significantly from Gaussian, which is the case for x^c at $x = 0$, as shown in Figure 5, 6, 7, and 8. Thus, for $(x+a)^c$ in which a is another constant, $P(x) = \delta x/|x-a|$ in Formula (2.44).

The variance formula using Taylor expansion shows the nature of the calculation, such as $\delta x \to P(e^x)$, $\delta x \to \delta \sin(x)$, $P(x) \to P(x^c)$, and $P(x) \to \delta \ln(x)$, or the difference speed of variance increases of x^c depending on different c.

Due to the uncorrelated uncertainty assumption, the Taylor expansion can be applied to multiple inputs, such as Formula (2.52) [37] .

$$f(x+\tilde{x}, y+\tilde{y}) = \sum_{m=0}^{\infty}\sum_{n=0}^{\infty} \frac{f_{(x,y)}^{(m,n)}}{m!n!}\tilde{x}^m\tilde{y}^n; \tag{2.50}$$

$$\begin{aligned}\overline{f(x,y)} &= \int\int f(x+\tilde{x}, y+\tilde{y})\rho(\tilde{x}, x, \delta x)\rho(\tilde{y}, y, \delta y)\ d\tilde{x}d\tilde{y} \\ &= \sum_{m=0}^{\infty}\sum_{n=0}^{\infty}(\delta x)^{2m}(\delta y)^{2n}\frac{\zeta(2m)\zeta(2n)f_{(x,y)}^{(2m,2n)}}{(2m)!(2n)!};\end{aligned} \tag{2.51}$$

$$\begin{aligned}\delta^2 f(x,y) = \overline{f(x,y)^2} - \overline{f(x,y)}^2 &= \sum_{m=0}^{\infty}\sum_{n=0}^{\infty}(\delta x)^{2m}(\delta y)^{2n} \\ &(\zeta(2m)\zeta(2n)\sum_{i=0}^{2m}\sum_{j=0}^{2n}\frac{f_{(x,y)}^{(i,j)}}{i!\ j!}\frac{f_{(x,y)}^{(2m-i,2n-j)}}{(2m-i)!\ (2n-j)!} \\ &-\sum_{i=0}^{m}\sum_{j=0}^{n}\frac{\zeta(2i)\zeta(2j)f_{(x,y)}^{(2i,2j)}}{(2i)!(2j)!}\ \frac{\zeta(2m-2i)\zeta(2n-2j)f_{(x,y)}^{(2m-2i,2n-2j)}}{(2m-2i)!(2n-2j)!}) \\ &\simeq (\delta x)^2(f_{(x,y)}^{(1,0)})^2 + (\delta y)^2(f_{(x,y)}^{(0,1)})^2 \\ &+ (\delta x)^4 f_{(x,y)}^{(1,0)}(\frac{1}{2}f_{(x,y)}^{(2,0)} + f_{(x,y)}^{(3,0)}) + (\delta y)^4(\frac{1}{2}f_{(x,y)}^{(0,2)} + f_{(x,y)}^{(0,1)}f_{(x,y)}^{(0,3)}) \\ &+ (\delta x)^2(\delta y)^2((f_{(x,y)}^{(1,1)})^2 + f_{(x,y)}^{(0,1)}f_{(x,y)}^{(2,1)} + f_{(x,y)}^{(1,0)}f_{(x,y)}^{(1,2)});\end{aligned} \tag{2.52, 2.53}$$

Formula (2.8) and (2.14) are special cases of Formula (2.52).

2.11 Dependency Tracing

$$\delta^2(f+g) = \delta^2 f + \delta^2 g + 2(\overline{fg} - \overline{f}\overline{g}); \tag{2.54}$$

$$\delta^2(fg) = \overline{f^2g^2} - \overline{fg}^2; \tag{2.55}$$

$$\delta^2 f(g) = \overline{f(g)^2} - \overline{f(g)}^2; \tag{2.56}$$

$$\delta^2(c_1 f + c_0) = c_1^2\delta^2 f; \tag{2.57}$$

When the inputs obeys the uncorrelated uncertainty assumption, variance arithmetic uses statistics to trace dependency between uncertainties. For example:

- Formula (2.54) shows $\delta^2(f+g)$, whose dependence tracing can be demonstrated by $\delta^2(f-f) = 0$ and $\delta^2(f(x)+g(y)) = \delta^2 f + \delta^2 g$, with the latter case corresponding to Formula (2.8).
- Formula (2.55) shows $\delta^2(fg)$, whose dependence tracing can be demonstrated by $\delta^2(f/f) = 0$ and $\delta^2(f(x)g(y)) = \overline{f}^2(\delta^2 g) + (\delta^2 f)\overline{g}^2 + (\delta^2 f)(\delta^2 g)$, with the latter case corresponding to Formula (2.14).
- Formula (2.56) shows $\delta^2 f(g(x))$, whose dependence tracing can be demonstrated by $\delta^2(f^{-1}(f)) = \delta^2 x$.

Formula	Result difference	Wrong independence
(1.7)	0	
(1.8)	$4x(\delta x)^2$	between x^2 and x
(1.9)	$(-2x^2+2x)(\delta x)^2-(\delta x)^4$	between $x-1$ and x

Table 1: The dependency problem when applying variance arithmetic without dependency tracing for x^2-x when the input variance is $(\delta x)^2$. The correct result variance is $(2x-1)^2(\delta x)^2+2(\delta x)^4$.

- Formula (2.57) shows the variance of linear transformation of a function, which can be applied to Formula (2.54) and (2.55) for more generic dependency tracing.

Because of dependency tracing using Taylor expansion, variance arithmetic does not suffer from the dependency problem for analytic expressions [21] which has plagued interval arithmetic.

2.12 Progressive Execution and Dependency Problem

Dependency tracing requires final analytic solution to apply Taylor expansion for the result mean and variance, such as using Formula (2.27) and (2.52). It conflicts with traditional numerical algorithms which execute progressively. For example:

- Traditionally, x^2-x can be calculated equivalently as Formula (1.7), (1.8), and (1.9). When applying Formula (2.8), (2.14), and (2.46) as separated and independent arithmetic operations, only Formula (1.7) gives the correct result, while the other two violate dependency tracing and give wrong results, as shown in Table 1. It shows that the dependency problem of interval arithmetic [21] may be due to the lack of dependency tracing.
- In Gaussian elimination for linear equation [12], the execution is carried out progressively in the path to minimize rounding errors using conventional floating-point calculations, without any consideration for dependency tracing. The existence of different execution paths suggests that the solution itself is designed with dependency problem. Instead, in Section 6, variance arithmetic uses the matrix inversion formula by the determinants of sub-matrices, which is advised against by traditional numerical algorithm [12].
- Formula (2.58) is the single-variable version of Formula (2.56). If instead, $\delta^2 f(g(x))$ is calculated as $\delta^2 f(x)$ with $\delta^2 x=\delta^2 g(x)$, the result Formula (2.59) depends on the sequence of the composite functions, such as $P(\sqrt{x}^2)^2\simeq P(x)^2+\frac{9}{2}P(x)^4$ v.s. $P(\sqrt{x^2})^2\simeq P(x)^2+\frac{9}{8}P(x)^4$.

Thus, when the dependency tracing is not applied strictly, variance arithmetic will have dependency problem. To avoid such dependency problem, all the conventional numerical algorithms need to be reevaluated for variance arithmetic.

$$\begin{aligned}\delta^2 f(g(x)) &= (\delta x)^2(f_x^{(1)}g_x^{(1)})^2+(\delta x)^4((f_x^{(1)})^2(g_x^{(1)}g_x^{(3)}+\frac{1}{2}(g_x^{(2)})^2)+\frac{1}{2}(f_x^{(2)})^2(g_x^{(1)})^4\\ &\quad+f_x^{(1)}f_x^{(2)}(g_x^{(1)})^4+4f_x^{(1)}f_x^{(2)}(g_x^{(1)})^2g_x^{(2)})+o((\delta x)^6); \qquad (2.58)\\ \delta^2 f(x)|_{\delta^2 x=\delta^2 g(x)} &= (\delta x)^2(f_x^{(1)}g_x^{(1)})^2+(\delta x)^4((f_x^{(1)})^2(g_x^{(1)}g_x^{(3)}+\frac{1}{2}(g_x^{(2)})^2)+\frac{1}{2}(f_x^{(2)})^2(g_x^{(1)})^4\\ &\quad+f_x^{(1)}f_x^{(3)}(g_x^{(1)})^4)+o((\delta x)^6); \qquad (2.59)\end{aligned}$$

2.13 A Variance Representation

To be compatible with the conventional 64-bit floating-point standard [10], a variance representation has the following bit assignment for Formula (2.11) and (2.12):

- The bit count for E is 11.
- The bit counts for S and V are 53 respectively.
- The representation needs 1 bit for sign, 1 bit for the carry error sign for S, and 1 bit for the carry error sign for V, respectively.
- The representation stores the bounding leakage ϵ as an 8-bit floating-point value, with 4-bit for the exponent. The maximal value for ϵ is $15\ 2^{15-19} = 15/16$, while the minimal non-zero value for ϵ is $2^{-19} = 1.90710^{-6}$ which is equivalent to $\sigma = 4.763$. The result bounding leakage is approximated as $\epsilon = \epsilon_1 + \epsilon_2 - \epsilon_1\epsilon_2$ in which ϵ_1 and ϵ_2 are the bounding leakages of the two operands for a calculation, respectively.

The total bit count of each imprecise value in the variance representation is 128.

In this variance representation, the precision 2^{-p} of an imprecise value determines the *significance bit count* which is the bit count to record S besides the leading 0s:

1. To represent a value 1 ± 2^{-p}, V is normalized as 2^{52}.
2. According to Formula (2.12), $\delta^2 x = (2^{-p})^2 = V\ 2^{2E} \Rightarrow E = -26 - p$.
3. According to Formula (2.11), $x = 1 = S\ 2^E \Rightarrow S = 2^{26+p}$.

Thus, the effective bit counts for S and V depend on the statistical precision $P \equiv \delta x/|x|$ of an imprecise value x:

- When $P \geq 2^{-26} \simeq 1.5\ 10^{-8}$, the significance bit count is $26 - \log_2 P$, which means maximally 26-bit calculated inside uncertainty.
- When $2^{-26} > P > 2^{-53} \simeq 1.1\ 10^{-16}$, the maximal significance bit count is kept at 53, and V decreases toward 0 for larger P.
- When $2^{-53} \geq P$, V becomes 0, so that the variance representation represents a precise value which is identical to the conventional 64-bit floating-point value.

Thus, variance arithmetic is a modern implementation of significance arithmetic [32] [33] [34].

3 Verification Method For Variance Arithmetic

3.1 Verification Standards [37]

Analytic functions each with a precisely known result are used to characterize a uncertainty-bearing arithmetic. The difference between the arithmetic result and the analytic result is defined as the *value error*. The statistics of the value errors defines the *tracking ratio*:

- For variance arithmetic, it is the ratio of the standard deviation of the value error to the output deviation.
- For interval arithmetic, it is the ratio of the range of the value error to the output range.

An ideal tracking ratio should be close to 1, regardless of input uncertainty precision, amount of calculation, and the nature of the calculation.

3.2 Verification Method [37]

For an analytic function with know result, Gaussian noise or white noise [1] of specified uncertainty deviation are added to input data, and statistical methods are used to characterize the corresponding value errors.

3.3 Types of Uncertainties [37]

There are four sources of result uncertainty for 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 variance 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 variance 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 variance arithmetic, the result precision of a calculation is determined by the inputs and the calculation itself. Section 8 and 9 will provide such cases of applying truncation rule to Taylor expansion and numerical integration, respectively.

The *modelling errors* arise when an approximate analytic solution is used, or when a real problem is simplified to obtain the solution. For example, Section 5 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 variance arithmetic.

3.4 Types of Calculations to Verify [37]

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:

- transformation
- generation
- reduction

A *transformation* 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 according to Formula (**??**), which provides an objective testing standard for a uncertainty-bearing arithmetic. A test of uncertainty-bearing arithmetic using FFT algorithms is provided in Section 5, and a test of matrix inversion is provided in Section 6.

A *generation* 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 7 demonstrates such an algorithm, which calculates a table of the sine function using trigonometric relations and two precise input data, $sin(0) = 0$ and $sin(\pi/2) = 1$. The generated data have different precision.

A *reduction* 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 9.

4 Verification Using Math Library Functions

Formula (2.35), (2.38), (2.41), and (2.44) are tested and verified by the corresponding math library functions *exp*, *log*, *sin*, and *pow*, respectively.

Figure 9 shows that the measured result deviations agree very well with the calculated result deviations using Formula (2.41). Smaller input uncertainty always result in smaller output uncertainty. Figure 9 also shows that $\delta^2 \sin(x)$ has the same periodicity as $\sin(x)$.

- When $x = 0$, $\sin(x) \simeq x$, so that $\delta^2 \sin(x) \simeq (\delta x)^2$.
- When $x = \pi/2$, $\sin(x) \simeq 1$, so that $\delta^2 \sin(x) \simeq 0$.

Figure 10 shows that the measured result deviations agree very well with the calculated result deviations of $(1 \pm \delta x)^c$ using Formula (2.44) except when $\delta x = 0.2$ and $c < -1$. The measured results of using white noise is also displayed in Figure 10, which are always slightly smaller than the corresponding results using Gaussian noise, probably due to the long tails of Gaussian distributions. As expected, the results of using Gaussian noise fit better with the calculated result of variance arithmetic, especially when the deviation is larger. However, using white noise has one advantage over using Gaussian noise: It shows clearer the effect of the non-linearity of $f(x)$ in Formula (2.16), e.g., the result distribution deviates more from a uniform distribution for a larger input deviation.

The reason why Formula (2.44) is not applicable at the estimated applicable precision threshold $1/\sigma$ needs further discussion.

- Formula (2.46), (2.47), (2.48), and (2.49) show that $(1 \pm 0.2)^c$ diverges faster when $c < -1$, which is confirmed by Figure 12, in which $(1 \pm 0.2)^{-2}$ clearly diverges.
- When the deviation is reduced slightly to below the estimated applicable precision threshold $1/\sigma$, Figure 12 shows that Formula (2.49) converges for $(1 \pm 0.195)^{-2}$, which is confirmed by Figure 10.
- Figure 11 shows that the measured probability density distributions for $(1 \pm 0.2)^{-2}$ and $(1 \pm 0.2)^{-0.195}$ are quite similar, suggesting that the convergence of $(1 \pm 0.195)^{-2}$ and the divergence of $(1 \pm 0.2)^{-2}$ is not related to underlying data.

Thus, $(x \pm \delta x)^{-2}$ has an applicable precision threshold lower than 0.2, so that it needs to be calculated with a $\sigma < 5$, and the result has a larger bounding leakage. For the discussion simplicity, the cases of applicable precision thresholds less than 1/5 are avoided in this paper.

