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.

As shown previously [37], interval arithmetic grossly exaggerates calculation uncertainty, e.g., for $f^{-1}(f(x))$ the result uncertainty is much larger than the original uncertainty. On the other hand, as shown later in this paper, the variance arithmetic provides stable bounding for a given bounding leakage, e.g., $\pm$3.1-fold of the result deviation from the mean for a bounding leakage of 2×10^{-7}.

[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.

One problem of significance arithmetic is that itself can not properly specifies the uncertainty [37]. For example, if the least significand bit of significand is used to specify uncertainty, then the representation have very coarse precision, such as $1 \pm 10^{-3} = 1024 \times 2^{-10}$ [37]. Introducing limited bits calculated inside uncertainty can not avoid this problem completely. Thus, the resolution of the conventional floating-point representation is desired. This is the reason why variance arithmetic abandoned the significance arithmetic nature of its predecessor [37].

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 of analytic functions, as described in Section 2. Compare to its predecessor [37]:

- The introduction section 1 remains almost unchanged, except it quotes the negative results from [37] for interval arithmetic and significance arithmetic.
- The theoretical foundation section 2 has the following significant differences:
    1. Both have identical uncorrelated uncertainty assumption.
    2. The variance arithmetic has different digital presentation.
    3. The variance arithmetic has similar Taylor expansion formulas, but the variance is centered around statistical means instead of the calculation results assuming no uncertainty, so variance arithmetic agrees with statistics strictly.
    4. The theoretical convergence and statistical binding leakage of the result uncertainties of the variance arithmetic are provided.
    5. The variance arithmetic has no dependency problem for analytic questions. Instead, the variance arithmetic uses statistics to trace dependency.
- Section 3 remains largely unchanged, except major improvement on the descriptions of validation methods and standards.
- Section 4 is new, to provide a statistical test of variance arithmetic in using the common math library functions.
- Section 6 reproduces the previous result for the input data with added noises. It added a new signal for test. It also evaluates the calculation errors of the library sin and tan functions.

2 The Variance Arithmetic

2.1 Foundation for the Variance Arithmetic

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.

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.

Taylor expansion can be used to calculate the result variance of analytic functions.

The uncorrelated uncertainty assumption only applies to imprecise inputs. When the inputs obeys the *uncorrelated uncertainty assumption*, in the intermediate steps, the variance arithmetic uses statistics to account for the correlation between uncertainties through their associated variables. Thus, the variance arithmetic has no dependency problem.

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. 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.

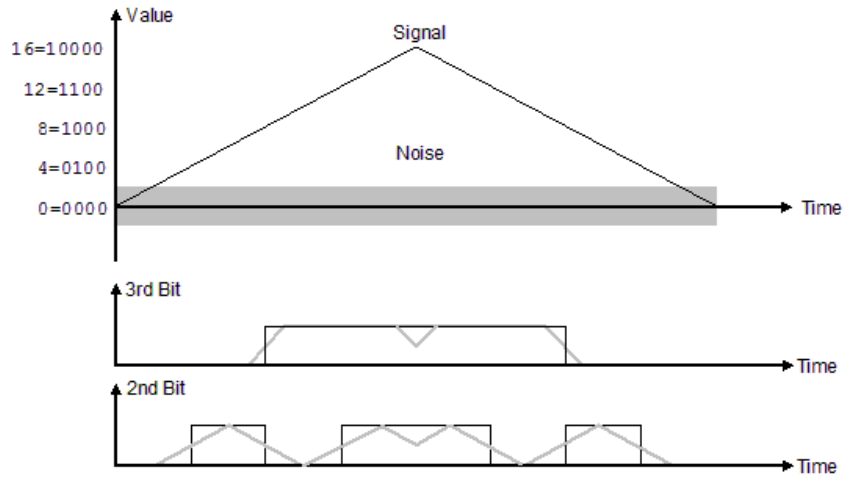

Figure 1: Effect of noise on bit values of a measured value. The triangular wave signal and the added white noise are shown at top using the thin black line and the grey area, respectively. The values are measured by a theoretical 4-bit Digital-to-Analog Converter in ideal condition, assuming LSB is 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. This figure has been published previously [37].

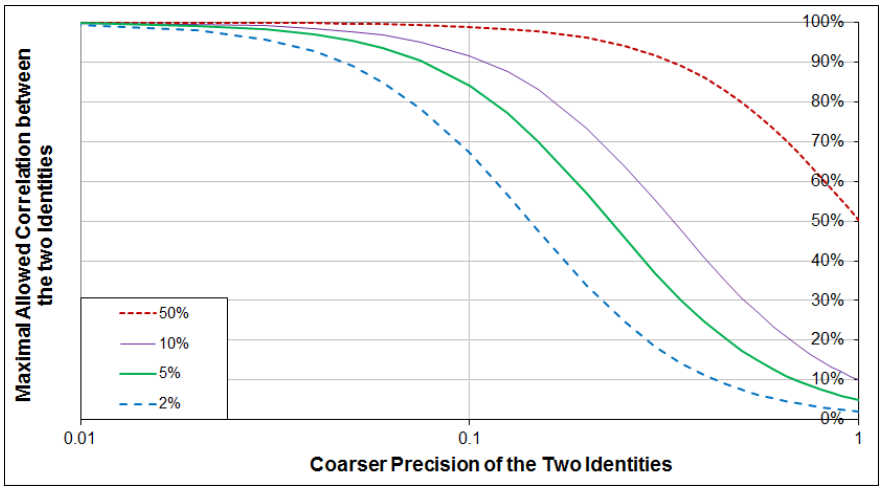

Figure 2: Allowed maximal correlation between two values vs. input precisions and independence standard (as shown in legend) for the independence uncertainty assumption of variance arithmetic to be true. This figure has been published previously [37].

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

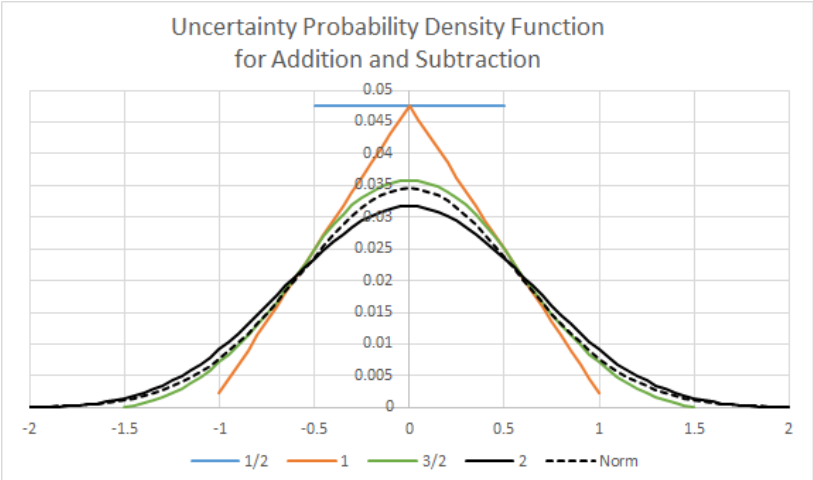

Figure 3: The probability distribution $P_{\frac{n}{2}}(x)$ of Formula (2.6), assuming $P_{\frac{1}{2}}(x)$ is uniformly distributed between $(-1/2, +1/2)$. The legend shows n. $P_2(x)$ is already close to the corresponding Gaussian distribution with the same mean and deviation, which is plotted in dash line of the same color. This figure has been published previously [37].

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 Result Uncertainty For Pure Rounding [37]

Empirically, the pure rounding error due to round-off is shown to be uniformly distributed between the half bit of the least significant bit of the siginficand [37], whose probability density function is shown in Formula (2.5):

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

2.4 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].

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

[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..

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:

- $P_{n/2}(x)$ converges in distribution to a Gaussian distribution.
- 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 (2.7) give the result variance of addition and subtraction surrounding $x \pm y$. Formula (2.8) 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 (2.8) can be normalized as Formula (2.9).

$$\delta^2(x \pm y) = (\delta x)^2 + (\delta y)^2; \tag{2.7}$$

$$\rho(\tilde{x}, x, \delta x) = \frac{1}{\delta x} N(\frac{\tilde{x} - x}{\delta x}); \tag{2.8}$$

$$\tilde{z} \equiv \frac{\tilde{x} - x}{\delta x} : \quad \rho(\tilde{x}, x, \delta x) d\tilde{x} = N(\tilde{z}) d\tilde{z}; \tag{2.9}$$

Formula (2.7) 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.

2.5 A Variance Representation

A variance arithmetic representation uses a pair of conventional floating-point numbers to represent the value and the variance of an imprecise value. The square root of a variance is defined as an *uncertainty*. When the uncertainty is not specified:

- A conventional floating-point value has $1/\sqrt{3}$-fold of its least significant value as its uncertainty, in which the *least significant value* is defined as the equivalent value of the least significant bit of the significand of the conventional floating-point value, e.g., the least significant value of 1.0 is 2^{-52}. Essentially, the floating value is regarded as containing a rounding error uniformly distributed within the least significant bit of its significand.
- An integer value within the range of $(-2^{53}, +2^{53})$ has 0 as its uncertainty.
- An integer value outside the range of $(-2^{53}, +2^{53})$ is converted to a conventional floating-point value.

2.6 Comparison

Two imprecise values can be compared statistically using their difference.

When the value difference is zero, the two imprecise values are equal. Such definition of equality deviates from statistics. In statistics, such two imprecise value has 50% possibility to be less than or greater to each other but no chance to be equal to each other.

Otherwise, the standard z-method [4] is used to judge whether they are equal to each other, or less or more than each other. For example, the difference between 1.002 ± 0.001 and 1.000 ± 0.002 is 0.002 ± 0.00224, so that the z-value for the difference is $z = 0.002/0.00224$, and the probability for them to be not equal is $\xi(|z|) - \xi(-|z|) = 62.8\%$, in which $\xi(z)$ is the cumulative density function for normal distribution [4]. If the threshold probability of not equality is 50%, $1.000 \pm 0.002 < 1.002 \pm 0.001$. Instead of using the threshold probability, the bounding range for z can be used, such as $|z| \leq 0.67448975$ for the equivalent threshold probability of 50%.