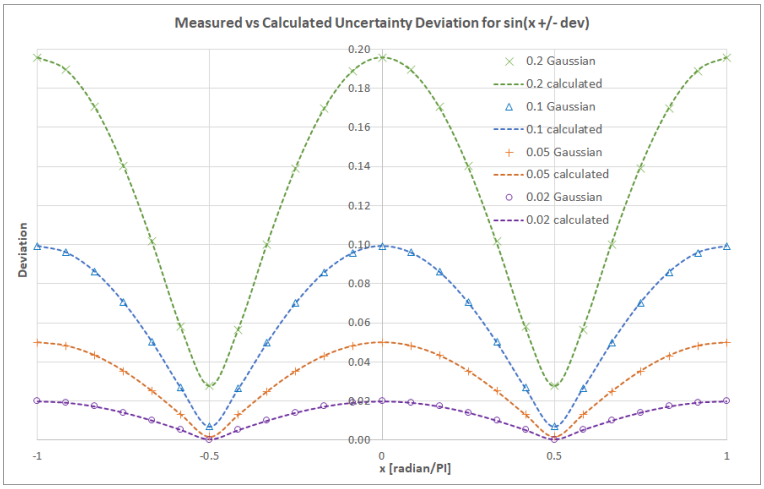

Figure 9: The measured vs calculated uncertainty deviation for $\sin(x \pm \delta x)$, for different δx as shown in the legend, in which "Gaussian" means using Gaussian noise, and "calculated" means the result from variance arithmetic. The x-axis is in radians divided by π, from -1 to $+1$. The calculated values are obtained by Formula (2.41), while each measured deviation is obtained by 10000 samples using library *sin* function.

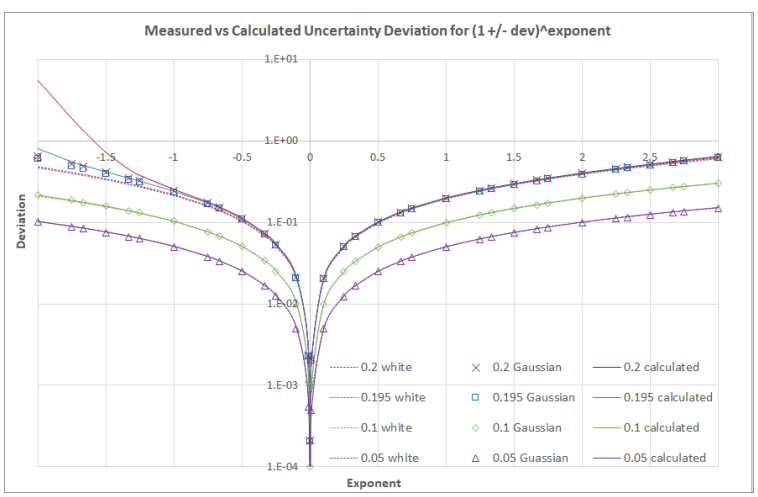

Figure 10: The measured vs calculated uncertainty deviation for $(1 \pm \delta x)^c$, for different δx as shown in the legend, in which "white" means using white noise, "Gaussian" means using Gaussian noise, and "calculated" means the result from variance arithmetic. The x-axis is the exponent c from -2 to $+3$. The calculated values are obtained by Formula (2.44), while each measured deviation is obtained by 10000 samples using library *pow* function.

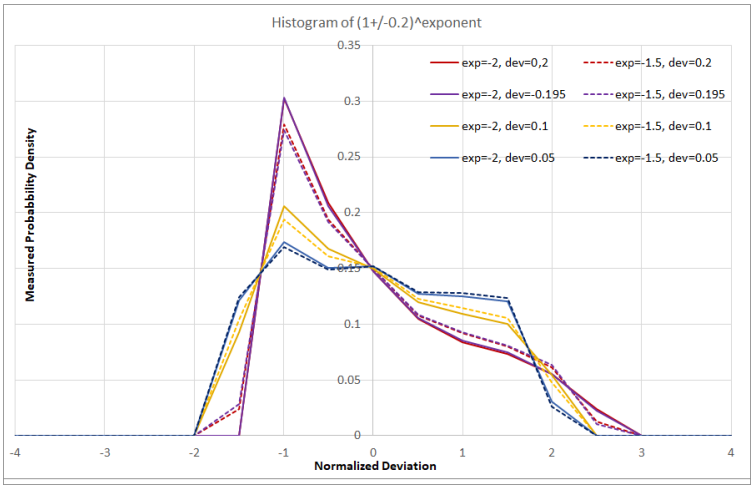

Figure 11: The measured probability density of $(1 \pm \delta x)^c$ for different $exp{=}c$ and $dev{=}\delta x$ as shown in the legend. The x-axis is the normalized deviation from the mean of $(1 \pm \delta x)^c$. The y-axis is the measured histogram at each x.

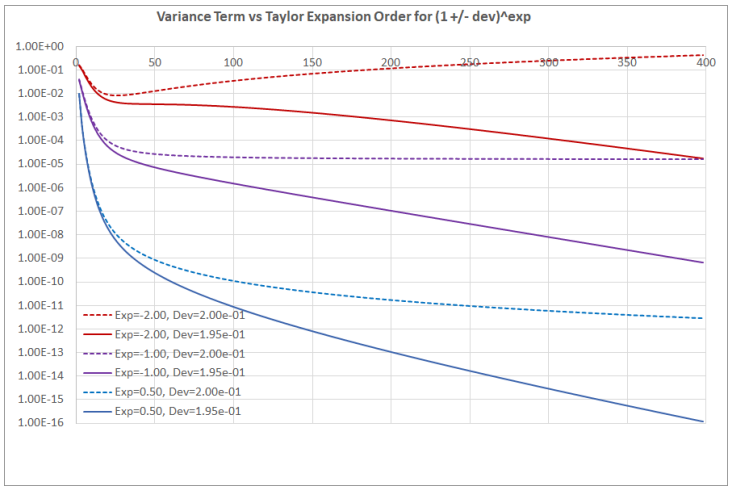

Figure 12: The individual variance term of $(1 \pm \delta x)^c$ for the different Taylor expansion order $2n$, different $exp{=}c$ and $dev{=}\delta x$ as shown in the legend. The x-axis is Taylor expansion order $2n$ from 1 to 400 in Formula (2.27). The y-axis is the contribution to $(1 \pm \delta x)^c$ at each Taylor expansion order $2n$ according to Formula (2.44). It shows that $(1 \pm 0.2)^{-2}$ clearly diverges, $(1 \pm 0.2)^{-1}$ seems diverges, and other cases clearly converge.

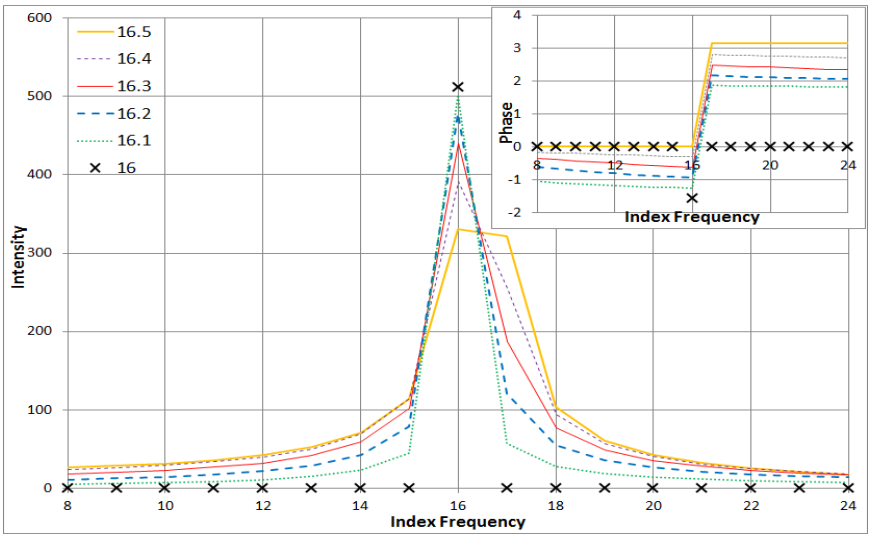

Figure 13: Unfaithful representations of perfect sine signals in the Discrete Fourier Transformation. The calculation is done on 1024 samples using FFT on a series of perfect sine signals having amplitude of 1 and slightly different frequencies as shown in legends. In the drawing, x axis shows frequency, 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.

5 Verification Using FFT

5.1 Unfaithful Frequency Response of Discrete Fourier Transformation [37]

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 discrete Fourier transformation is only faithful for Fourier transformation at certain frequencies, and it has a different degree of faithfulness for other frequencies. This is called the frequency response of the discrete Fourier transformation in this paper.

For each signal sequence $h[k], k = 0, 1 \ldots N-1$, in which N is a positive integer, the discrete Fourier transformation $H[n], n = 0, 1 \ldots N-1$ and its reverse transformation is given by Formula (5.1) [12], in which k is the *index frequency* for the discrete Fourier transformation:

$$H[n] = \sum_{k=0}^{N-1} h[k]\, e^{i2\pi \frac{k}{N} n}; \qquad h[k] = \frac{1}{N} \sum_{n=0}^{N-1} H[n]\, e^{-i2\pi \frac{n}{N} k}; \tag{5.1}$$

The $H[n]$ of a pure sine signal $h[k] = \sin(2\pi f k/N)$ is calculated by Formula (5.2), in which f is the frequency of the sine wave. The solution for Formula (5.2) is Formula (5.3), which transform the sine signal to a delta-like function with $\pi/2$ as phase only

when f is an integer between 0 and $N/2$. In other cases, how much the result of discrete Fourier transformation deviates from continuous Fourier transformation on how much f deviates from an integer, e.g., when f is exactly between two integers, the phase of the transformation is that of cosine instead of sine according to Formula (5.3). Examples of unfaithful representations of fractional frequency by the discrete Fourier transformation are shown in Figure 13.

$$H[n] = \frac{1}{2i}\left(\sum_{k=0}^{N-1} e^{i2\pi(n+f)\frac{k}{N}} - \sum_{k=0}^{N-1} e^{i2\pi(n-f)\frac{k}{N}}\right) \tag{5.2}$$

$$= \begin{cases} iN/2, & f \text{ is integer} \\ N/\pi, & f \text{ is integer} + 1/2 \\ \frac{1}{2}\frac{\sin(2\pi f - 2\pi\frac{f}{N}) + \sin(2\pi\frac{f}{N}) - \sin(2\pi f)e^{-i2\pi\frac{n}{N}}}{\cos(2\pi\frac{n}{N}) - \cos(2\pi\frac{f}{N})} & \text{otherwise} \end{cases} \tag{5.3}$$

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 discrete Fourier transformation 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 artifacts 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 discrete Fourier transformation 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 discrete Fourier transformation. The continuous Fourier transformation has an infinitive signal range so that:

$$h(t) \Leftrightarrow H(s): \quad h(t-\tau) \Leftrightarrow H(s)e^{i2\pi s\tau}; \tag{5.4}$$

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

$$G[n] = (H[n] + h[N] - h[0])e^{i2\pi n/N}; \tag{5.5}$$

Applying Formula (5.4) to Formula (5.5) results in Formula (5.6).

$$h[N] = h[0]; \tag{5.6}$$

Thus, the discrete Fourier transformation has an implied assumption that the signal $h[k]$ repeats itself outside the region of $[0, N-1]$ [42]. 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 sine or cosine signals with frequencies as an integer-fold of $2\pi/N$.

5.2 FFT (Fast Fourier Transformation) [37]

Forward:

$$
\begin{aligned}
L=1: o=0: & \quad [0,1]; \\
o=1: & \quad F=[0+1,0-1]; \\
L=2: o=0: & \quad [0,2,1,3]; \\
o=1: & \quad [0+2,0-2,1+3,1-3]; \\
o=2: & \quad [0+1+2+3,0+i1-2-i3,0-1+2-3,0-i1-2+i3; \\
[0,1,0,-1]: & \quad [0,i2,0,-i2]; \\
[1,0,-1,0]: & \quad [0,2,0,2];
\end{aligned}
$$

Reverse:

$$
\begin{aligned}
L=1: o=0: & \quad [0,1]; \\
o=1: & \quad F=[0+1,0-1]; \\
L=2: o=0: & \quad [0,2,1,3]; \\
o=1: & \quad [0+2,0-2,1+3,1-3]; \\
o=2: & \quad [0+1+2+3,0-i1-2+i3,0-1+2-3,0+i1-2-i3; \\
[0,i2,0,-i2]: & \quad [0,4,0,-4]; \\
[0,2,0,2]: & \quad [4,0,-4,0];
\end{aligned}
$$

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

$$m=L: \quad H[n,\frac{k}{2^m}]=h[j], k,n=0,1\ldots N-1; \tag{5.7}$$

$$m=L-1\ldots 0: \quad H[n,\frac{k}{2^m}]=H[n,\frac{k}{2^m}]+H[n,\frac{k}{2^{m+1}}]\ \exp(+i2\pi\frac{n}{2^{L-m}}); \tag{5.8}$$

$$m=0: \quad H[n]=H[n,\frac{k}{2^m}]; \tag{5.9}$$

Thus, each output value is obtained after applying Formula (5.8) L times. L is called the FFT order in this paper.

The calculation of the term $\exp(i2\pi\frac{n}{2^{L-m}})$ in Formula (5.8) can be simplified. Let $\ll$ denote a bit left-shift operation and let & denote a bitwise AND operation:

$$\varphi[n]\equiv\exp(i2\pi\frac{n}{2^L}): \quad \exp(i2\pi\frac{n}{2^{L-m}})=\varphi[(n\ll m)\&((1\ll L)-1)]; \tag{5.10}$$

Formula (5.8) always sums up two mutually independent operands.

5.3 Evaluating Calculation Inside Uncertainty

Figure 14 shows the output deviations and value errors for a noisy sine signal after forward FFT. It shows that the output deviations using variance 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 variance arithmetic, output deviations of Formula (5.8) are no longer a constant. One interesting consequence is that only in variance arithmetic the output deviations for a noisy input signal are larger than those for a corresponding clean input signal.

Figure 14 shows that the value errors calculated using variance arithmetic are comparable to those using conventional floating-point arithmetic, and they are both comparable to the output deviations using either variance 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 15 compares the output value errors of variance 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 14, Figure 15 shows that the result using variance arithmetic with the 4-bit calculation inside uncertainty approaches that using independence arithmetic so that the 4-bit calculation inside uncertainty seems sufficient. variance arithmetic with the 4-bit calculation inside uncertainty is used for further tests.

5.4 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 16 and Figure 17 show that such histograms using wither independence arithmetic or variance 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 variance arithmetic, e.g., during normalization, all values which is less than 4-fold of resolution becomes 0. This phenomenon is confirmed by Figure 17.

variance 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 variance 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 17, 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.

5.5 Evaluating Uncertainty-Tracking

Figure 18 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 19 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 18 and Figure 19 are modeled by Formula (5.11), 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:

$$\delta y = \alpha \beta^L \delta x; \tag{5.11}$$

β measures the propagation speed of the deviation with an increased amount of calculation in Formula (5.11). It is called *propagation base rate*. Unless β is close to 1, β dominates α in fitting, thus determining characteristics of Formula (5.11).

It turns out that Formula (5.11) is a very good fit for both average output deviations and value errors for all three arithmetics, such as demonstrated in Figure 20. 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 (5.12) and Formula (5.13), 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 (5.11) for average output deviations and value errors, respectively:

$$z = \alpha \beta^L; \tag{5.12}$$

$$\alpha = \alpha_{err}/\alpha_{dev}; \quad \beta = \beta_{err}/\beta_{dev}; \tag{5.13}$$

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 21, which shows that the average output tracking ratios using variance 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 (5.11) and Formula (5.12) 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 (5.8).
- 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 (5.11) and Formula (5.14) , in which $(\alpha_{for}, \beta_{for})$ are corresponding fitting parameters of Formula (5.11) for the forward FFT, while the average output tracking ratios are expected to obey Formula (5.13) with the same α and β as those of the forward FFT.

$$\alpha = \alpha_{for}; \quad \beta = \beta_{for}/2; \tag{5.14}$$

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 (5.11) and Formula (5.15), in which $(\alpha_{for}, \beta_{for})$ and $(\alpha_{rev}, \beta_{rev})$ are corresponding fitting parameters of Formula (5.11) for the forward FFT and the reverse FFT, respectively. Its tracking ratios are expected to fit Formula (5.12) and Formula (5.15), in which $(\alpha_{for}, \beta_{for})$ and $(\alpha_{rev}, \beta_{rev})$ are corresponding fitting parameters of Formula (5.12) for the forward FFT and the reverse FFT, respectively.

$$\alpha = \alpha_{for}\alpha_{rev}; \quad \beta = \beta_{for}\beta_{rev}; \tag{5.15}$$

Figure 22, Figure 23 and Figure 24 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 (5.13). The estimated β for average uncertainty deviations and value errors for the reverse FFT and the roundtrip FFT are obtained from Formula (5.14) and Formula (5.15), 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 22 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.
- variance 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 (**??**) 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 25 shows that for the forward FFT, the measured average output tracking ratios using either variance arithmetic or independence arithmetic are approximately constant of 0.8 in both cases, regardless of the FFT order. In contrast, Figure 25 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 variance arithmetic is better than the indirect uncertainty tracking provided by interval arithmetic.

Figure 26 shows that using variance 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, variance arithmetic restores the original signal and the corresponding uncertainty for FFT. Such behavior seems ideal for a reversible algorithm. In contrast, Figure 27 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.

5.6 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 (5.11) well using any arithmetic experimentally. Formula (5.12) and Formula (5.13) can be used to estimate the maximal bounding ratio as well. For example, Figure 28 shows that the maximal output bounding ratios using variance arithmetic fit Formula (5.12) well. Unlike average output tracking ratios in Figure 25, the maximal output bounding ratios increase slowly with the FFT order using either variance 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 29 shows that the deviation leakages is roughly a constant using variance arithmetic, suggesting the statistical nature of uncertainty bounding using variance arithmetic. Whether variance 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 variance arithmetic may not be satisfied.
- Otherwise, variance arithmetic should be more suitable for normal usages.

6 Comparison Using Matrix Inversion

6.1 Uncertainty Propagation in Matrix Determinant

Let vector $[p_1, p_2 \dots p_n]_n$ denote a permutation of the vector $(1, 2 \dots n)$ [44]. Let $\$[p_1, p_2 \dots p_n]_n$ denote the permutation sign of $[p_1, p_2 \dots p_n]_n$ [44]. 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 (6.1) [12] and let the sub-determinant at index (i, j) be defined as Formula (6.2) [44]:

$$|M| \equiv \sum_{[p_1 \dots p_n]_n} \$[p_1 \dots p_n]_n \prod_k x_{k,p_k}; \tag{6.1}$$

$$|M|_{i,j} \equiv \sum_{[p_1 \dots p_n]_n}^{p_i = j} \$[p_1 \dots p_n]_n \prod_k^{k \neq i} x_{k,p_k}; \tag{6.2}$$

$(-1)^{i+j}|M_{(i,j)}|$ 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 (6.3) holds for the arbitrary row index i or the arbitrary column index j [12]:

$$|M| = \sum_{j=1}^{n} |M_{i,j}| x_{i,j} = \sum_{i=1}^{n} |M_{i,j}| x_{i,j}; \tag{6.3}$$

Assuming $p_1, p_2 \in \{1, 2 \dots n\}$, 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 (6.3) can be applied to $M_{i,j}$, as:

$$|M_{<i_1,i_2>_n[j_1,j_2]_n}| \equiv \sum_{[p_1 \dots p_n]_n}^{p_{i_1}=j_1, p_{i_2}=j_2} \$[p_1 \dots p_n]_n \prod_k^{k \neq i_1, k \neq i_2} x_{k,p_k}; \tag{6.4}$$

$$|M| = \sum_{j_1} x_{i_1,j_1} |M_{i_1,j_1}| = \sum_{j_1} \sum_{j_2}^{i_2 \neq i_1, j_2 \neq j_1} x_{i_1,j_1} x_{i_2,j_2} |M_{<i_1,i_2>_n[j_1,j_2]_n}|; \tag{6.5}$$

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 (6.6).

$$||M_{<i_1,i_2>_n[j_1,j_2]_n}|| = ||M|_{<i_1,i_2>_n[j_2,j_1]_n}||; \tag{6.6}$$

The definition of a sub-determinant can be extended to Formula (6.7), in which $m \in \{1, 2 \dots n\}$. Formula (6.5) can be generalized as Formula (6.8), in which $m \in \{1, 2 \dots n\}$ and $< i_1 \dots i_m >_n$ is an arbitrary ordered permutation. Formula (6.8) can be viewed as the extension for both Formula (6.3) and Formula (6.1).

$$|M_{<i_1 \dots i_m>_n[j_1 \dots j_m]_n}| \equiv \sum_{[p_1 \dots p_n]_n}^{p_{i_k}=j_k, k \in \{1 \dots m\}} \$[p_1 \dots p_n]_n \prod_{k \in \{1 \dots n\}}^{k \notin \{i_1 \dots i_m\}} x_{k,p_k}; \tag{6.7}$$

$$|M| = \sum_{[j_1 \dots j_m]_n} |M_{<i_1 \dots i_m>_n[j_1 \dots j_m]_n}| \prod_{k=1}^{m} x_{i_k,j_k}; \tag{6.8}$$

According to the basic assumption of variance 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 (6.8) results in Formula (6.9), which results in Formula (6.10) after applying Formula (2.12):

$$|\widetilde{M}| - |M| = \sum_{m=1}^{n} \sum_{<i_1 \ldots i_m>_n} \sum_{[j_1 \ldots j_m]_n} |M_{<i_1 \ldots i_m>_n [j_1 \ldots j_m]_n}| \prod_{k=1}^{m} \widetilde{y}_{i_k,j_k}; \tag{6.9}$$

$$\delta|M|^2 = \sum_{m=1}^{n} \sum_{<i_1 \ldots i_m>_n} \sum_{[j_1 \ldots j_m]_n} |M_{<i_1 \ldots i_m>_n [j_1 \ldots j_m]_n}|^2 \prod_{k=1}^{m} \delta x^2_{i_k,j_k}; \tag{6.10}$$

Defining $|M_{<>_n<>_n}| \equiv |M|$, Formula (6.11) is an recursive form of Formula (6.10):

$$\delta|M_{<p_1 \ldots p_k>_n <q_1 \ldots q_k>_n}|^2 = \sum_{p_i} \sum_{q_j} \delta x^2_{p_i,q_j}$$
$$(|M_{<p_1 \ldots p_i \ldots p_k>_n <q_1 \ldots q_j \ldots q_k>_n}|^2 + \delta|M_{<p_1 \ldots p_i \ldots p_k>_n <q_1 \ldots q_j \ldots q_k>_n}|^2); \tag{6.11}$$

When using Formula (6.3) 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 [45].

In another word, using conventional floating-point arithmetic, the calculation of determinant is one leap of faith. Instead, Formula (6.11) shows that the result uncertainty is the aggregation of uncertainties from all possible path of Formula (6.3). To accounts for all such uncertainties, Formula (6.11) 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 [44]:

$$z_{i,j} = \frac{|M_{j,i}|}{|M|}; \tag{6.12}$$

Formula (6.12) 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 [44], whose elements are not directly affected by M^{-1}. M^A is recommended to replace M^{-1} whenever the application allows [12].

6.2 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 (6.12). 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.

6.3 Testing Matrix Stability

Each matrix has a different stability [45], 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 (6.1) mean less stability of the matrix [11][12], with the Hilbert matrix [46] being the most famous unstable matrix. The condition number has been defined to quantify the stability of a matrix [45]. 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.

variance arithmetic accounts for all rounding error with stable characterization of result uncertainties. More mutual cancellations in Formula (6.1) 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 variance 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 (6.12). This hypothesis is confirmed by Figure 30, 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 31 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 (6.12) 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 32 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 [45] only gives the result sensitivity to one matrix element, Formula (6.10) contains the result sensitivity to any matrix element, any combination of matrix elements, as well as the aggregated result uncertainty deviation. Therefore, Formula (6.10) and Formula (6.11) may be better than the condition numbers for describing matrix stability.

6.4 Testing Uncertainty Propagation in Adjugate Matrix

When the adjugate matrix is calculated using variance arithmetic, Figure 33 shows that the average output deviations for the adjugate matrix increase linearly with the input deviation, which is in good agreement with Formula (5.11). Such relation is also true for maximal and average output values errors. Formula (5.11) 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 33 suggests that L increases with N^2 for the average output precision and average output error[7].

Figure 34 shows that the average output tracking ratio of the adjugate matrix using variance arithmetic is approximately a constant of 0.8. Figure 34 is very similar to Figure 21. Similar to the maximal output bounding ratios of FFT algorithms, the maximal output bounding ratios for the adjugate matrix using precision also obey Formula (5.12) 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 35, which is very similar to Figure 17. 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 variance arithmetic is a generic arithmetic for linear algorithms.

6.5 Calibration

Because $|M_{j,i}|$ and $|M|$ are not independent of each other, M^{-1} calculated by Formula (6.12) contains the dependency problem. Figure 35 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 35 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 36. Figure 37 shows that average tracking ratios

[7]The amount of calculation L does not mean the calculation complexity using the Big O notation [47]. It is just a measurement of how output uncertainty increases with a dimension of calculation according to (5.11) [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 (6.11) 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 36.

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.

7 Comparison Using Recursive Calculation of Sine Values

Starting from Formula (7.1), Formula (7.2) and Formula (7.3) can be used recursively to calculate the phase array $\varphi[n]$ in Formula (5.10).

$$\sin(0) = \cos(\frac{\pi}{2}) = 0; \qquad \sin(\frac{\pi}{2}) = \cos(0) = 1; \tag{7.1}$$

$$\sin\left(\frac{\alpha+\beta}{2}\right) = \sqrt{\frac{1-\cos(\alpha+\beta)}{2}} = \sqrt{\frac{1-\cos(\alpha)\cos(\beta)+\sin(\alpha)\sin(\beta)}{2}}; \tag{7.2}$$

$$\cos\left(\frac{\alpha+\beta}{2}\right) = \sqrt{\frac{1+\cos(\alpha+\beta)}{2}} = \sqrt{\frac{1+\cos(\alpha)\cos(\beta)-\sin(\alpha)\sin(\beta)}{2}}; \tag{7.3}$$

This algorithm is very different from both FFT and matrix inversion in nature because Formula (7.2) and Formula (7.3) 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 (7.2) and Formula (7.3) 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 38 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 39 shows that in response to the increased amount of calculations:

- The average tracking ratio for variance arithmetic is a constant about 0.25;
- The maximal output bounding ratio for variance 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 variance arithmetic is generic, regardless of the algorithms used.

8 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 variance 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 (2.24) or Formula (2.50) is no longer subjective. Formula (**??**) is decomposed into the contribution of each successive term for Tylor expansion, as Formula (8.1):

$$
\begin{aligned}
(\delta\sum_{j=0}^{J+1}a_jx^j)^2 &= \int(\sum_{j=0}^{J}a_j(x+\widetilde{y})^j-\sum_{j=0}^{J}a_jx^j+a_{J+1}(x+\widetilde{y})^{J+1}-a_{J+1}x^{J+1})^2\rho(\widetilde{y})d\widetilde{y}\\
&= \int(\sum_{j=0}^{J}a_j(x+\widetilde{y})^j-\sum_{j=0}^{J}a_jx^j)^2\rho(\widetilde{y})d\widetilde{y}+a_{J+1}^2\int(\sum_{k=1}^{J+1}C_{J+1}^k\widetilde{y}^kx^{J+1-k})^2\rho(\widetilde{y})d\widetilde{y}\\
&\quad+2\int(\sum_{j=0}^{J}a_j\sum_{k=1}^{j}C_j^k\widetilde{y}^kx^{j-k})(a_{J+1}\sum_{k=1}^{J+1}C_{J+1}^k\widetilde{y}^kx^{J+1-k})\rho(\widetilde{y})d\widetilde{y}\\
(\delta\sum_{j=0}^{J+1}a_jx^j)^2-(\delta\sum_{j=0}^{J}a_jx^j)^2 &= \sum_{k_1=1}^{J+1}\sum_{k_2=1}^{J+1}a_{J+1}^2C_{J+1}^{k_1}C_{J+1}^{k_2}M(k_1+k_2)(\delta x)^{k_1+k_2}x^{2J+2-k_1-k_2}\\
&\quad+2\sum_{k_1=1}^{J+1}\sum_{j=0}^{J}\sum_{k_2=1}^{j}a_ja_{J+1}C_{J+1}^{k_1}C_j^{k_2}M(k_1+k_2)(\delta x)^{k_1+k_2}x^{J+1+j-k_1-k_2};
\end{aligned}
\tag{8.1}
$$

Applying Formula (8.1) to Taylor expansion:

1. Formula (8.1) 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 variance 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 variance 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 (8.1) 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 (8.2) provides an example test in Taylor expansion, in which n is a positive

integer.

$$f_n(x) = \sum_{j=0}^{n} (-x)^j; \quad \lim_{n\to\infty} f_n(x) = 1/(1+x); \tag{8.2}$$

In Formula (8.2), the absolute value of $(n+1)$th term in the expansion is the Cauchy remainder estimator of the nth order expansion. Formula (8.2) 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 (8.2) as a test case, Figure 40 confirms the above Taylor expansion process using variance 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 40 repeats the above process with 4-bit calculated inside uncertainty, which only differs from Figure 40 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 42 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 43:

- 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 variance 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.