2.7 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 (2.10), which leads to Formula (2.11) [37] and (2.12) [37] for the result variance and precision surrounding the result xy, respectively.

$$(x \pm \delta x) \times (y \pm \delta y) = (x + (0 \pm \delta x)) \times (y + (0 \pm \delta y)); \tag{2.10}$$

$$\delta^2(xy) = x^2(\delta^2 y) + y^2(\delta^2 x) + (\delta^2 x)(\delta^2 y); \tag{2.11}$$

$$xy \neq 0 : P(xy)^2 = P(x)^2 + P(y)^2 + P(x)^2 P(y)^2; \tag{2.12}$$

Formula (2.11) is identical to Formula (1.12) for statistical propagation of uncertainty except their cross term, representing difference in their statistical requirements, respectively. Formula (2.11) 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].

2.8 Uncertainty Distribution

Let $\tilde{y} = f(\tilde{x})$ be a strictly monotonic function, so that $\tilde{x} = f^{-1}(\tilde{y})$ exist. Formula (2.13) is the uncertainty distribution density function [1]. In Formula (2.13) [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 (2.13), Formula (2.14), (2.15), and (2.16) give the $\rho(\tilde{y}, y, \delta y)$ for e^x, $\ln(x)$, and x^c, respectively.

$$N(\tilde{z})d\tilde{z} = \rho(\tilde{x}, x, \delta x)d\tilde{x} = \rho(f^{-1}(\tilde{y}), x, \delta x)\frac{d\tilde{x}}{d\tilde{y}}d\tilde{y} = \rho(\tilde{y}, y, \delta y)d\tilde{y}; \tag{2.13}$$

$$y = e^x : \rho(\tilde{y}, y, \delta y) = \frac{1}{\tilde{y}}\frac{1}{\delta x}N(\frac{\log(\tilde{y}) - x}{\delta x}); \tag{2.14}$$

$$y = \ln(x) : \rho(\tilde{y}, y, \delta y) = e^{\tilde{y}}\frac{1}{\delta x}N(\frac{e^{\tilde{y}} - x}{\delta x}); \tag{2.15}$$

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

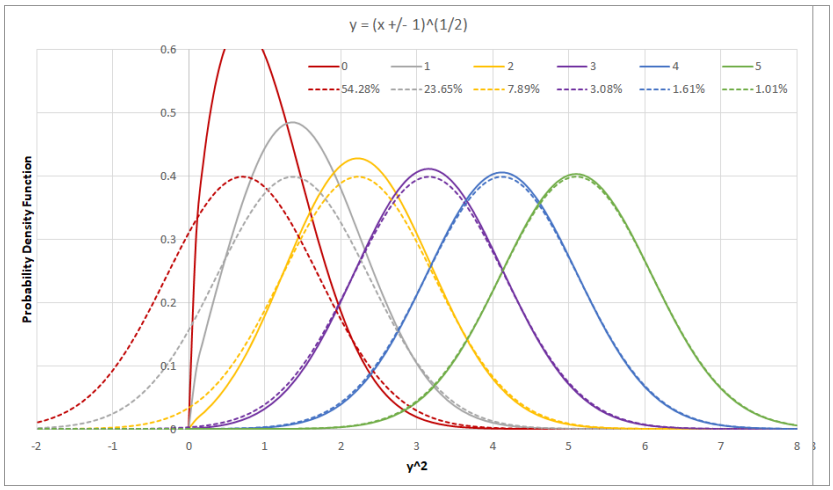

Figure 4: The probability density function for $\tilde{y} = \sqrt{x \pm 1}$, for different x as shown in the legend. The x-axis is scaled as $\tilde{y}^2$. Each probability density function $\rho(\tilde{x})$ in solid line is compared with a Gaussian distribution $\varrho(\tilde{x})$ of the same mode 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.

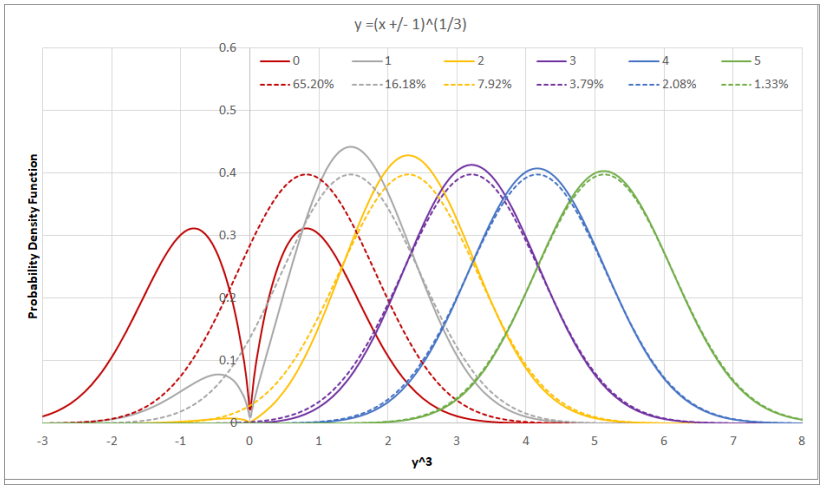

Figure 5: The probability density function for $\sqrt[3]{x \pm 1}$, for different x as shown in the legend. The y-axis is scaled as $\tilde{y}^3$. Each probability density function $\rho(\tilde{x})$ in solid line is compared with a Gaussian distribution $\varrho(\tilde{x})$ of the same mode and the same deviation in the dash line of the same color. The legend for the Gaussian distribution shows the value of Gaussian difference in percentage.

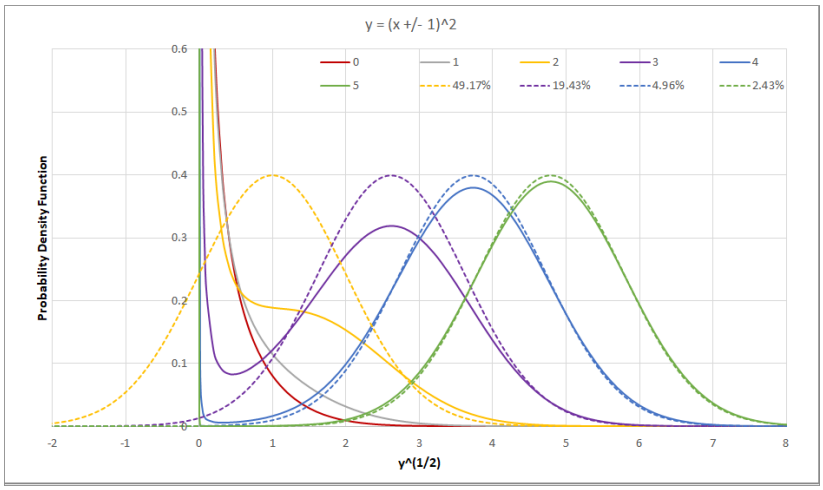

Figure 6: The probability density function for $(x \pm 1)^2$, for different x as shown in the legend. The y-axis is scaled as $\sqrt{\tilde{y}}$. Some probability density function $\rho(\tilde{x})$ in solid line is compared with a Gaussian distribution $\varrho(\tilde{x})$ of the same mode and the same deviation in the dash line of the same color. The legend for the Gaussian distribution shows the value of Gaussian difference in percentage.

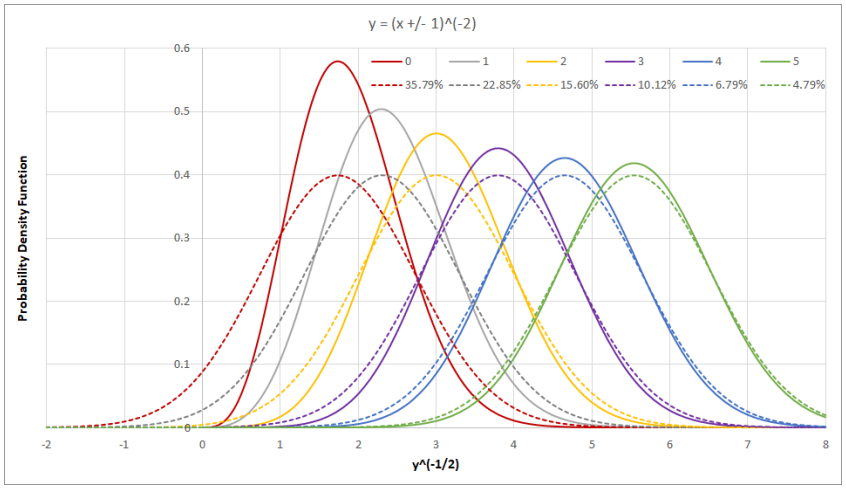

Figure 7: The probability density function for $1/(x \pm 1)^2$, for different x as shown in the legend. The y-axis is scaled as $1/\sqrt{\tilde{y}}$. The probability density function $\rho(\tilde{x})$ in solid line is compared with a Gaussian distribution $\varrho(\tilde{x})$ of the same mode and the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

Using Formula (2.13), Formula (2.17) 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 (2.18), (2.19), and (2.20) respectively.

$$\tilde{z} = \frac{f^{-1}(\tilde{y}) - x}{\delta x} : \quad \frac{d^2\tilde{z}}{d\tilde{y}^2} = \frac{d\tilde{z}}{d\tilde{y}}\tilde{z}; \tag{2.17}$$

$$\tilde{y} = e^{\tilde{x}} : \quad \ln(\tilde{y}_m) = x - (\delta x)^2; \tag{2.18}$$

$$\tilde{y} = \ln(\tilde{x}) : \quad e^{\tilde{y}_m} = \frac{x + \sqrt{x^2 + 4(\delta x)^2}}{2}; \tag{2.19}$$

$$\tilde{y} = \tilde{x}^c : \quad (\tilde{y}_m)^{1/c} = \frac{x + \sqrt{x^2 - 4(c-1)(\delta x)^2}}{2}; \tag{2.20}$$

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 $f_x^{(1)} = \infty \to \rho(\tilde{y}, y, \delta y) = 0$, while a *pole* happens when $f_x^{(1)} = 0 \to \rho(\tilde{y}, y, \delta y) = \infty$. 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 (2.13).

Figure 4 shows the probability density function for $\sqrt{x \pm 1}$ according to Formula (2.16), 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 shows the probability density function for $\sqrt[3]{x \pm 1}$ according to Formula (2.16), 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 6 shows the probability density function for $(x \pm 1)^2$ according to Formula (2.16), 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 (2.20), $(0 \pm 1)^2$ and $(1 \pm 1)^2$ has no mode. $(0 \pm 1)^2$ is actually the χ^2 distribution [1].

Figure 7 shows the probability density function for $1/(x \pm 1)^2$ according to Formula (2.16), which has a zero at $x = 0$. Figure 7 looks more like Figure 4 than Figure 6, showing that pole or zero determines the properties of uncertainty distributions.

In all the figures, the probability density function in the $\tilde{z}$ representation becomes more Gaussian-like when the mode of the distributions is further away from either zero or pole. Such observation leads to the following Taylor method to calculate the result variances when the distribution is nearly Gaussian in the $\tilde{z}$ representation.

2.9 Analytic Functions

Formula (2.13) 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.

2.9.1 Uncertainty of Taylor Expansion

An analytic function $f(x)$ can be accurately given in a range by the Taylor series as shown in Formula (2.21). Using Formula (2.9), Formula (2.23) and (2.24) [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 (2.22), and σ is defined as the *bounding factor*.

$$\tilde{y} = f(x+\tilde{x}) = f(x+\tilde{z}\delta x) = \sum_{n=0}^{\infty} \frac{f_x^{(n)}}{n!} \tilde{z}^n (\delta x)^n; \tag{2.21}$$

$$\zeta(n) \equiv \int_{-\sigma}^{+\sigma} \tilde{z}^n N(\tilde{z}) d\tilde{z}; \quad \zeta(2n+1) = 0; \tag{2.22}$$

$$\overline{f(x)} = \int_{-\sigma}^{+\sigma} f(x+\tilde{z}\delta x) N(\tilde{z}) d\tilde{z} = \sum_{n=0}^{\infty} (\delta x)^{2n} \frac{f_x^{(2n)} \zeta(2n)}{(2n)!}; \tag{2.23}$$

$$\begin{aligned} \delta^2 f(x) = \overline{f(x)^2} - \overline{f(x)}^2 &= \sum_{n=0}^{\infty} (\delta x)^{2n} \\ &\left(\zeta(2n) \sum_{j=1}^{2n-1} \frac{f_x^{(j)}}{j!} \frac{f_x^{(2n-j)}}{(2n-j)!} - \sum_{j=1}^{n-1} \frac{f_x^{(2j)} \zeta(2j)}{(2j)!} \frac{f_x^{(2n-2j)} \zeta(2n-2j)}{(2n-2j)!} \right); \end{aligned} \tag{2.24}$$

$$\begin{aligned} &= (\delta x)^2 (f_x^{(1)})^2 + (\delta x)^4 \left(f_x^{(1)} f_x^{(3)} + \frac{1}{2} (f_x^{(2)})^2 \right) \\ &\quad + (\delta x)^6 \left(\frac{1}{4} f_x^{(1)} f_x^{(5)} + \frac{1}{2} f_x^{(2)} f_x^{(4)} + \frac{5}{12} (f_x^{(3)})^2 \right) + o((\delta x)^8); \end{aligned} \tag{2.25}$$

When $f(x) \neq 0$, $\frac{\delta f(x)}{|f(x)|}$ is defined as the *normalized uncertainty* of $f(x)$. It is an approximation to the precision $P(f(x)) = \frac{\delta f(x)}{|f(x)|}$

2.9.2 Binding Factor

The choice of the bounding factor σ needs careful considerations. If $\sigma = \infty$, $\zeta(2n) = (2n-1)!!$, which may cause Formula (2.24) to diverge in most cases. When σ is limited, Formula (2.22) becomes Formula (2.26):

- For $2n < 10$ when $5 \leq \sigma$, $\zeta(2n)$ can be approximated by $\zeta(2n) = (2n-1)!!$ according to Formula (2.27).
- For large $2n$, Formula (2.26) reduces to Formula (2.28), 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.

$$\zeta(2n) = 2N(\sigma)\sigma^{2n}\sum_{j=1}^{\infty}\sigma^{2j-1}\frac{(2n-1)!!}{(2n-1+2j)!!}; \tag{2.26}$$

$$= (2n-1)\zeta(\sigma, 2n-2) - 2N(\sigma)\sigma^{2n-1}; \tag{2.27}$$

$$\sigma^2 \ll 2n: \quad \zeta(2n) \simeq 2N(\sigma)\frac{\sigma^{2n+1}}{2n+1}; \tag{2.28}$$

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 (2.22) is replaced by a uniform distribution between $[-\sigma, +\sigma]$, Formula (2.29) is similar to Formula (2.28):

$$\zeta(2n) \equiv \int_{-\sigma}^{+\sigma}\frac{1}{2\sigma}\tilde{z}^{2n}d\tilde{z} = 2\frac{1}{2\sigma}\frac{\sigma^{2n+1}}{2n+1}; \tag{2.29}$$

The limited range of $\tilde{x} \in (-\sigma\delta x, +\sigma\delta x)$ causes a bounding leakage ϵ according to Formula (2.30), 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].

$$\epsilon = 2 - 2\xi(\frac{1}{2} + \sigma); \tag{2.30}$$

When $\sigma \geq 5$, $\zeta(2n)$ is almost identical to $(2n-1)!!$ when $2n \leq 16$, which makes Formula (2.24) quite insensitive to the actual value of σ.

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

2.9.3 One-dimensional Examples

Formula (2.32) gives the result variance for e^x, which always converge:

$$e^{x+\tilde{x}} = e^x\sum_{n=0}^{\infty}\frac{\tilde{x}^n}{n!}; \tag{2.31}$$

$$\frac{\delta^2 e^x}{(e^x)^2} = \sum_{n=1}^{\infty}(\delta x)^{2n}\left(\sum_{j=1}^{2n-1}\frac{\zeta(2n)}{j!\,(2n-j)!} - \sum_{j=1}^{n-1}\frac{\zeta(2j)}{(2j)!}\frac{\zeta(2n-2j)}{(2n-2j)!}\right) \tag{2.32}$$

$$= (\delta x)^2 + \frac{3}{2}(\delta x)^4 + \frac{7}{6}(\delta x)^6 + \frac{5}{8}(\delta x)^8 + o((\delta x)^{10}); \tag{2.33}$$

Formula (2.35) gives the result variance for $\sin(x)$, which converge when $\delta x < 1$:

$$\sin(x+\tilde{x}) = \sum_{n=0}^{\infty}\sin(x)(-1)^n\frac{\tilde{x}^{2n}}{(2n)!} + \sum_{n=0}^{\infty}\cos(x)(-1)^n\frac{\tilde{x}^{2n+1}}{(2n+1)!}; \tag{2.34}$$

$$\delta^2\sin(x) = \sum_{n=1}^{\infty}(\delta x)^{2n}(-1)^{n-1}$$
$$\left(\cos(x)^2\sum_{j=0}^{n-1}\frac{\zeta(2n)}{(2j+1)!(2n-2j-1)!} - \sin(x)^2\sum_{j=1}^{n-1}\frac{\zeta(2n)-\zeta(2j)\zeta(2n-2j)}{(2j)!(2n-2j)!}\right) \tag{2.35}$$

$$= (\delta x)^2\cos(x)^2 - (\delta x)^4(\cos(x)^2\frac{3}{2} - \frac{1}{2}) + (\delta x)^6(\cos(x)^2\frac{7}{6} - \frac{1}{2}) + o((\delta x)^8); \tag{2.36}$$

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 x^c, in which $\binom{c}{n} = \frac{\prod_{j=0}^{n-1}(c-j)}{n!}$:

$$(x+\tilde{x})^c = x^c(1+\frac{\tilde{x}}{x})^c = x^c \sum_{n=1}^{\infty} \frac{\tilde{x}^n}{x^n} \binom{c}{n}; \tag{2.40}$$

$$\frac{\delta^2 x^c}{(x^c)^2} = \sum_{n=1}^{\infty} P(x)^{2n} \left(\sum_{j=1}^{2n-1} \zeta(2n) \binom{c}{j} \binom{c}{2n-j} - \sum_{j=1}^{n-1} \zeta(2j) \binom{c}{2j} \zeta(2n-2j) \binom{c}{2n-2j} \right) \tag{2.41}$$

$$= c^2 P(x)^2 + \frac{3}{2} c^2 (c-1)(c-\frac{5}{3}) P(x)^4 + \frac{7}{6} c^2 (c-1)(c-2)^2 (c-\frac{16}{7}) P(x)^6 + o(P(x)^8); \tag{2.42}$$

As the special cases for Formula (2.41), Formula (2.43) gives the result variance for x^2, Formula (2.44) gives the result variance for $\sqrt{x}$, Formula (2.45) gives the result variance for $1/x$, and Formula (2.46) gives the result variance for $1/x^2$:

$$P(x^2)^2 = 4P(x)^2 + 2P(x)^4; \tag{2.43}$$

$$P(\sqrt{x})^2 = \frac{1}{4} P(x)^2 + \frac{7}{32} P(x)^4 + \frac{75}{128} P(x)^6 + o(P(x)^8); \tag{2.44}$$

$$P(1/x)^2 = P(x)^2 + 8P(x)^4 + 69P(x)^6 + o(P(x)^8); \tag{2.45}$$

$$P(1/x^2)^2 = 4P(x)^2 + 66P(x)^4 + 960P(x)^6 + o(P(x)^8); \tag{2.46}$$

2.9.4 Convergence and Termination of Taylor Expansion

If Formula (2.24) can be expressed as $\sum_{n=1}^{\infty} \upsilon(2n)\zeta(2n)P(x)^{2n}$, such as Formula (2.38) and (2.41), and if $|\upsilon(2n)| \sim \lambda^{2n}$, then Formula (2.24) 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 (2.24) in the $P(x)$-form to converge is defined as the *applicable precision threshold*, which can be estimated as $1/\sigma$ according to Formula (2.28).

Formula (2.24) 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 4, 5, 6, and 7. Thus, for $(x+a)^c$ in which a is another constant, $P(x) = \delta x/|x-a|$ in Formula (2.41).

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 $P(x) \to \delta \ln(x)$, and the difference speed of variance increases of x^c depending on different c.