9 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:

$$x_{mid} \equiv (x_{start} + x_{end})/2; \tag{9.1}$$

$$f_{err} \equiv (f(x_{start}) + f(x_{end}))/2 - f(x_{mid}); \tag{9.2}$$

$$f_{\Delta} \equiv f(x_{mid})(x_{end} - x_{start}); \tag{9.3}$$

If f_{err} becomes insignificant, the interval $[x_{start}, x_{end}]$ is considered to be fine enough, and f_{Δ} 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 (9.4) provides an example test for the above simple algorithm, in which n is a positive integer.

$$\frac{4^{n+1} - 10^{-6(n+1)}}{n+1} = \int_{10^{-6}}^{4} x^n dx; \tag{9.4}$$

Table 2 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 2 is that even though Formula (9.3) 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 variance 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 2: Uncertainty deviation and value error of numerical integration vs. expected results using variance arithmetic for different power function. The search range is deepest near 10^{-6}.

10 Comparison Using Progressive Moving-Window Linear Regression

10.1 Progressive Moving-Window Linear Regression Algorithm

Formula (10.1) 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].

$$\begin{aligned} \alpha &= \frac{\sum_j Y_j}{\sum_j 1}; \\ \beta &= \frac{\sum_j X_j Y_j \ \sum_j 1 - \sum_j X_j \ \sum_j Y_j}{\sum_j X_j X_j \ \sum_j 1 - \sum_j X_j \ \sum_j X_j}; \end{aligned} \tag{10.1}$$

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 window's half width so that $\sum_j X_j = 0$, to reduce (10.1) into (10.2):

$$\begin{aligned} \alpha_j &= \alpha \ 2H = \sum_{X=-H+1}^{H} Y_{j-H+X}; \\ \beta_j &= \beta \ \frac{H(H+1)(2H+1)}{3} = \sum_{X=-H}^{H} X Y_{j-H+X}; \end{aligned} \tag{10.2}$$

According to Figure 44, 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 (10.2) into a progressive moving-window calculation of (10.3):

$$\begin{aligned} \beta_j &= \beta_{j-1} - \alpha_{j-1} + H(Y_{j-2H-1} + Y_j); \\ \alpha_j &= \alpha_{j-1} - Y_{j-2H-1} + Y_j; \end{aligned} \tag{10.3}$$

10.2 Dependency Problem in a Progressive Algorithm

(10.3) 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 45 shows such trend for the

average output tracking ratio using variance 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 46 shows the output deviation and the value errors vs. time while Figure 47 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 46 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 47.

In contrast, while variance 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 variance 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 variance arithmetic in Figure 46. This mechanism of error tracking in variance arithmetic is also demonstrated in Figure 48 and Figure 49. Figure 48 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 variance arithmetic in Figure 48 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 49 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 49, revealing similar underline error-tracking mechanism in both cases. Figure 49 also shows that for the same bits calculated inside uncertainty, the average tracking ratios deviate from independence at exactly the same time. Figure 48 and Figure 49 thus demonstrate a uniform and granular error tracking mechanism of the variance arithmetic for different bits calculated inside uncertainty.

Is such increase of the value errors with the increase of uncertainty deviation using variance 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 variance arithmetic. Second, the stability of the average output tracking ratios and the maximal bounding ratios of variance 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 variance 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 variance 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 50 and 52, the input deviation is also increased by 10-fold to simulate an increase in measured uncertainty.
- *Defective*: In Figure 51 and 53, 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 52 while they are not affected at all in Figure 53. 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 variance 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 variance arithmetic is still interpretable.

10.3 Choosing a Better Algorithm for Imprecise Inputs

Formula (10.3) has much less calculations than Formula (10.2), and it seems a highly efficient and optimized algorithm according to conventional criterion [12]. However, from the perspective of uncertainty-bearing arithmetic, Formula (10.3) is progressive while Formula (10.2) is expressive, so that Formula (10.2) should be better. Figure 50 and Figure 54 respectively show the output deviations and the value errors vs. time for using either Formula (10.3) or Formula (10.2) 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 (10.3), Formula (10.2) is confirmed to be a better solution for this linear regression problem.

10.4 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, variance arithmetic should be used generally in progressive algorithms.
- Figure 56 shows that the result tracking ratios of the progressive linear regression is exponentially distributed, while Figure 57 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.

11 Conclusion and Discussion

11.1 Summary

The starting point of variance 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 variance 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 variance 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 (**??**), and the result uncertainty deviation of a function given by Formula (2.27) and its multi-dimension extensions such as Formula (**??**).

Formula (5.11) is shown to describe the general uncertainty deviation propagation in variance arithmetic. The average tracking ratios and the maximal bounding ratio using variance 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 variance arithmetic is better than interval arithmetic in all tests done so far.

The statistical nature of variance 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 variance arithmetic over interval arithmetic.

variance has a central role in variance 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.
- variance 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.

11.2 Efficiency of Precision Arithmetic

variance 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 [45]. 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, variance 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 (6.11), variance 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, variance arithmetic takes order-of-magnitude more time than a single leap-of-faith calculation.

However, it is wrong to conclude that variance 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 (6.11), variance arithmetic only need one calculation. Thus, conventional floating-point arithmetic may be less efficient than variance 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.

11.3 Choose a better algorithm

Because variance 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 variance 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 variance arithmetic, which converges all input uncertainty distribution to *ubiquitously Gaussian* at the outputs, and which further *quantifies* the source of the dependency problem. Using variance 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 (6.11), 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 (10.2), becomes the better method in variance arithmetic, and vice versa.

Due to the requirement of minimizing dependence problem, variance arithmetic has much less operational freedom than conventional arithmetic and may require extensive symbolic calculations, following practices in affine arithmetic [41]. Also, the comparison relation in conventional arithmetic needs to be re-evaluated in variance arithmetic, which brings about another reason for different algorithm selection.

11.4 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 variance 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 variance arithmetic need to be studied further. For example, Formula (2.12) 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 variance 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, variance 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.

11.5 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*, 65:113–134, 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] Jagdish K. Patel; Campbell B Read Handbook of the normal distribution (2nd ed.). CRC Press. ISBN 0-8247-9342-0.

[41] C. Pennachin, M. Looks, and João 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.

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

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

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

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

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

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

12 Figures

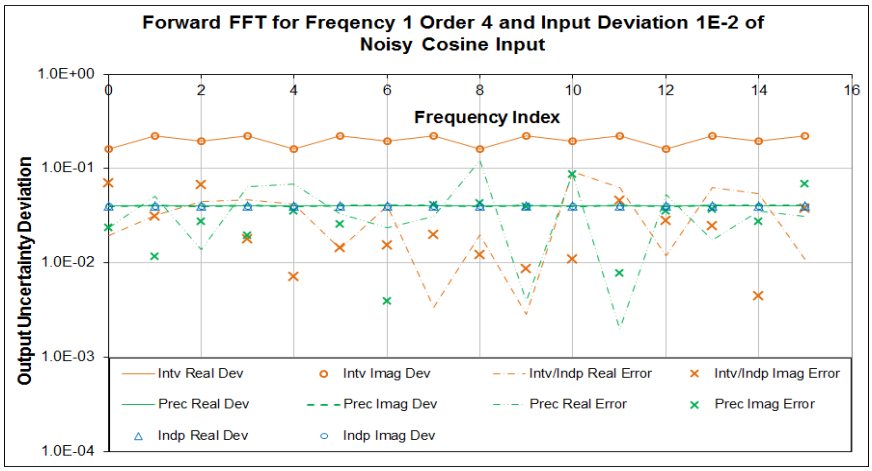

Figure 14: The output deviations and value errors of the forward FFT on a noisy sine signal of FFT order 4, index frequency 1 and input deviation 10^{-2}. In the legend, "Intv" means interval arithmetic, "Indp" means independence arithmetic, "Prec" means variance arithmetic, "Dev" means output uncertainty 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.

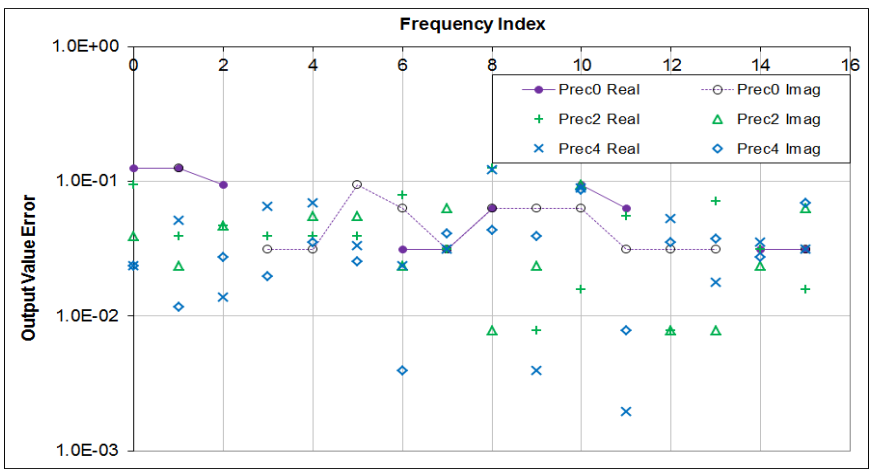

Figure 15: The output value errors of the forward FFT on a noisy sine signal of index frequency 1 and input deviation 10^{-2} using variance arithmetic with different bit inside uncertainty. In the legend, "Prec0" means variance arithmetic with 0-bit calculated inside uncertainty, "Prec2" means variance arithmetic with 2-bit calculated inside uncertainty, and "Prec4" means variance arithmetic with 4-bit calculated inside uncertainty.

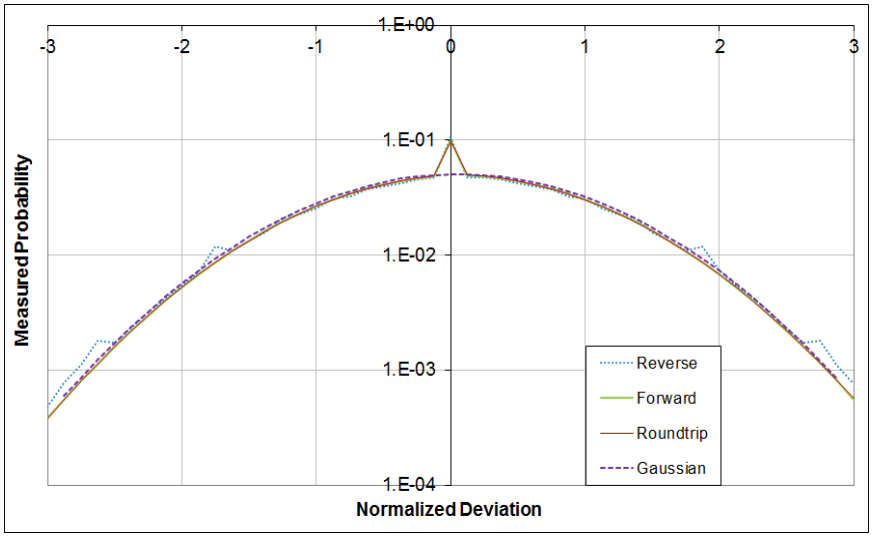

Figure 16: The measured tracking ratio distributions using independence arithmetic for FFT algorithms (as shown in legend). They are best fitted by a Gaussian distribution with the mean of 0.06 and deviation of 0.98.

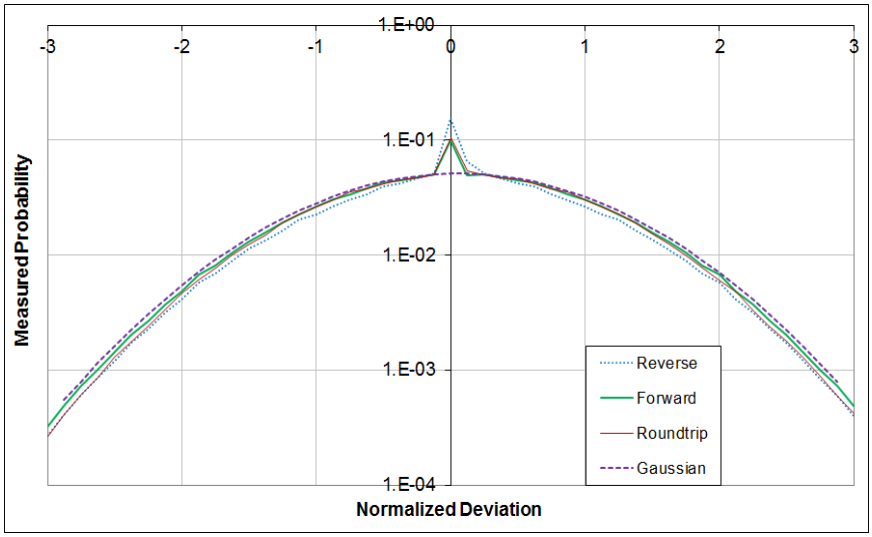

Figure 17: The measured tracking ratio distributions using variance arithmetic for FFT algorithms (as shown in legend). They are best fitted by a Gaussian distribution with the mean of 0.06 and deviation of 0.97.

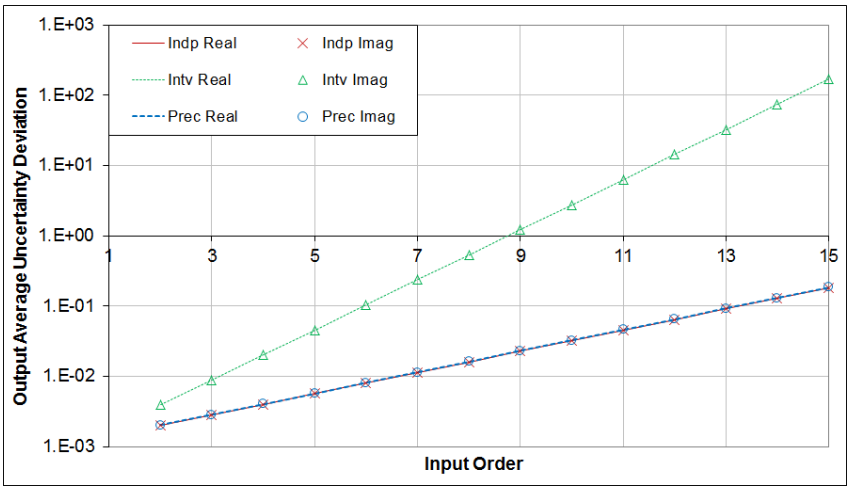

Figure 18: For the same input deviation of 10^{-3}, the empirical average output deviations of the forward FFT increase exponentially with the FFT order for all uncertainty-bearing arithmetics. In the legend, "Intv" means interval arithmetic, "Indp" means independence arithmetic, "Prec" means variance arithmetic, "Real" means real part, and "Imag" means imaginary part.