Both Formula (2.23) and (2.24) are subject to numerical errors and rounding errors such as from $f_x^{(j)}$, $\zeta(2j)$ and $(\delta x)^{2j}$, so both needs to be calculated using the variance arithmetic.

- The variance from Formula (2.23) can be added to the corresponding result of Formula (2.24).
- An uncertainty only needs to be correct on order of magnitude, such as with a precision of $1/\sigma = 0.2$ or finer. If the precision of the uncertainty becomes coarser than $1/\sigma$, the result should be invalidated.
- If within the range of $(x - \sigma\delta x, x + \sigma\delta x)$ an analytic function is monotonic, each term in either Formula (2.23) or (2.24) should be smaller than its corresponding previous term. Otherwise, the corresponding sum will diverge, and the result should be invalidated. On the other hand, if the new terms for both formula start to be smaller than the corresponding least significant values, the sums can be stopped.

2.9.5 Multiple Dimensional Expansion

Due to the uncorrelated uncertainty assumption, the Taylor expansion can be applied to multiple inputs, such as Formula (2.49) [37] .

$$f(x+\tilde{x}, y+\tilde{y}) = \sum_{m=0}^{\infty}\sum_{n=0}^{\infty}\frac{f_{(x,y)}^{(m,n)}}{m!n!}\tilde{x}^m\tilde{y}^n; \tag{2.47}$$

$$\begin{aligned}\overline{f(x,y)} &= \int\int f(x+\tilde{x}, y+\tilde{y})\rho(\tilde{x}, x, \delta x)\rho(\tilde{y}, y, \delta y)\, d\tilde{x}d\tilde{y} \\ &= \sum_{m=0}^{\infty}\sum_{n=0}^{\infty}(\delta x)^{2m}(\delta y)^{2n}\frac{\zeta(2m)\zeta(2n)f_{(x,y)}^{(2m,2n)}}{(2m)!(2n)!};\end{aligned} \tag{2.48}$$

$$\begin{aligned}\delta^2 f(x,y) &= \overline{f(x,y)^2} - \overline{f(x,y)}^2 = \sum_{m=0}^{\infty}\sum_{n=0}^{\infty}(\delta x)^{2m}(\delta y)^{2n} \\ &\quad (\zeta(2m)\zeta(2n)\sum_{i=0}^{2m}\sum_{j=0}^{2n}\frac{f_{(x,y)}^{(i,j)}}{i!\,j!}\frac{f_{(x,y)}^{(2m-i,2n-j)}}{(2m-i)!\,(2n-j)!} \\ &\quad - \sum_{i=0}^{m}\sum_{j=0}^{n}\frac{\zeta(2i)\zeta(2j)f_{(x,y)}^{(2i,2j)}}{(2i)!(2j)!}\frac{\zeta(2m-2i)\zeta(2n-2j)f_{(x,y)}^{(2m-2i,2n-2j)}}{(2m-2i)!(2n-2j)!})\end{aligned} \tag{2.49}$$

$$\begin{aligned}&\simeq (\delta x)^2(f_{(x,y)}^{(1,0)})^2 + (\delta y)^2(f_{(x,y)}^{(0,1)})^2 \\ &\quad + (\delta x)^4 f_{(x,y)}^{(1,0)}(\frac{1}{2}f_{(x,y)}^{(2,0)} + f_{(x,y)}^{(3,0)}) + (\delta y)^4(\frac{1}{2}f_{(x,y)}^{(0,2)} + f_{(x,y)}^{(0,1)}f_{(x,y)}^{(0,3)}) \\ &\quad + (\delta x)^2(\delta y)^2((f_{(x,y)}^{(1,1)})^2 + f_{(x,y)}^{(0,1)}f_{(x,y)}^{(2,1)} + f_{(x,y)}^{(1,0)}f_{(x,y)}^{(1,2)});\end{aligned} \tag{2.50}$$

Formula (2.7) and (2.11) are special cases of Formula (2.49). Although Formula (2.49) is only for 2-dimensional, it can be extended to any dimensional.

2.10 Dependency Tracing

$$\delta^2(f+g) = \delta^2 f + \delta^2 g + 2(\overline{fg} - \overline{f}\,\overline{g}); \tag{2.51}$$

$$\delta^2(fg) = \overline{f^2g^2} - \overline{fg}^2; \tag{2.52}$$

$$\delta^2 f(g) = \overline{f(g)^2} - \overline{f(g)}^2; \tag{2.53}$$

$$\delta^2(c_1 f + c_0) = c_1^2\delta^2 f; \tag{2.54}$$

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

- Formula (2.51) 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 (2.7). Formula (2.55) shows that Formula (2.24) uses Formula (2.51) between any two terms in the Taylor expansion of Formula (2.21).

$$\delta^2\left(\frac{f_x^{(m)}\tilde{x}^m}{m!}+\frac{f_x^{(n)}\tilde{x}^n}{n!}\right)=(\delta x)^{2m}(\frac{f_x^{(m)}}{m!})^2\eta(2m)+(\delta x)^{2n}(\frac{f_x^{(n)}}{n!})^2\eta(2n)+$$
$$2(\delta x)^{m+n}\left(\frac{f_x^{(m)}f_x^{(n)}}{m!\;n!}\eta(m+n)-\frac{f_x^{(m)}}{m!}\eta(m)\;\frac{f_x^{(n)}}{n!}\eta(n)\right); \tag{2.55}$$

- Formula (2.52) 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 (2.11).
- Formula (2.53) shows $\delta^2 f(g(x))$, whose dependence tracing can be demonstrated by $\delta^2(f^{-1}(f))=\delta^2 x$.
- Formula (2.54) shows the variance of linear transformation of a function, which can be applied to Formula (2.51) and (2.52) 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.

2.11 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 (2.24) and (2.49). 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 (2.7), (2.11), and (2.43) 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 7, variance arithmetic uses the matrix inversion formula by the determinants of sub-matrices, which is advised against by traditional numerical algorithm [12].
- Formula (2.56) is the single-variable version of Formula (2.53). 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 (2.57) 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. $P(\sqrt{x^2})^2\simeq P(x)^2+\frac{9}{8}P(x)^4$.

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 $(2x-1)^2(\delta x)^2+2(\delta x)^4$.

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.

$$\begin{aligned}\delta^2 f(g(x)) &= (\delta x)^2(f_x^{(1)}g_x^{(1)})^2+(\delta x)^4((f_x^{(1)})^2(g_x^{(1)}g_x^{(3)}+\frac{1}{2}(g_x^{(2)})^2)+\frac{1}{2}(f_x^{(2)})^2(g_x^{(1)})^4\\ &\quad+f_x^{(1)}f_x^{(2)}(g_x^{(1)})^4+4f_x^{(1)}f_x^{(2)}(g_x^{(1)})^2g_x^{(2)})+o((\delta x)^6); \qquad (2.56)\\ \delta^2 f(x)|_{\delta^2 x=\delta^2 g(x)} &= (\delta x)^2(f_x^{(1)}g_x^{(1)})^2+(\delta x)^4((f_x^{(1)})^2(g_x^{(1)}g_x^{(3)}+\frac{1}{2}(g_x^{(2)})^2)+\frac{1}{2}(f_x^{(2)})^2(g_x^{(1)})^4\\ &\quad+f_x^{(1)}f_x^{(3)}(g_x^{(1)})^4)+o((\delta x)^6); \qquad (2.57)\end{aligned}$$

3 Verification of the Variance Arithmetic

3.1 Verification Methods and Standards

Analytic functions each with precisely known results are used to validate the result uncertainties from variance arithmetic using the following statistical properties:

- *value error*: as the difference between the result using conventional floating-point calculation and the corresponding known analytic result.
- *normalized error*: as the ratio of a value error to the corresponding uncertainty.
- *error deviation*: as the standard deviation of the normalized errors.
- *error distribution*: as the histogram of the normalized errors.
- *uncertainty mean*: as the mean deviation of the result uncertainties.
- *uncertainty deviation*: as the deviation of the result uncertainties. When the uncertainty deviation is much smaller than the corresponding uncertainty mean, the result uncertainties are almost a constant.
- *error response*: as the relation between the input uncertainties and the result uncertainties.
- *calculation response*: as the relation between the amount of calculation and the result uncertainties.

One goal of calculation involving uncertainty is to precisely account for all source of input errors, to achieve *ideal coverage*. In the case of ideal coverage:

1. The error deviation is exactly 1.
2. The central limit theorem converges the result error distribution toward normal distribution.
3. The error response fits the function, such as a linear error response is expected for a linear function.
4. The calculation response fits the function, such as more calculations generally results in larger result uncertainties.

If instead, the input uncertainty is only correct for the input errors on the order of magnitude, the *proper coverage* is achieved. In case of proper coverage, the error deviation should be close to 1: to be in the range of $[0.1, 10]$.

When the input contains uncertainties whose deviation is not precisely known, such as he numerical errors of library functions using conventional floating-point calculations, Gaussian or white noises of increasing deviations can be added to the input, until idea coverage is reach. The needed minimal noise deviation gives a good estimation of the amount of the unspecified input uncertainties.

Gaussian or white noises of specified deviations can also be added to the input to measure the error responses of a function.

3.2 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 6 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.

3.3 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 a *round-trip* transformation which is a *forward* followed by a *reverse* transformation. The errors after the roundtrip transformation are 0 ideally [6], so both the error deviation and the error maximums should be 0. A test of uncertainty-bearing arithmetic using FFT algorithms is provided in Section 6, and a test of matrix inversion is provided in Section 7.

[6] When noises are added to the original input signal, the value errors should be the results minus the overall input data. The previous approaches [37] used the original input data wrongly.

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 5 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 Math Library Functions

Formula (2.32), (2.38), (2.35), and (2.41) are tested by the corresponding math library functions *exp*, *log*, *sin*, and *pow*, respectively.

At each point x for an input uncertainty δx, the result uncertainty is calculated by the variance arithmetic. The corresponding value deviation is obtained by:

1. Take 10000 samples from either Gaussian noise or uniform distribution with δx as the deviation, and construct $\tilde{x}$ which is x plus the sampled noise.
2. For each $\tilde{x}$, use the corresponding library function to calculate the value error as the difference between using $\tilde{x}$ and using x as the input.
3. The value error is divided by the result uncertainty, as the normalized error.
4. The standard deviation of the 10000 normalized errors is the error deviation.
5. In addition, for each of these tests, all value errors follow a same underlying distribution. The deviation of the value errors is defined as the *value deviation*, which the uncertainty should match.

4.1 Exponential

Figure 8 shows that the calculated uncertainties using Formula (2.32) agree very well with the measured value deviations for $e^{x+\delta x}$. As a result, the error deviations are very close to 1, even both the uncertainties and the value error deviations increase exponentially with x and δx. Such strong tracking power holds for all x of $e^{x+\delta x}$.

4.2 Logarithm

Figure 9 shows that the calculated uncertainties using Formula (2.38) agree very well with the measured value deviations for $\log(x+\delta x)$, so that the result error deviations are very close to 1, until the uncertainties suddenly diverge when the input precision is coarser than $1/5$ which is the estimated application precision threshold. After the result uncertainties diverge, the corresponding value deviations still continue on the same trends until the input data to the log function start to have negative values whenthe input noises are uniformly distributed. If Gaussian noises are used instead, the cutoffs for both value errors and error deviations are strictly at $1/5$. The result probability density function for $\log(x)$ has a pole at $x = 0$.

Such divergences of the result uncertainties are expected when the input precision is coarser than the estimated applicable precision threshold of $1/\sigma$, in which $\sigma = 5$ is the bonding factor of the variance arithmetic. If σ is reduced from 5 to 4, the measured applicable precision thresholds increase from $1/5$ to $1/4$, allowing more result uncertainties to be valid. The reduction of σ increases the bounding leakage ϵ from 5.7×10^{-7} to 6.3×10^{-5} according to Formula (2.30). A large leakage ϵ raises the logic questions on the validity of the result, because the leakage may mean $x \leq 0$ for $\log(x)$ in this case. In an extreme approach, in another variance arithmetic representation:

1. Each imprecise value carries its own σ or ϵ to indicate the statistical significance of its value to its uncertainty.
2. The result σ or ϵ involving two imprecise values can be found statistically.
3. The result of an analytic function always converges but at the expense of ϵ, such that when $x \to 0$, $\epsilon \to 1$ for $\log(x+\delta x)$.

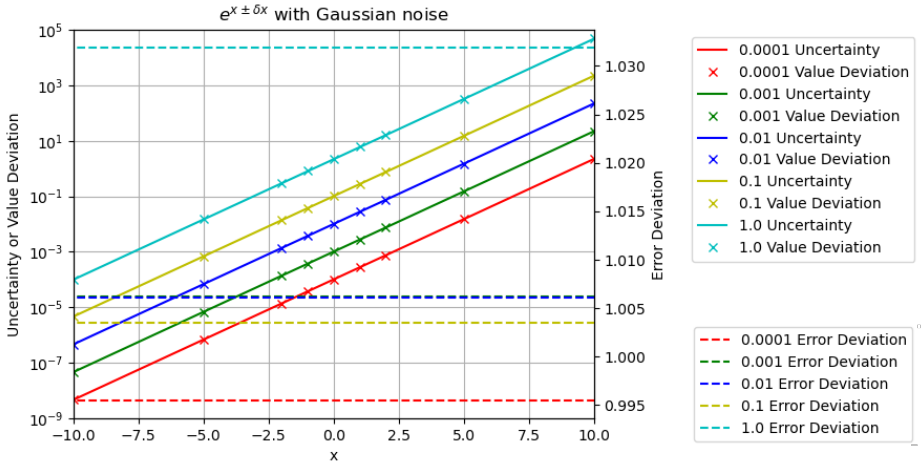

Figure 8: The calculated uncertainties vs the measured value deviations and the measured error deviations for $e^{x \pm \delta x}$, for different δx as shown in the legend. The uncertainties and the value deviations are drawn using the logarithmic y scales on the left side, while the error deviations are drawn using the linear y scales on the right side. Each color represent a δx. Gaussian noises are used to produce the input noise δx.

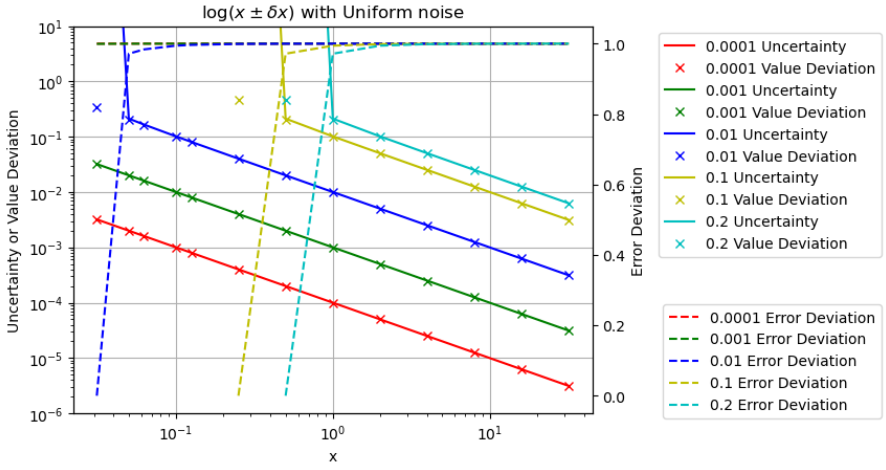

Figure 9: The calculated uncertainties vs the measured value deviations and the measured error deviations for $\log(x \pm \delta x)$, for different δx as shown in the legend. The uncertainties and the value deviations are drawn using the logarithmic y scales on the left side, while the error deviations are drawn using the linear y scales on the right side. Each color represent a δx. Gaussian noises are used to produce the input noise δx.

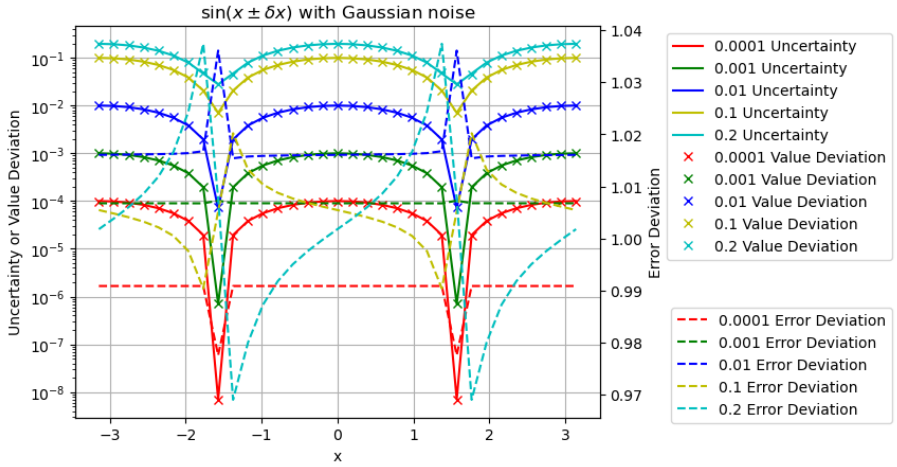

Figure 10: The calculated uncertainties vs the measured value deviations and the measured error deviations for $\sin(x \pm \delta x)$, for different δx as shown in the legend. The uncertainties and the value deviations are drawn using the logarithmic y scales on the left side, while the error deviations are drawn using the linear y scales on the right side. Each color represent a δx. Gaussian noises are used to produce the input noise δx.

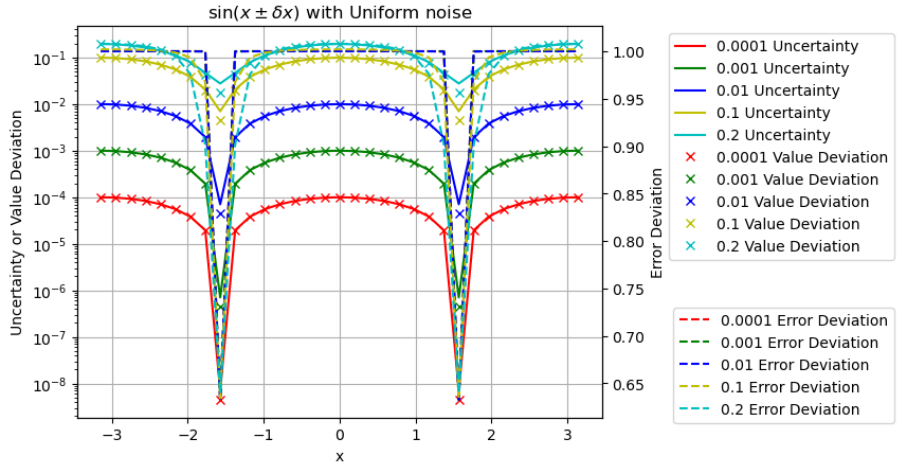

Figure 11: The calculated uncertainties vs the measured value deviations and the measured error deviations for $\sin(x \pm \delta x)$, for different δx as shown in the legend. The uncertainties and the value deviations are drawn using the logarithmic y scales on the left side, while the error deviations are drawn using the linear y scales on the right side. Each color represent a δx. Uniform noises are used to produce the input noise δx.

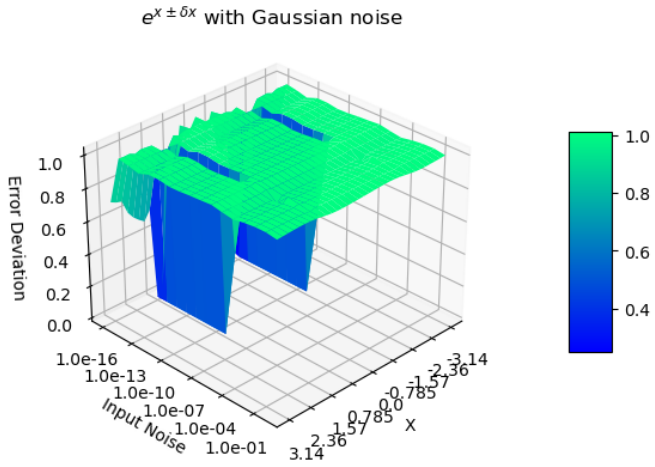

Figure 12: The error deviation for $\sin(x \pm \delta x)$, for x and δ. The x-axis is x between $-\pi$ and $+\pi$. The y-axis is δ between -10^{-16} and 1. The z-axis is the error deviation. Gaussian noises are used to produce the input noise δx.