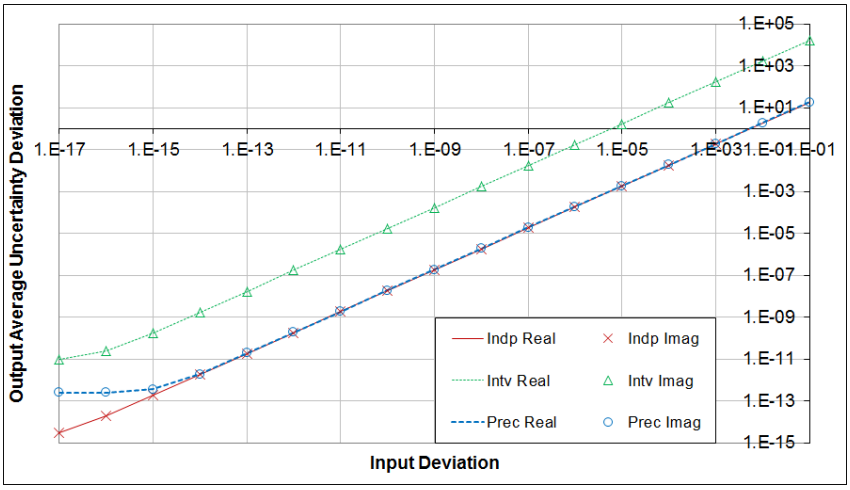

Figure 19: For the same order of the FFT calculation of 15, the empirical average output deviations of the forward FFT increases linearly with the input deviation for all uncertainty-bearing arithmetics. In the legend, "Intv" means interval arithmetic, "Indp" means independence arithmetic, "Prec" means variance arithmetic, "Real" means real part, and "Imag" means imaginary part.

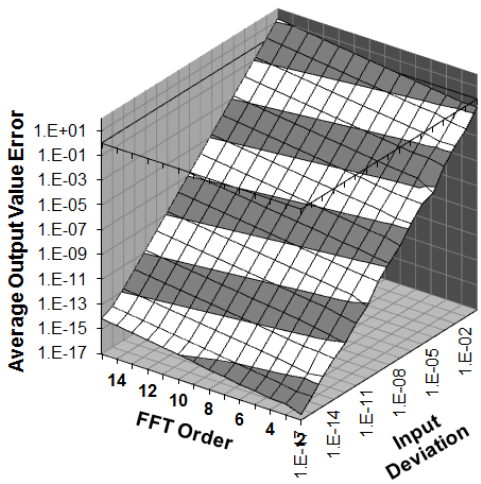

Figure 20: The empirical average output value errors using variance arithmetic increase exponentially with the FFT order and linearly with the input deviation, respectively.

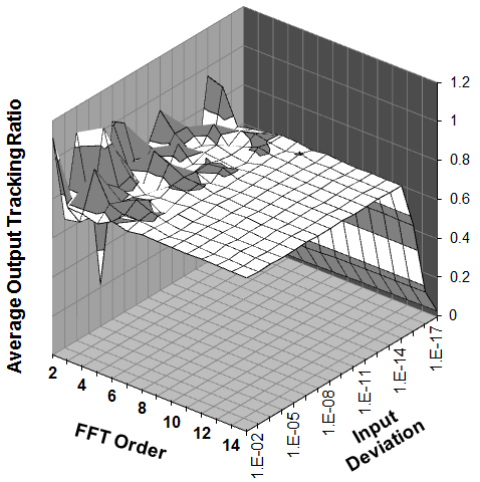

Figure 21: The empirical average output tracking ratios using variance arithmetic is a constant when the input deviation is larger than 10^{-14} and the FFT order is more than 5 for forward FFT algorithms. Because the precision of conventional floating-point representation is at 10^{-16}, adding Gaussian 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.

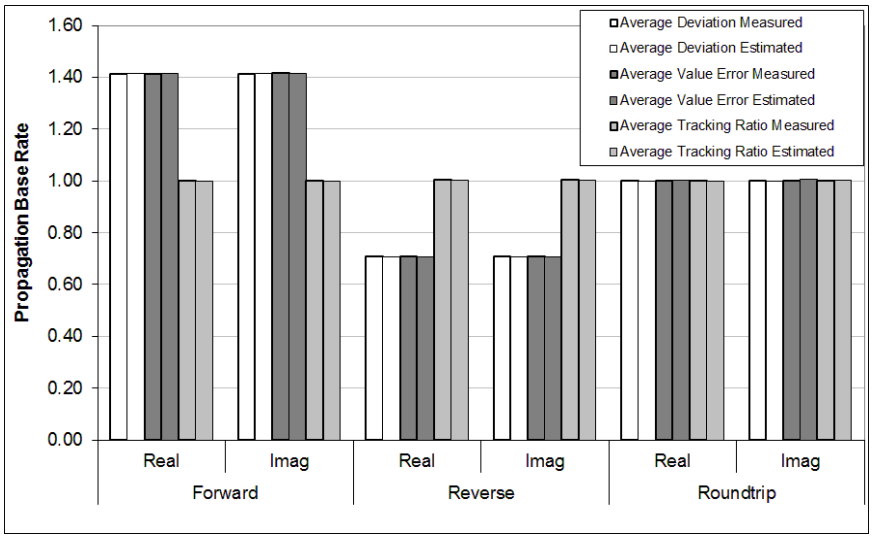

Figure 22: Empirical and theoretical β for fitting average output deviations, value errors and tracking ratios for forward, reverse and roundtrip FFT using independence arithmetic on noisy sine signals. In the chart, "Real" means real part, and "Imag" means imaginary part.

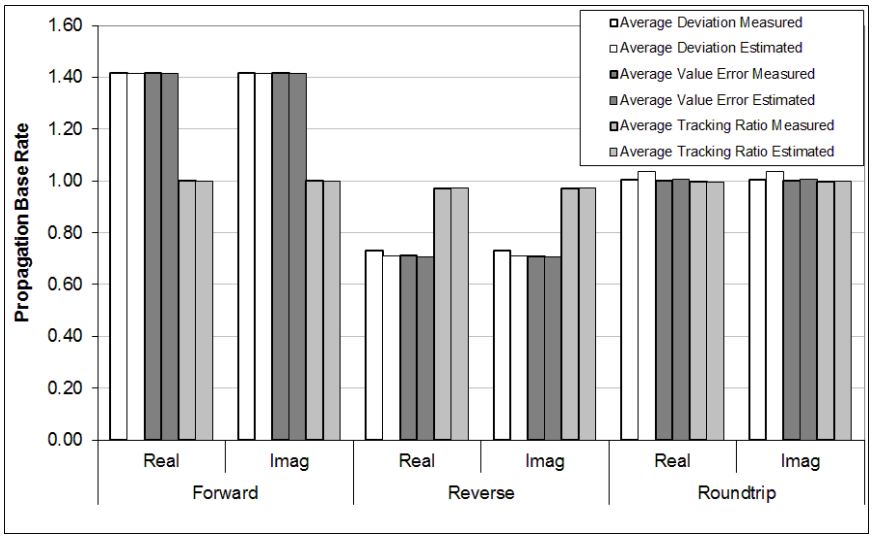

Figure 23: Empirical and theoretical β for fitting average output deviations, value errors and tracking ratios for forward, reverse and roundtrip FFT using variance arithmetic on noisy sine signals. In the chart, "Real" means real part, and "Imag" means imaginary part.

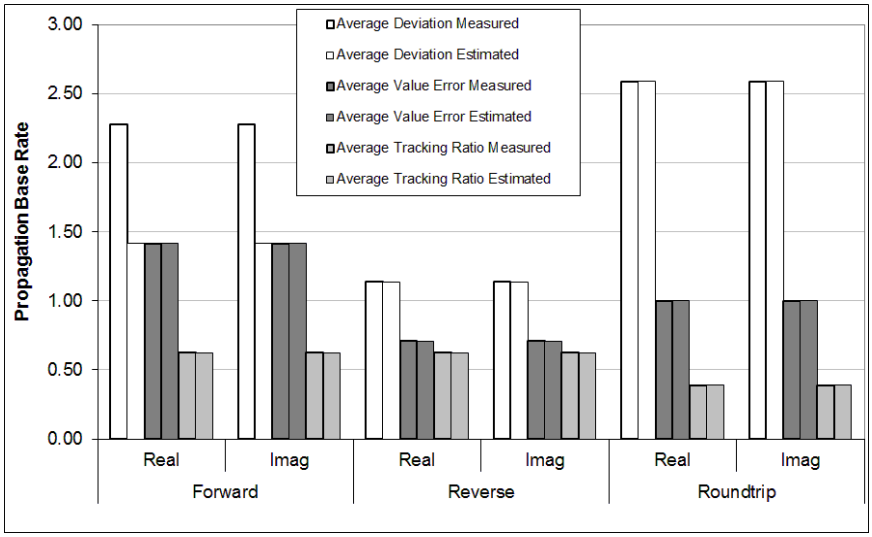

Figure 24: Empirical and theoretical β for fitting average output deviations, value errors and tracking ratios for forward, reverse and roundtrip FFT using interval arithmetic on noisy sine signals. In the chart, "Real" means real part, and "Imag" means imaginary part.

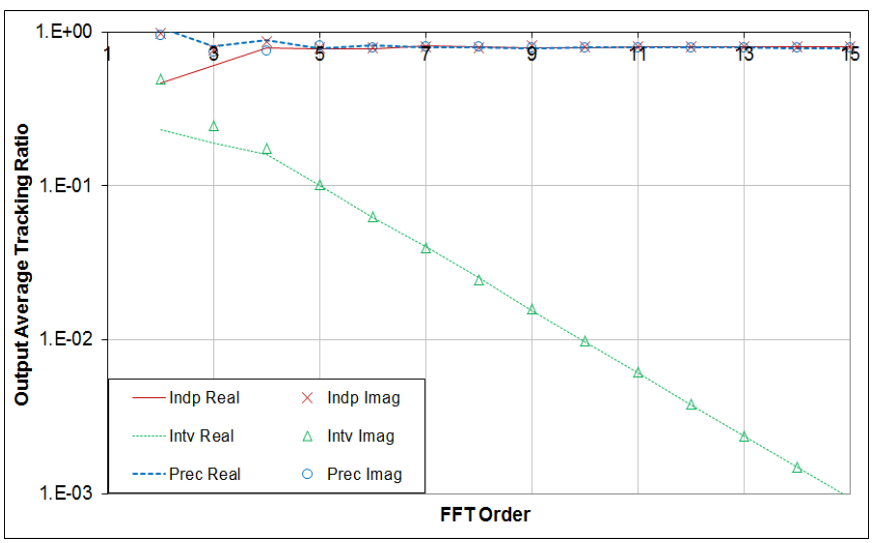

Figure 25: The empirical output average tracking ratios vs. the FFT order of the forward FFT for all three arithmetics when the input uncertainty deviation is 10^{-3}. In the legend, "Intv" means interval arithmetic, "Indp" means independence arithmetic, "Prec" means variance arithmetic, "Real" means real part, and "Imag" means imaginary part.

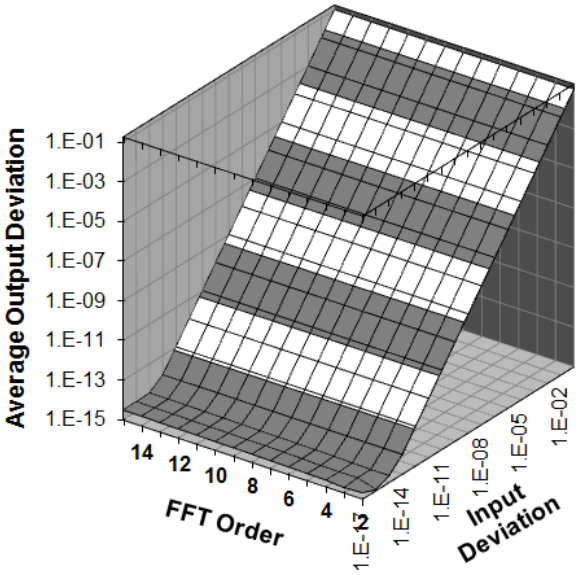

Figure 26: The empirical average output deviations vs. the FFT order and input deviations using variance arithmetic for the round-trip FFT algorithm.

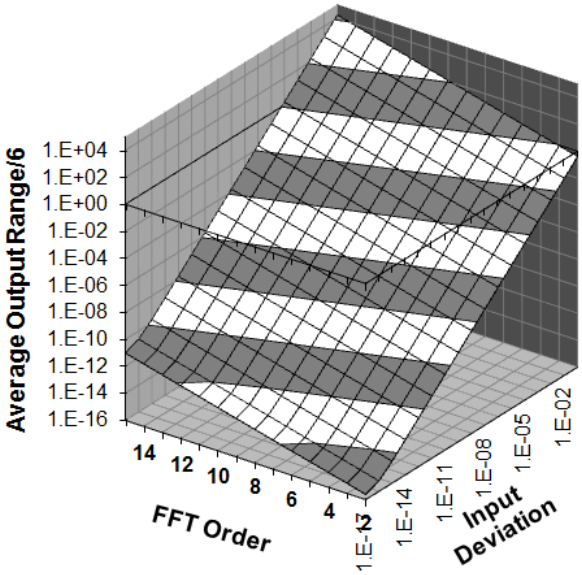

Figure 27: The empirical average output deviations vs. the FFT order and input deviations using interval arithmetic for the round-trip FFT algorithm.

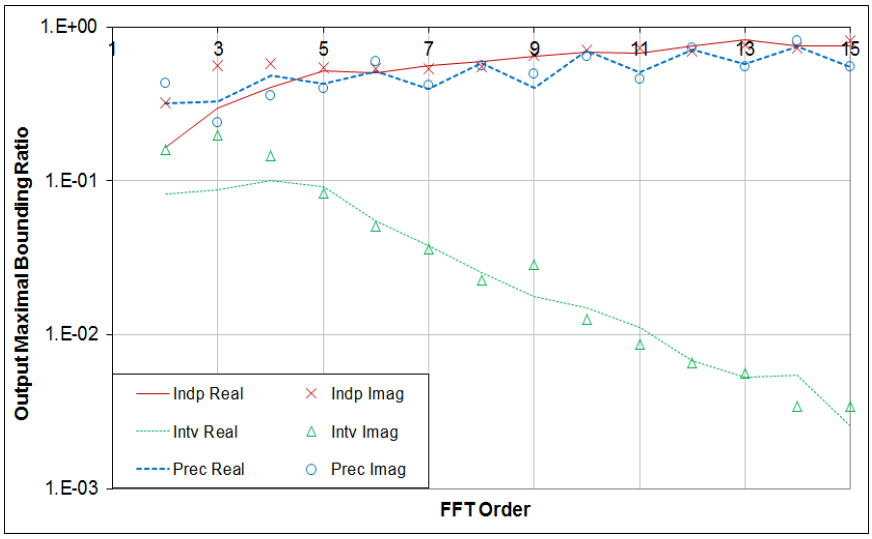

Figure 28: The empirical maximal output bounding ratios vs. the FFT order of the forward FFT for all three arithmetics. In the legend, "Intv" means interval arithmetic, "Indp" means independence arithmetic, "Prec" means variance arithmetic, "Real" means real part, and "Imag" means imaginary part.