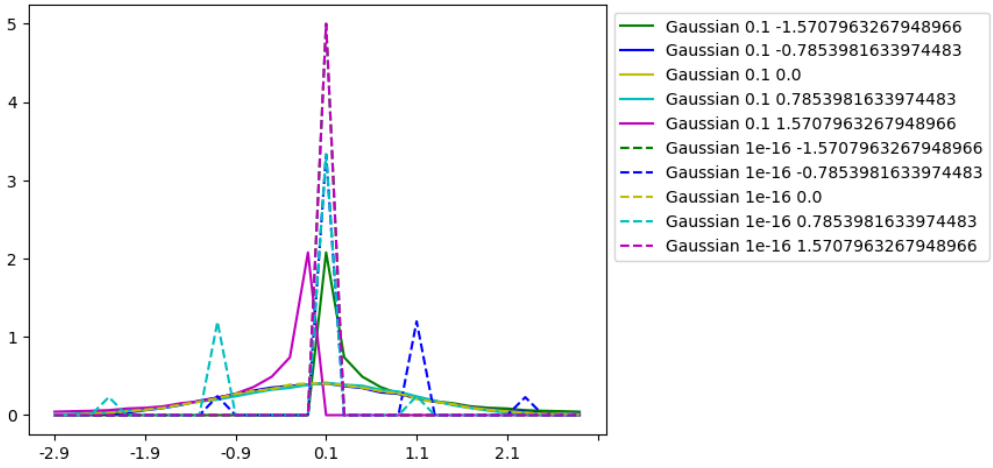

Figure 13: The histogram of the error deviation for $\sin(x \pm \delta x)$, for $x = -\pi/2, -\pi/4, 0, +\pi/4, +\pi/2$ and $\delta x = 10^{-1}, 10^{-16}$ as shown in the legend. The y-axis is the normalized count. Each color represents a different x, while each line pattern represents a different δx. Uniform noises are used to produce the input noise δx.

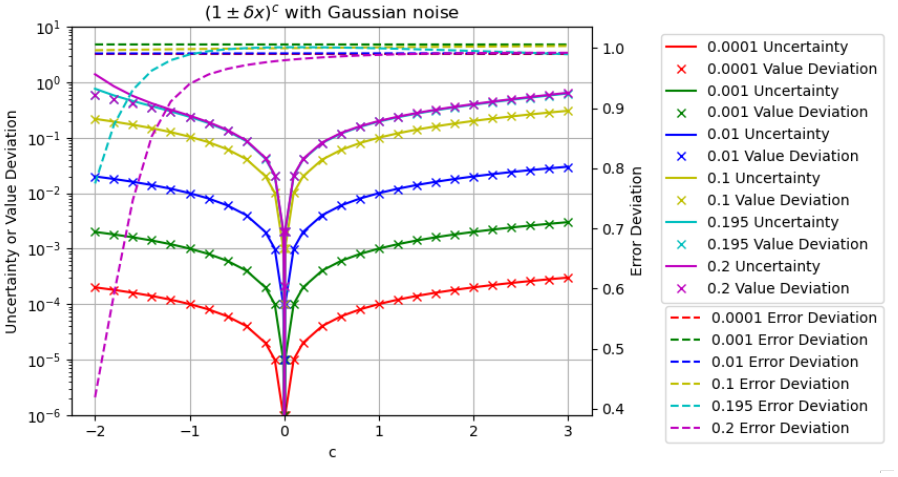

Figure 14: The calculated uncertainties vs the measured value deviations and the measured error deviations for $(1 \pm \delta x)^c$, for different δx as shown in the legend. The uncertainties and the value deviations are drawn using the logarithmic y scales on the left side, while the error deviations are drawn using the linear y scales on the right side. Each color represent a δx.

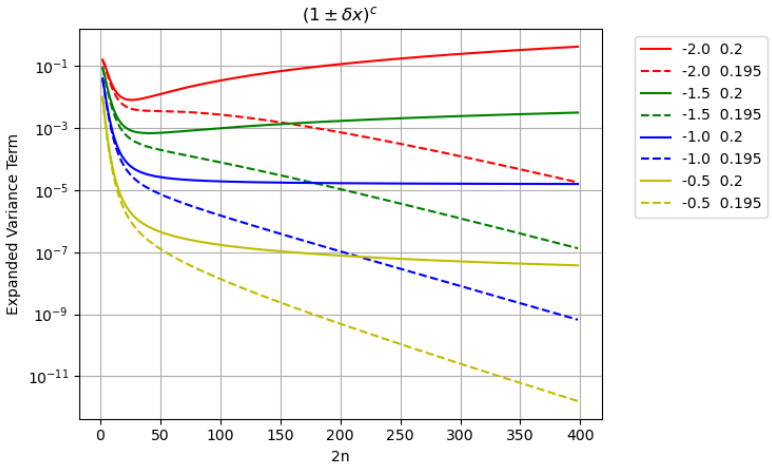

Figure 15: The individual variance term of $(1 \pm \delta x)^c$ vs the different Taylor expansion order $2n$, for different c and δx as shown in the legend. The x-axis is Taylor expansion order $2n$ from 1 to 400 in Formula (2.24). The y-axis is the contribution to $(1 \pm \delta x)^c$ at each Taylor expansion order $2n$ according to Formula (2.41). It shows that $(1 \pm 0.2)^{-2}$ and $(1 \pm 0.2)^{-1.5}$ clearly diverge, $(1 \pm 0.2)^{-1}$ seems diverges, and other cases clearly converge.

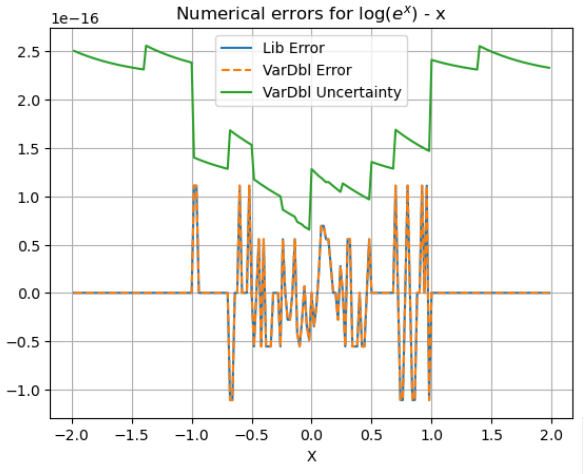

Figure 16: The values and uncertainties of $\log(e^x) - x$ vs x, as *VarDbl Error* and *VarDbl Uncertainty* in the legend. The result of the same calculation using conventional floating-point library functions is shown as *Lib Error* in the legend.

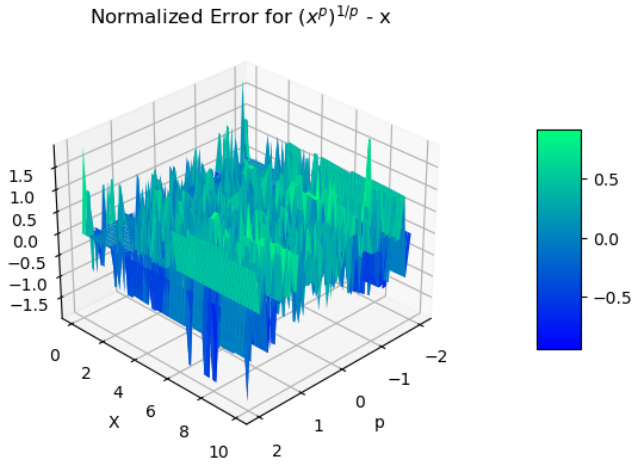

Figure 17: The normalized errors of $(x^p)^{\frac{1}{p}} - x$ vs x and p.

4. A calculation may branch based on the probability ϵ.

For the discussion simplicity, the cases of the bounding factor σ other than constant 5 are avoided in this paper.

4.3 Sine

Figure 10 shows that the calculated uncertainties using Formula (2.35) agree very well with the measured value deviations for $\sin(x+\delta x)$. Figure 10 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 $\delta^2 \sin(x) \simeq 0$.

In Figure 10, Gaussian noises are used, while in Figure 11, ideal white noises are added for then given sample size of 10000. The difference of Figure 10 and 11 shows that the Gaussian noise is more desirable, because of its tail effect, e.g., the error deviation is much closer to 1 for $\sin(\pm\pi/2+\delta x)$. So only Gaussian noises will be used for other analysis by default.

Figure 12 shows that the error deviation for $\sin(x+\delta x)$ is 1 except when $x = \pm\pi/4$ and $\delta x < 10^{-12}$, at where the probability density function has a zero. Figure 13 shows the histogram of the error deviation for $\sin(x + \delta x)$ when Gaussian noises are used to produce the input noise δx:

- When $\delta x = 10^{-16}$, the noise is comparable to the numerical calculation error of $\sin(x)$. The histogram is highly structured, with peaks near 0, ±1, and ±2, suggesting that the value errors are integer-folds of the least significant value.
- When $\delta x = 10^{-1}$, the noise is much large than the numerical calculation error of $\sin(x)$, so that ideal coverage is achieved. When $x = \pm\pi/2$, $\sin(x)_x^{(1)} = 0$, so that the result probability density function has a zero. The corresponding histogram is distorted from Gaussian, with the peak at $x = \pm\pi/2$, and all other values on one side of the peak only. Such distortion of the histogram results in 0 error deviations at $x = \pm\pi/2$ in Figure 12.

4.4 Power

Figure 14 shows that the calculated uncertainties of $(1 \pm \delta x)^c$ using Formula (2.41) agree very well with the measured value deviations for $(1 + \delta x)^c$ except when $\delta x = 0.2$ and $c < -1$. The reason why Formula (2.41) is not applicable exactly at the estimated applicable precision threshold $1/\sigma$ needs further discussion. Figure 15 shows the individual variance term of $(1 \pm \delta x)^c$ for the different Taylor expansion order $2n$, for different c and δx.

- Formula (2.45) and (2.46) show that $(1 \pm 0.2)^{-2}$ diverges faster than $(1 \pm 0.2)^{-1}$, which is confirmed by both Figure 14 and 15.
- When the deviation is reduced slightly to below $1/\sigma$ from 0.2 to 0.195, Figure 14 shows that the error deviations become much closer to 1. As the confirmation, Figure 15 shows reducing δx increases the convergence, e.g., $(1 \pm 0.2)^{-1.5}$ diverges while $(1 \pm 0.195)^{-2}$ converges.
- Figure 14 and 15 shows that the convergence is more sensitive to δx than c, so that the applicable precision threshold is relatively hard, which is estimated as $\delta x = 0.196$ for $c = -2$.

Figure 15 shows that the convergence of an analytic function at each input can be judged numerically and automatically for the given maximum Taylor expansion orders.

4.5 Numerical Errors for Library Functions

The combined numerical error of the library function e^x and $\log(x)$ is calculated as $\log(e^x) - x$ vs x. Figure 16 shows that using either the variance arithmetic or the conventional floating-point library functions results in the same value errors. In both cases, the value errors are 0 when $1 < |x|$. The uncertainties of the variance arithmetic bounds the value errors effectively, resulting in an error deviation about 0.409 when $|x| \leq 1$.

The numerical error of the library function x^p is calculated as $(x^p)^{1/p} - x$ vs x. Figure 17 shows that the normalized errors is not specific to either x or p, resulting in an error deviation about 0.548.

When no input noise is added, the error deviation of $\sin(x)^2 + \cos(x)^2 - 1$ is about 0.535

4.6 Summary

Formula (2.32), (2.38), (2.35), and (2.41) gives effective uncertainties for the corresponding library functions. Generally, when the input precision is above 10^{-15}, the input uncertainty achieves ideal coverage.

In ideal coverage cases, the error deviation is very close to 1 except it is 0 at where:

- when the result probability density function is zero, and δx is small enough, the uncertainty mean is finite.
- when the result probability density function is pole, and δx is large enough, the uncertainty mean is infinite.

In non ideal coverage cases, the error deviation is about 0.5 so that proper coverage is achieved.

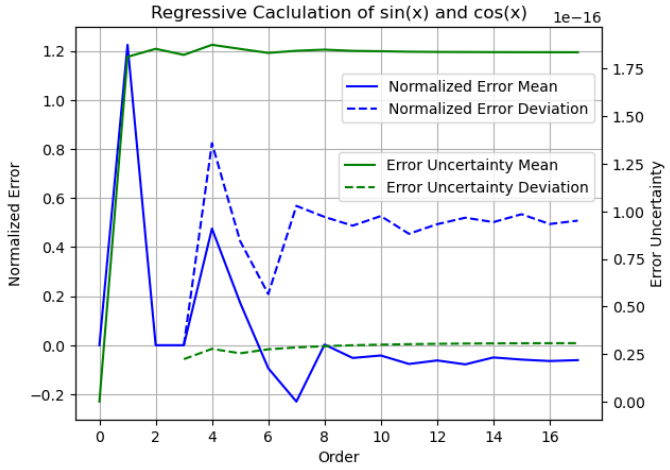

Figure 18: The error deviations and uncertainty means for $\sin(x)^2 + \cos(x)^2 - 1, x \in [0, \pi/4]$, for different regression order as shown in the legend.

5 Regressive Generation of Sin/Cos

Starting from Formula (5.1), Formula (5.2) and Formula (5.3) can be used recursively to calculate the sin library function.

$$\sin(0) = \cos(\frac{\pi}{2}) = 0; \qquad \sin(\frac{\pi}{2}) = \cos(0) = 1; \quad (5.1)$$

$$\sin\left(\frac{\alpha+\beta}{2}\right) = \sqrt{\frac{1-\cos(\alpha+\beta)}{2}} = \sqrt{\frac{1-\cos(\alpha)\cos(\beta)+\sin(\alpha)\sin(\beta)}{2}}; \quad (5.2)$$

$$\cos\left(\frac{\alpha+\beta}{2}\right) = \sqrt{\frac{1+\cos(\alpha+\beta)}{2}} = \sqrt{\frac{1+\cos(\alpha)\cos(\beta)-\sin(\alpha)\sin(\beta)}{2}}; \quad (5.3)$$

The number of regression is defined as the order. For each order n, 2^n $\sin(\frac{\pi}{2}\frac{i}{2^n})$ and $\cos(\frac{\pi}{2}\frac{i}{2^n})$ values are obtained, so that the result is more statistically stable with increasing n.

The errors of the generated $\sin(x)$ and $\cos(x)$ are checked by $\sin(x)^2 + \cos(x)^2 - 1$. The resulted value errors are comparable with but not identical to the value errors using floating-point library $\sin(x)$ and $\cos(x)$. Figure 18 shows that the result uncertainties remain almost a constant of about 1.2^{-16}, which is consist with uniformly distributed least significant values within the least significant bit of the significand. It also shows that the error deviations is close to 0.5, which is comparable to the value of 0.535 for the library $\sin(x)$ and $\cos(x)$ functions. The variance arithmetic is thus effective for this regressive generation algorithm.

Using either Taylor expansion or regressive generation of $\sin(x)$ and $\cos(x)$ are equivalent precision-wise for conventional floating-point calculations. In contrast, in the variance arithmetic, when $x \to 0$, Formula (5.2) has very coarse precision for Formula 2.44, which result in very coarse precision for $\sin(x)$. Thus, the variance arithmetic rejects this regressive generation algorithm for $\sin(x)$ and $\cos(x)$.

6 Verification Using FFT

6.1 Unfaithful Frequency Response of Discrete Fourier Transformation [37]

Each testing algorithm needs to come under careful scrutiny. One important issue 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 *unfaithful frequency response* of the discrete Fourier transformation.

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 (6.1) and (6.2), respectively [12], in which j is the *index time* and n is the *index frequency* for the discrete Fourier transformation [7]

$$H[n] = \sum_{k=0}^{N-1} h[k]\ e^{i2\pi kn/N}; \tag{6.1}$$

$$h[k] = \frac{1}{N} \sum_{n=0}^{N-1} H[n]\ e^{-i2\pi nk/N}; \tag{6.2}$$

Formula (6.3) is the discrete forward transformation $H[n]$ of a pure sine signal $h[k] = \sin(2\pi kf/N)$ in which f is the index frequency. The continuous forward transformation of the $h[k]$ is a delta function at $n = \pm f$ with phase $\pi/2$. $H[n]$ is delta-like function at $n = \pm f$ with phase $\pi/2$ only if f is an integer. In other cases, how much the result of discrete Fourier transformation deviates from continuous Fourier transformation depends 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 (6.3). Examples of unfaithful representations of fractional frequency by the discrete Fourier transformation are shown in Figure 19. The data for Figure 19 is generated using *SciPy*, which are very reproducible using any other math libraries, including *MathLab* and *Mathematica*.

$$\begin{aligned} H[n] &= \sum_{k=0}^{N-1} \sin(2\pi nk/N)\ e^{i2\pi nk/N} = \frac{1}{2i}\left(\sum_{k=0}^{N-1} e^{i2\pi(n+f)\frac{k}{N}} - \sum_{k=0}^{N-1} e^{i2\pi(n-f)\frac{k}{N}}\right) \\ &= \begin{cases} iN/2, & f \text{ is integer} \\ N/\pi, & f \text{ is integer} + 1/2 \\ \frac{1}{2}\frac{\sin(2\pi f - 2\pi\frac{f}{N}) + \sin(2\pi\frac{f}{N}) - \sin(2\pi f)e^{-i2\pi\frac{n}{N}}}{\cos(2\pi\frac{n}{N}) - \cos(2\pi\frac{f}{N})} & \text{otherwise} \end{cases} \end{aligned} \tag{6.3}$$

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

[7] The index frequency and index time are not necessarily related to time unit. The naming is just a convenient way to distinguish the two opposite domains in the Fourier transformation: the waveform domain vs the frequency domain.

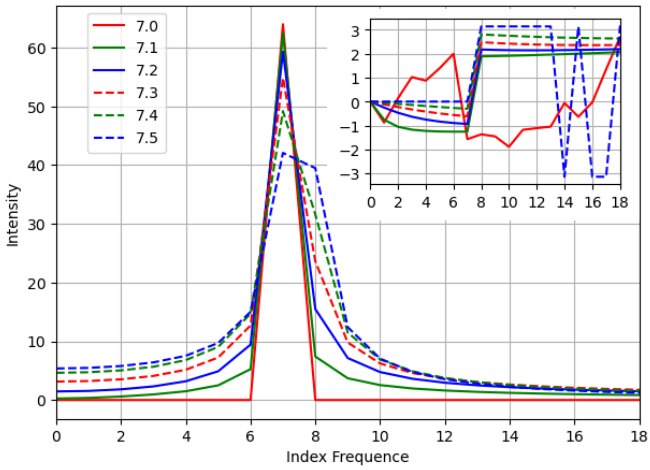

Figure 19: An unfaithful Discrete Fourier transformation is demonstrated by the spectra of a few sine signals having amplitude of 1 and slightly different frequencies as shown in legends. The x-axis shows the scale for the index frequency. The y-axis shows the scale for the intensity, while the y-axis of the embedded picture shows the scale for the phase. This figure is a practical reproduction of a previous theoretical figure [37].

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:

$$h(t) \Leftrightarrow H(s): \quad h(t-\tau) \Leftrightarrow H(s)e^{i2\pi s\tau}; \tag{6.4}$$

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 $h[k], k = 0 \ldots N-1$:

$$G[n] = (H[n] + h[N] - h[0])e^{i2\pi n/N}; \tag{6.5}$$

Applying Formula (6.4) to Formula (6.5) results in Formula (6.6).

$$h[N] = h[0]; \tag{6.6}$$

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 unfaithfulness of the discrete Fourier transformation to the Fourier transformation is a very serious example of modeling errors, but this problem has not be addressed seriously enough previously. Because the discrete Fourier transformation has widest applications in science and engineering, this problem need some serious attention.

6.2 FFT (Fast Fourier Transformation)

When $N = 2^L$, in which L is a positive integer, the generalized Danielson-Lanczos lemma [12] can be applied to the discrete Fourier transformation as FFT [12].