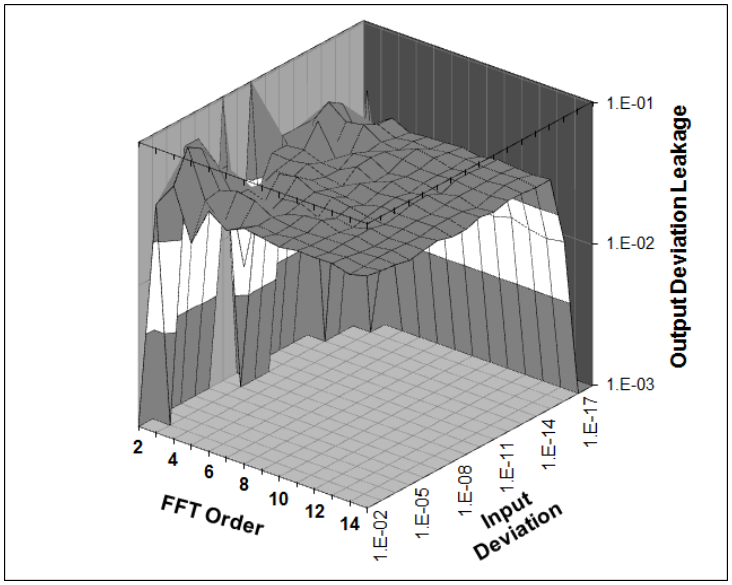

Figure 29: The empirical deviation leakages vs. the FFT order and input deviations using variance arithmetic for the forward FFT algorithm.

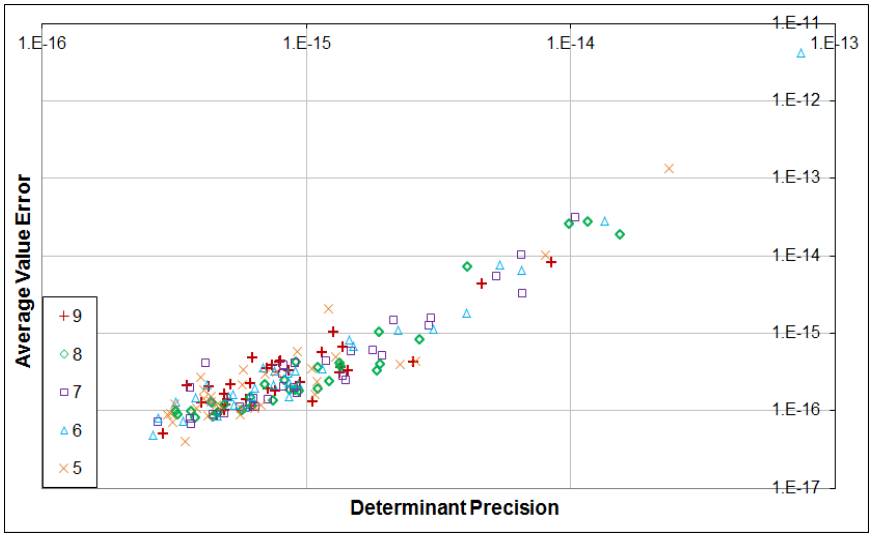

Figure 30: The empirical average value errors of the inverted matrix using conventional floating-point arithmetic vs. matrix determinant precision using variance arithmetic for clean matrices of different sizes (as shown in legend).

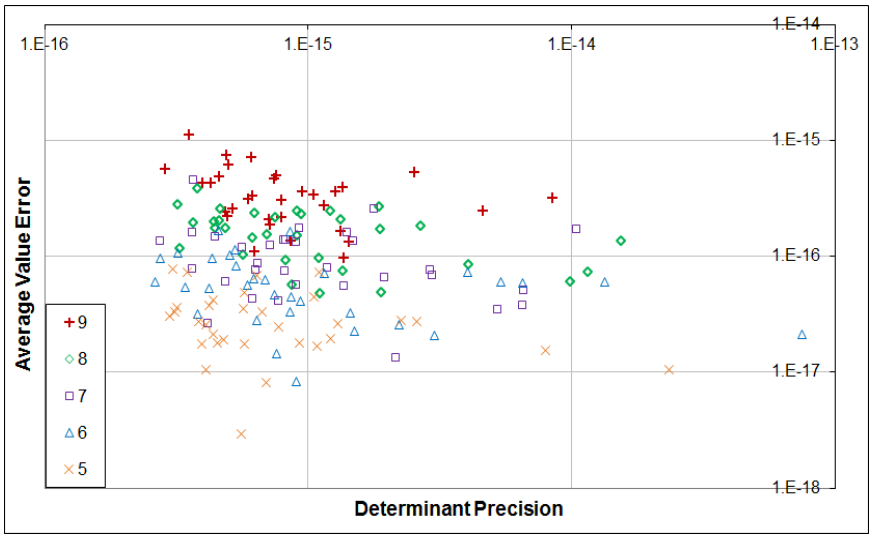

Figure 31: The empirical average value errors of the adjugate matrix using conventional floating-point arithmetic vs. matrix determinant precision using variance arithmetic for clean matrices of different sizes (as shown in legend).

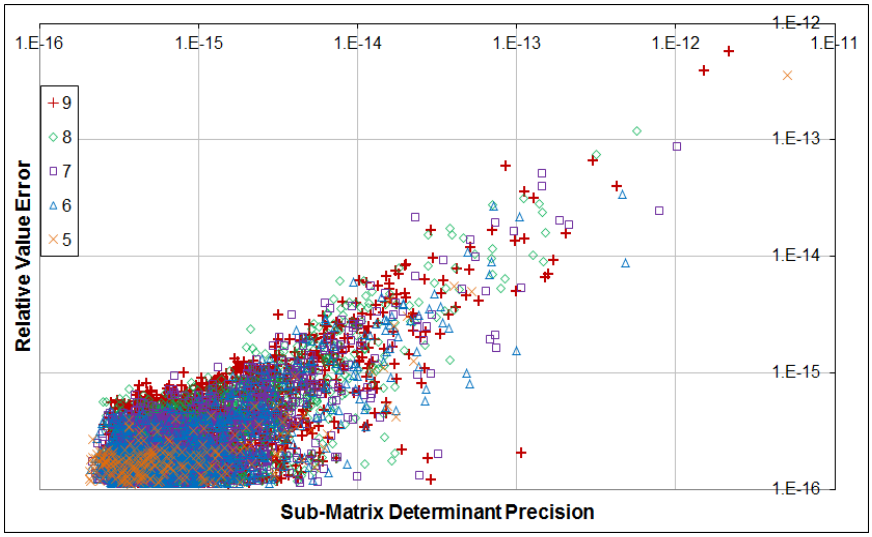

Figure 32: Empirical relative value errors of the adjugate matrix using conventional floating-point arithmetic vs. corresponding sub-matrix determinant precision using variance arithmetic for clean matrices of different sizes (as shown in legend).

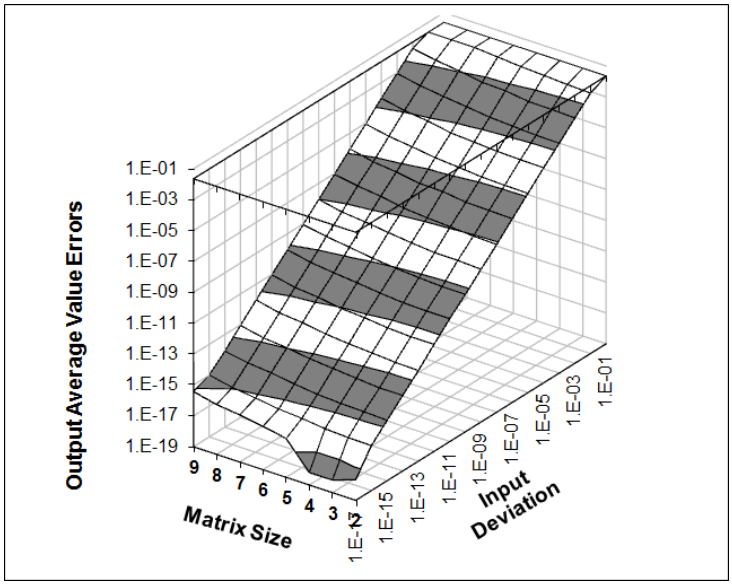

Figure 33: Using variance arithmetic, the average output deviations of the adjugate matrix vs. input precision and the matrix size.

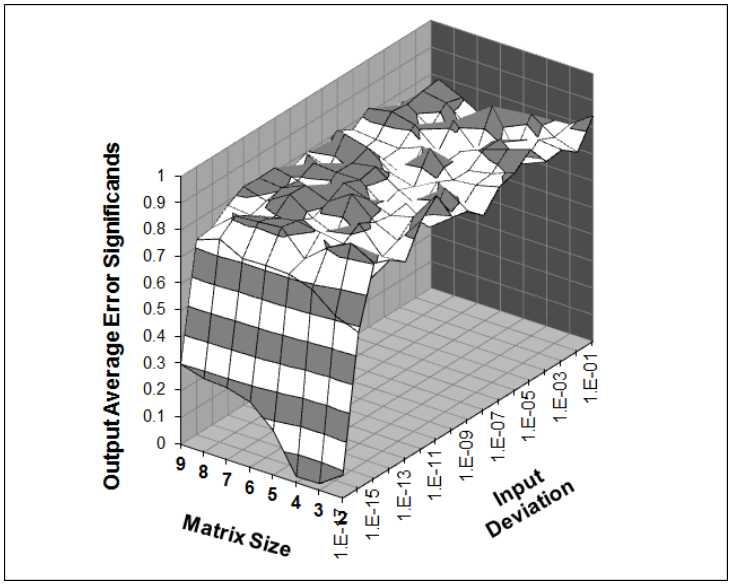

Figure 34: Using variance arithmetic, the average output tracking ratios of the adjugate matrix vs. input precision and the matrix size.

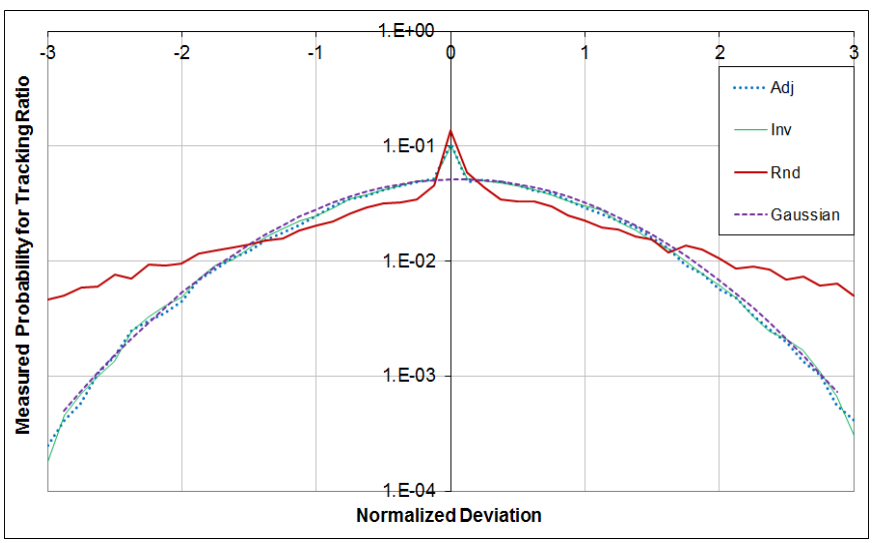

Figure 35: The measured tracking ratio distributions using variance arithmetic for matrix calculations of matrix size 9. They are best fitted by a Gaussian distribution with the mean of 0.06 and deviation of 0.96. In the legend, "Adj" means calculating adjugate M^A, "Inv" means calculating inverted M^{-1}, and "Rnd" means calculating double inverted $(M^{-1})^{-1}$.

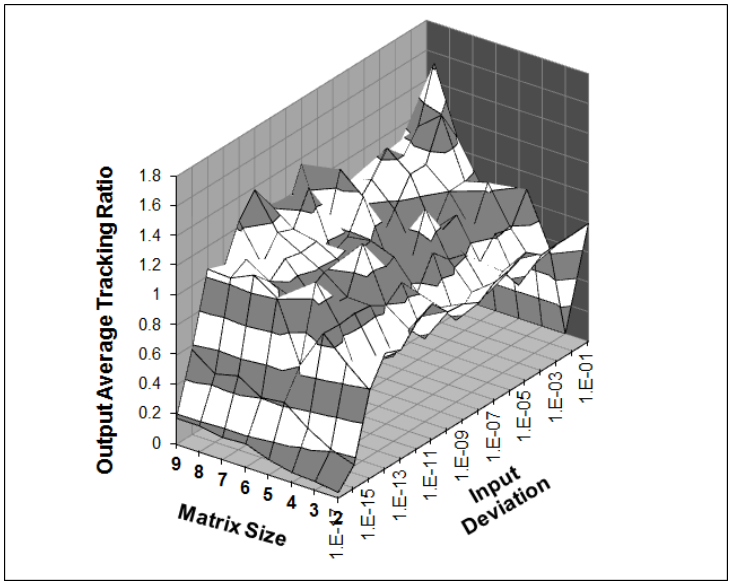

Figure 36: Using variance arithmetic, the average output tracking ratios of the double inverted matrix vs. input precision and the matrix size.

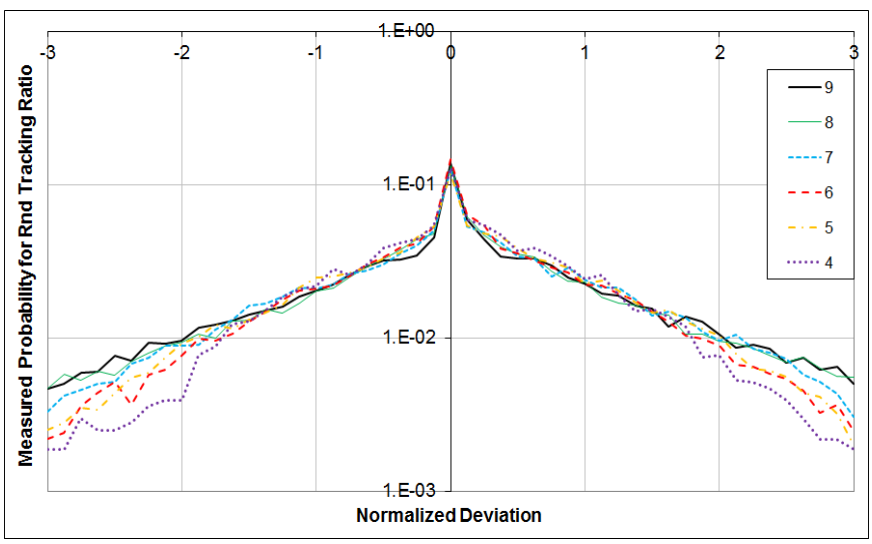

Figure 37: The measured tracking ratio distributions using variance arithmetic for matrix calculations of matrix size 9. They are best fitted by a Gaussian distribution with the mean of 0.06 and deviation of 0.96. In the legend, "Adj" means calculating adjugate M^A, "Inv" means calculating inverted M^{-1}, and "Rnd" means calculating double inverted $(M^{-1})^{-1}$.

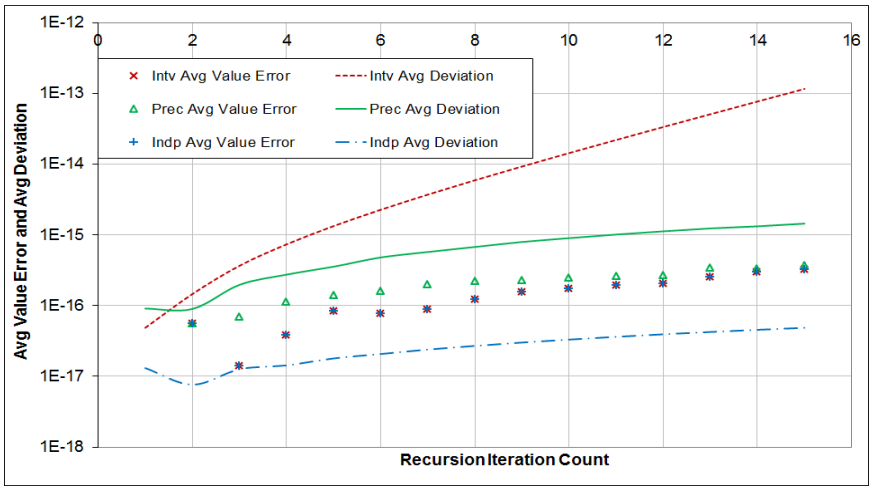

Figure 38: The empirical output average value errors and corresponding average output deviations vs. the recursion iteration count of the regressive calculation of sine values using interval arithmetic, variance arithmetic and independent arithmetic. The x-axis indicates the recursion iteration count 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 variance arithmetic.

Figure 39: 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 variance arithmetic. In the legend, "Intv" means interval arithmetic, "Indp" means independence arithmetic, and "Prec" means variance arithmetic.

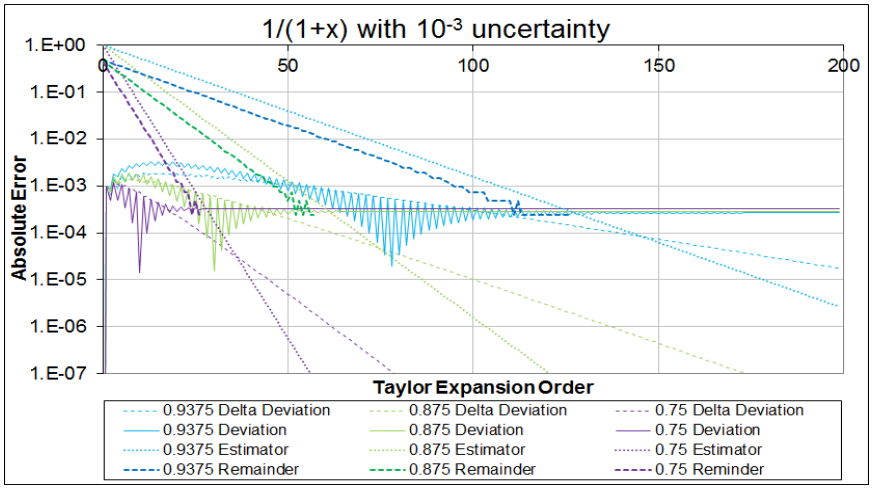

Figure 40: The delta deviation, deviations, Cauchy estimator and remainders of a Taylor expansion vs. the expansion orders for different input value with 10^{-3} input uncertainty using variance arithmetic with 0-bit calculated inside uncertainty. Different inputs are displayed using different color.

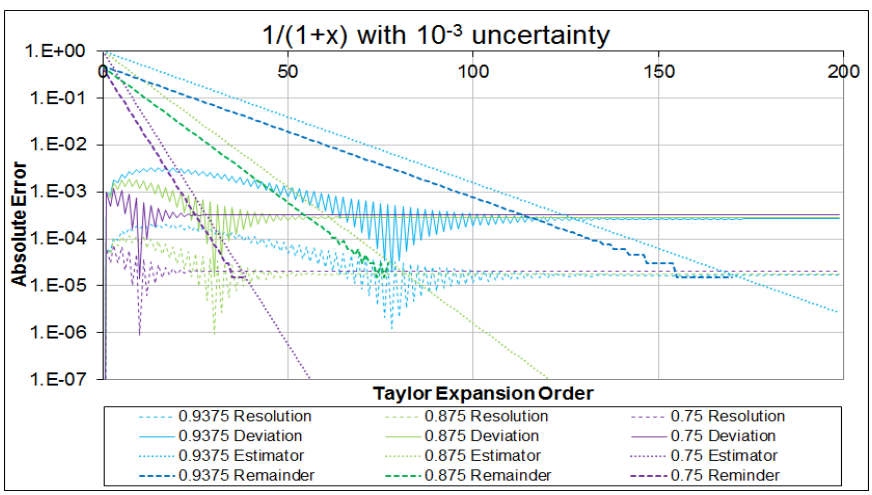

Figure 41: The deviations, resolutions, Cauchy estimator and remainders of a Taylor expansion vs. the expansion orders for different input value with 10^{-3} input uncertainty using variance arithmetic with 4-bit calculated inside uncertainty. Different inputs are displayed using different color.

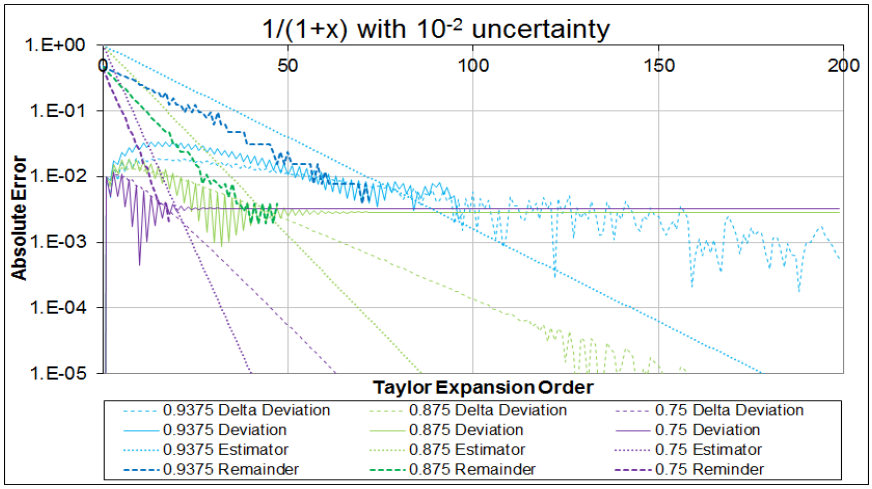

Figure 42: The delta deviation, deviations, Cauchy estimator and remainders of a Taylor expansion vs. the expansion orders for different input value with 10^{-2} input uncertainty using variance arithmetic with 0-bit calculated inside uncertainty. Different inputs are displayed using different color.

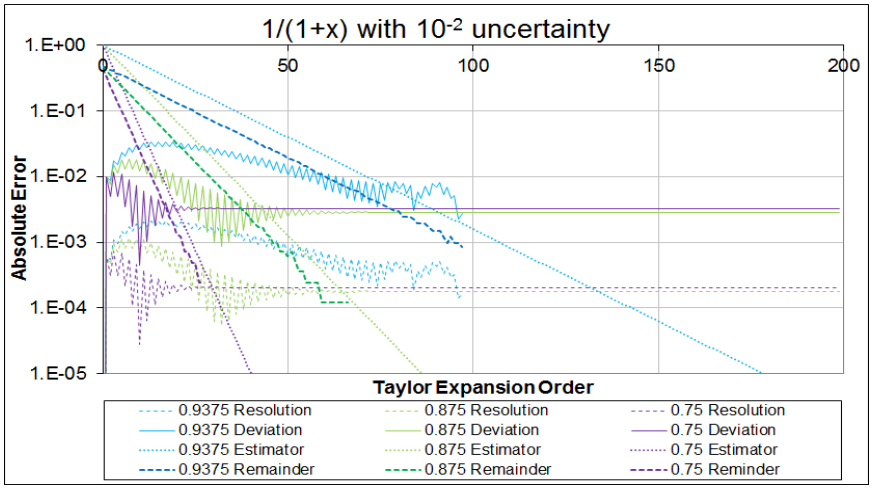

Figure 43: The deviations, resolutions, Cauchy estimator and remainders of a Taylor expansion vs. the expansion orders for different input value with 10^{-2} input uncertainty using variance arithmetic with 4-bit calculated inside uncertainty. Different inputs are displayed using different color.

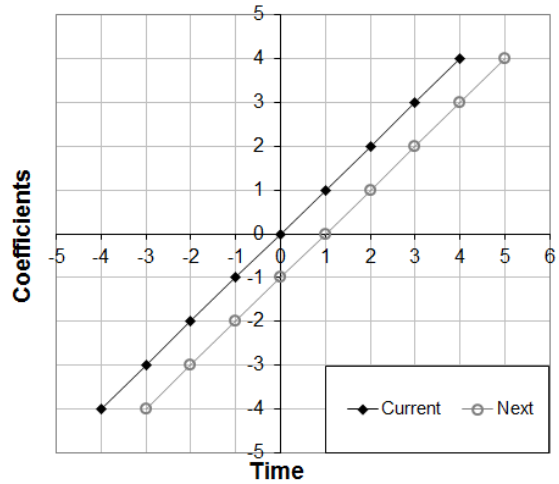

Figure 44: Coefficients of X in (10.2) at current and next position in a time series of the least square linear regression. Except the two end points at $X = -H$ and $X = H + 1$, respectively, the coefficient difference between the current and then next position in a time series are all by 1 in the overlapping region from $X = -H + 1$ to $X = H$, which results in (10.3).

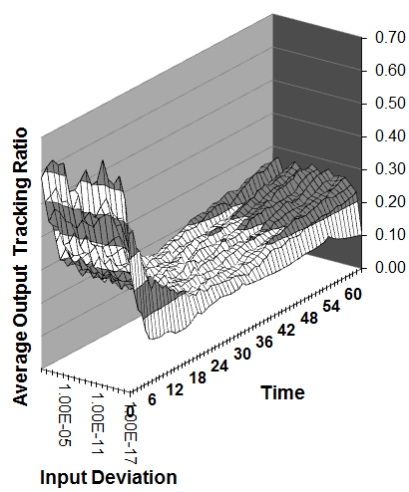

Figure 45: The average tracking ratio vs. time and the input uncertainty deviations for the progressive moving-window linear regression of a straight line using variance arithmetic with 4-bit calculated inside uncertainty.

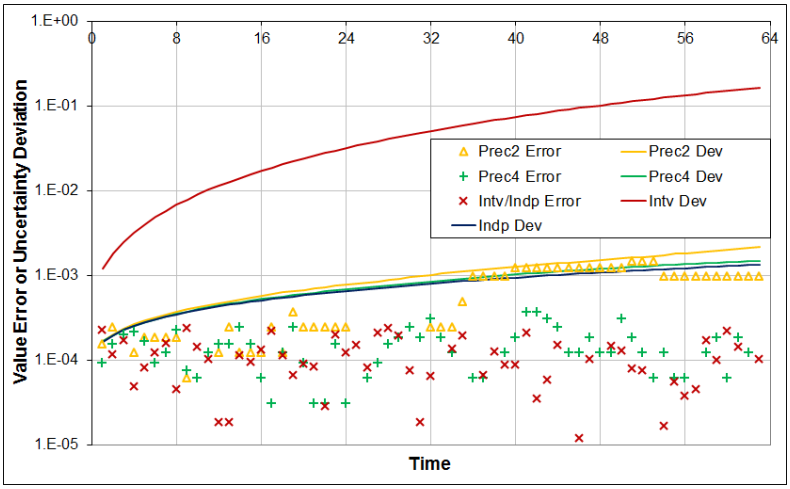

Figure 46: The output uncertainty deviations and the value errors vs. time for the progressive moving-window linear regression of a straight line. In the legend, "Indp" means independent arithmetic, "Intv" means interval arithmetic, "Prec4" and "Prec2" means the variance arithmetic with 4-bit and 2-bit calculated inside uncertainty, respectively.

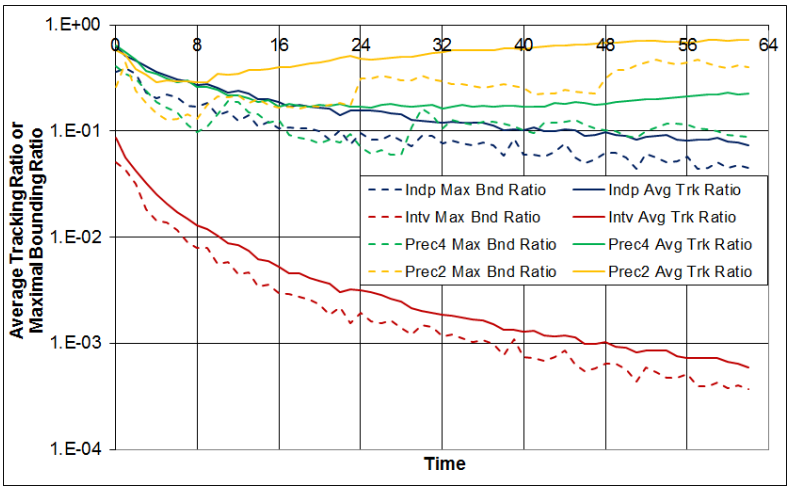

Figure 47: The average tracking ratios and the max bounding ratios vs. time for the progressive moving-window linear regression of a straight line. In the legend, "Indp" means independence arithmetic, "Intv" means interval arithmetic, "Prec4" and "Prec2" means the variance arithmetic with 4-bit and 2-bit 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.

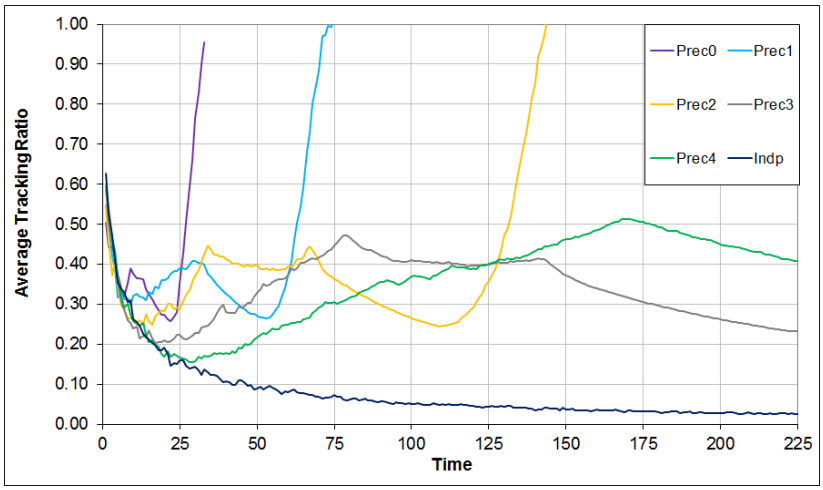

Figure 48: The average tracking ratios vs. time and the bits calculated inside uncertainty using variance arithmetic for the progressive moving-window linear regression of a straight line. In the legend, "Indp" means independence arithmetic, "PrecX" means the variance arithmetic with X-bit calculated inside uncertainty.

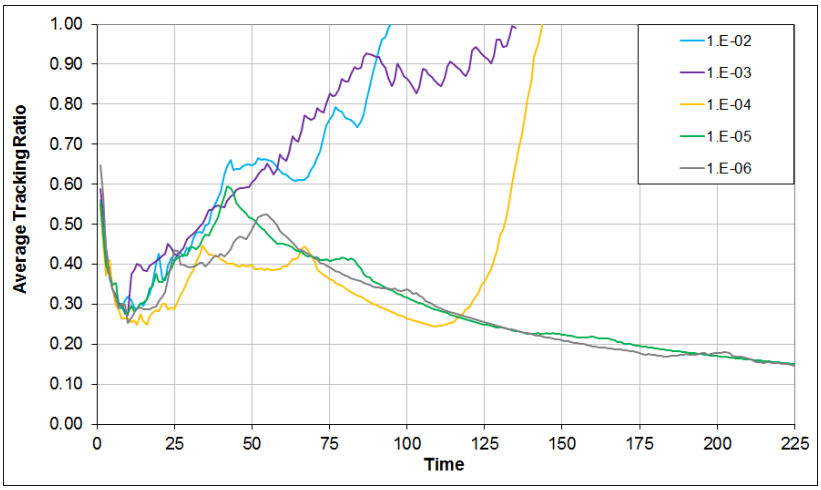

Figure 49: The average tracking ratios vs. time and the input precision using variance arithmetic with 2-bit calculated inside uncertainty for the progressive moving-window linear regression of a straight line for different input uncertainty deviations.

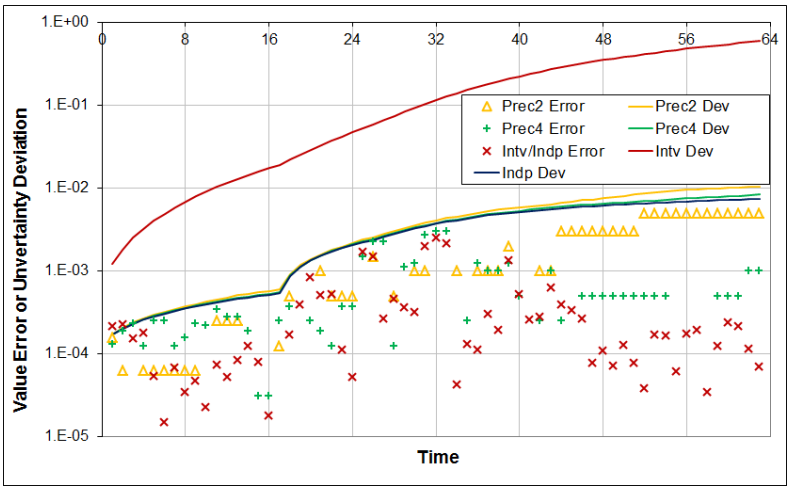

Figure 50: The output deviations and the value errors vs. time for the progressive moving-window linear regression of a straight line with 10-fold increase of both input noise and input uncertainty in the middle.

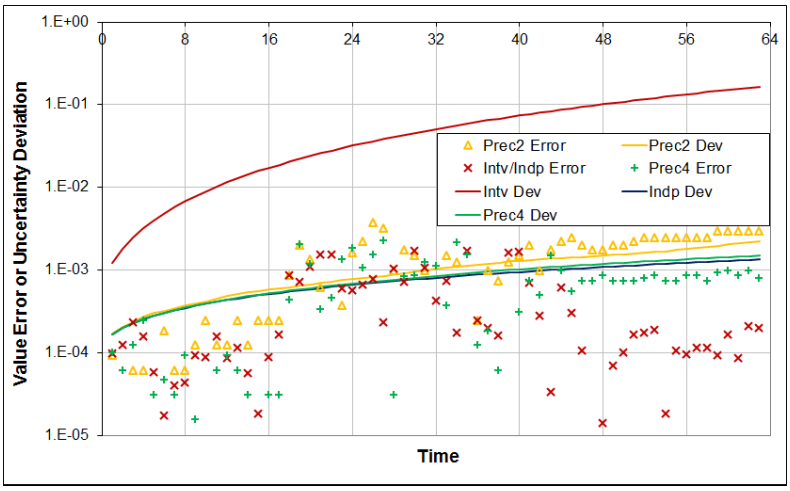

Figure 51: The output deviations and the value errors vs. time for the progressive moving-window linear regression of a straight line with only 10-fold increase of both input noise in the middle.

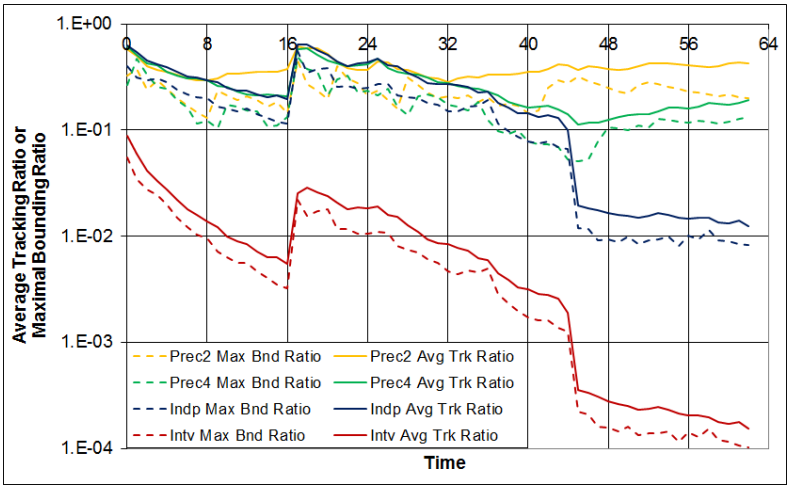

Figure 52: The average tracking ratios and the max bounding ratio vs. time for the progressive moving-window linear regression of a straight line with 10-fold larger input noise and deviation in the middle, to simulate larger noise following the straight line.

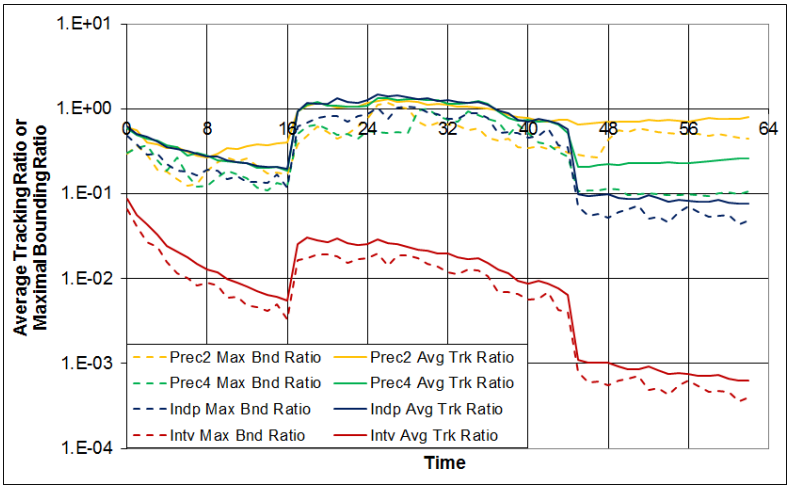

Figure 53: The average tracking ratios and the max bounding ratio vs. time for the progressive moving-window linear regression of a straight line with 10-fold larger input noise but same input deviation in middle, to simulate defects in obtaining the corresponding uncertainty deviations.

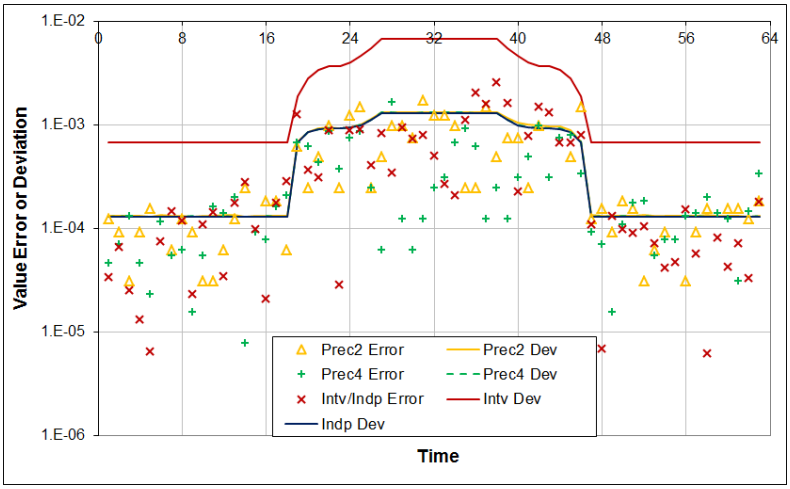

Figure 54: The output deviations and the value errors vs. time for the expressive moving-window linear regression of a straight line with 10-fold increase of both input noise and input uncertainty in the middle using variance arithmetic with 4-bit calculated inside uncertainty.

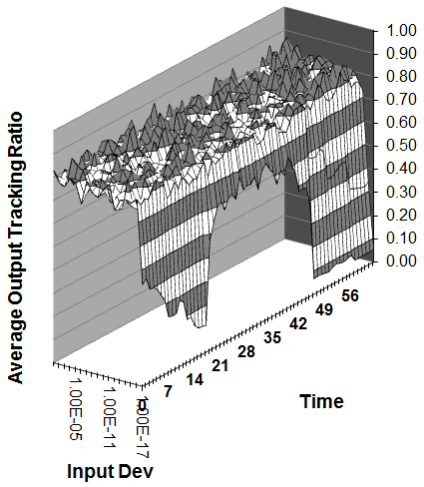

Figure 55: The average tracking ratio vs. time and the input uncertainty deviations for the expressive moving-window linear regression of a straight line with 10-fold increase of both input noise and input uncertainty in the middle using variance arithmetic with 4-bit calculated inside uncertainty.

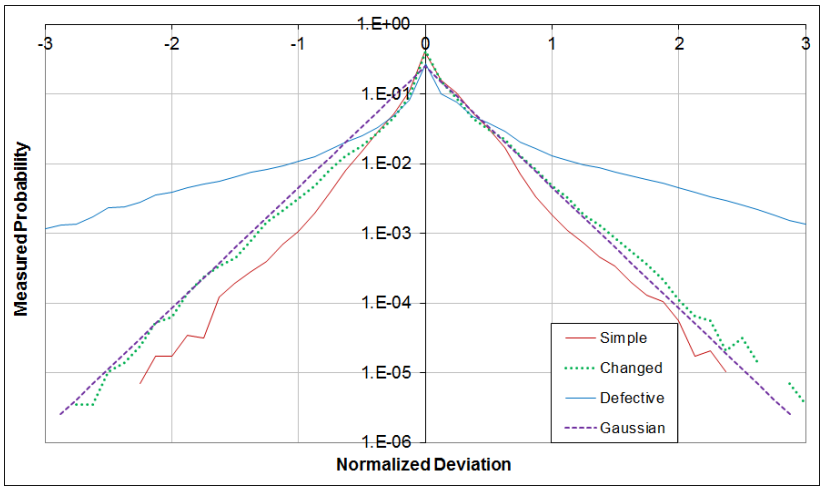

Figure 56: The measured tracking ratio distributions of the progressive moving-window linear regression for different cases (as shown in legend) using variance arithmetic with 4-bit calculated inside uncertainty. The case of "Changed" is best fitted by a exponential distribution with the mean of 0 and deviation of 0.25.

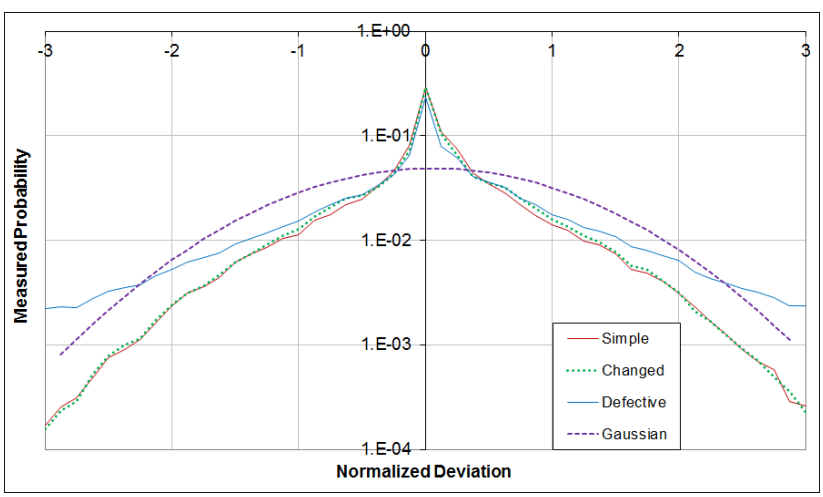

Figure 57: The measured tracking ratio distributions of the expressive moving-window linear regression using Formula (10.3) for different cases (as shown in legend) using variance arithmetic with 4-bit calculated inside uncertainty. The cases of "Simple" and "Changed" are best fitted by a Gaussian distribution with the mean of 0.06 and deviation of 0.97.