- For each output, each input is only used once, so there is no dependency problem when using Formula (2.7) and (2.11) as arithmetic operations. This simplicity avoids introducing numerical errors from Formula (2.24), because $f_x^{(j)}$ may contain numerical errors, while $(\delta x)^{2n}$ may contain rounding errors.
- When L is large, the large amount of input and output data enables high quality statistical analysis.
- The amount of calculation is L, because for each output, increasing L by 1 results in one additional step of sum of multiplication.
- Each step in the forward transformation thus increase the variance by 2-fold, so that the result uncertainty means increase with the FFT order L as $\sqrt{2}^L$. Because the reverse transformation divides the result by 2^L, the result uncertainty means decrease with the FFT order L as $\sqrt{1/2}^L$. The result uncertainty means for the roundtrip transformations is thus: $\sqrt{2}^L \times \sqrt{1/2}^L = 1$.
- The forward and reverse transformations are identical except a sign, so they are essentially the same algorithm, and their difference is purely due to input data.

A major question for the variance arithmetic is that whether the uncertainties can track the the corresponding value errors effectively or not. FFT transformations provide an ideal test for this statistical property: To test if the error deviations can be close to 1 with the increasing FFT order:

- The forward transformation cancels real imprecise data of a Sin/Cos signal into a spectrum of mostly 0 values, so that both its value errors and its result uncertainties are expected to grow faster.
- In contrast, the reverse transformation spread the spectrum of precise 0 values except at two peaks to data of a Sin/Cos signal, so that both its value errors and its result uncertainties are expected to grow slower.

6.3 Testing Signals

Only Formula (6.1) and (6.2) with integer n and k will be used.

The following signals are used for testing:

- *Sin*: $h[k] = \sin(2\pi kf/N), f = 1, 2, ...N/2 - 1$.
- *Cos*: $h[k] = \cos(2\pi kf/N), f = 1, 2, ...N/2 - 1$.

- *Linear*: $h[k] = k$, whose discrete Fourier transformation is Formula (6.7).

$$y \equiv i2\pi\frac{n}{N}: \quad G(y) = \sum_{k=0}^{N-1} e^{yk} = \frac{e^{Ny}-1}{e^y-1} = \begin{cases} y=0: & N \\ y\neq 0: & 0 \end{cases} \quad ;$$
$$H[n] = \frac{dG}{dy} = \begin{cases} n=0: & \frac{N(N-1)}{2} \\ n\neq 0: & -\frac{N}{2}(1+i/\tan(n\frac{\pi}{N})) \end{cases} \quad ; \tag{6.7}$$

Empirically, using the indexed sine functions:

- The results from Sin and Cos signals are statistically indistinguishable from each other.
- The results from Sin signals at different frequencies are statistically indistinguishable from each other.

So the results for Sin and Cos signals at all frequencies are pooled together for the statistical analysis, as the *Sin/Cos* signals.

6.4 Library Errors

Formula (6.1) and (6.2) limit the use of $\sin(x)$ and $\cos(x)$ to $x = 2\pi j/N$. To minimize the numerical errors of $\sin(x)$ and $\cos(x)$, *indexed sine functions* can be used instead of the library sine functions:

1. Instead of a floating-point value x for $\sin(x)$ and $\cos(x)$, the integer j is used to specify the input to $\sin(x)$ and $\cos(x)$, as $\sin(2\pi j/N)$ and $\cos(2\pi j/N)$, to avoid the floating-point rounding error of x.
2. The values of the indexed sine functions is extended from $j = 0, 1, \ldots N/8$ to the whole integer region using the periodicity of $\sin(2\pi j/N)$ and $\cos(2\pi j/N)$.

Figure 20 shows that the value errors between the library $\sin(x)$ and the indexed $\sin(x)$ increases with increasing x for FFT order 4. To confirm Figure 20, Figure 21 shows that the value errors of the library $\sin(x)$ for $\sin(j2\pi 2^4/2^6)$ increases with increasing j. The x range in Figure 20 covers all the input used in the FFT transformations of FFT order 4. The errors are also periodic in x by π, which may be amplified in the FFT calculations.

Figure 22 shows the value errors between the library $1/\tan(x)$ and the indexed $1/\tan(x)$ for $x \in (0, \pi)$. It covers all the integer inputs to the indexed cotan functions used for the Linear signal of FFT order 6 and 7. The value errors near $x = \pi$ increase with the FFT order.

Using the maximal absolute value errors, Figure 23 compares the value error strengths of the library $\sin(x)$ and $1/\tan(x)$ for increasing FFT orders. Figure 23 also shows that when checked by $\sin(x)^2 + \cos(x)^2 - 1$, the obtained indexed sine functions have proper coverage. In contrast, Figure 20, 21, and 22 all show that the library $\sin(x)$ and $1/\tan(x)$ do not have proper coverage.

It is clear that the indexed sine functions are much better than the library sine functions. Sadly, in reality, the library sine functions are used ubiquitously, so that both needs to be evaluated.

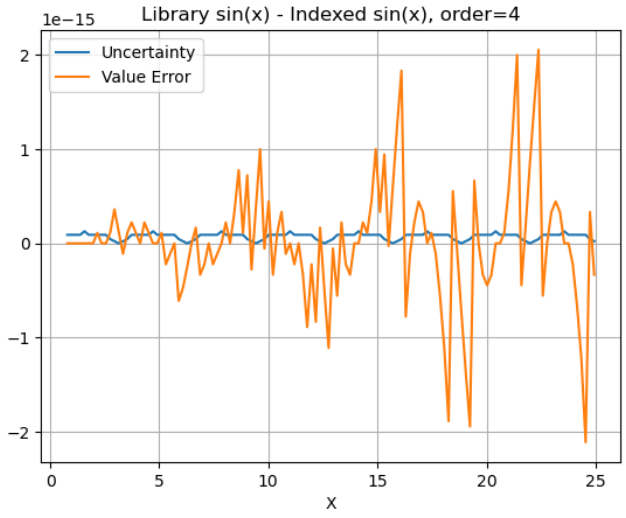

Figure 20: The difference between the library $\sin(x)$ and the indexed $\sin(x)$, for all integer input to the indexed sine functions used in the FFT transformations of FFT order 4. The uncertainty of the $\sin(x)$ values are also displayed, to mark the periodicity of π.

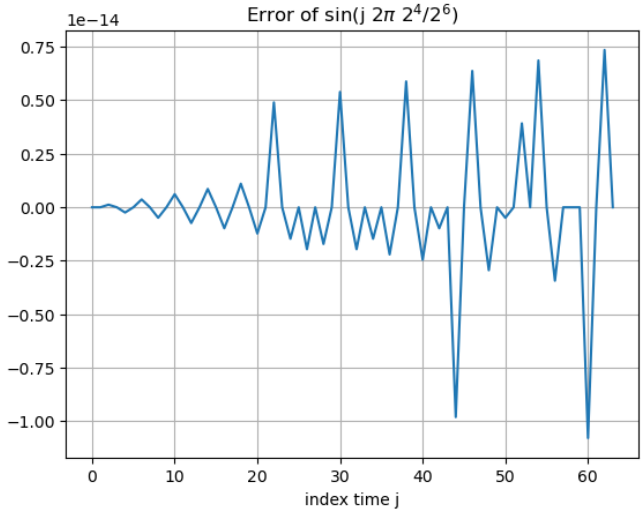

Figure 21: The difference between the library $\sin(x)$ and $\sin(j2\pi 2^4/2^6)$. $\sin(j2\pi 2^4/2^6)$ has precise values $[0, 1, 0, -1, \ldots]$, so that the corresponding least significant values are $[\ 0, 2.2\ 10^{-16},\ 0, 2.2\ 10^{-16}, \ldots]$.

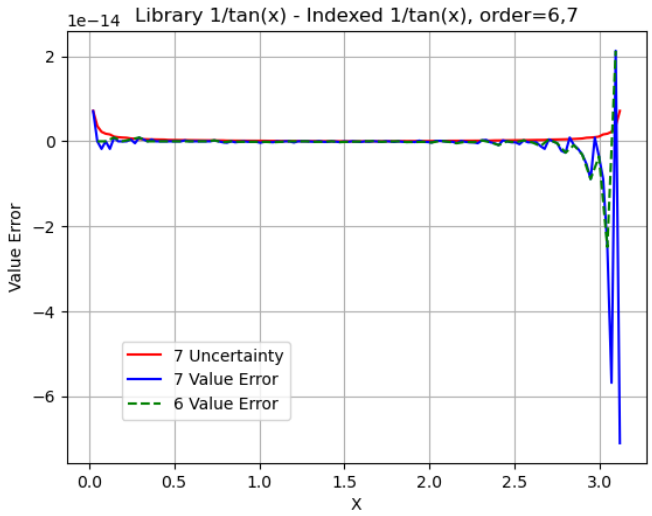

Figure 22: The difference between the library $\cos(x)/\sin(x)$ and the indexed $\cos(x)/\sin(x)$, for $x \in (0, \pi)$.

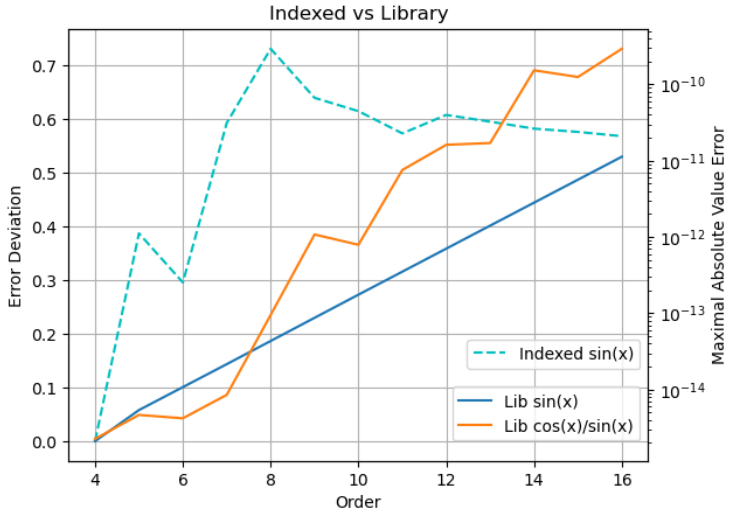

Figure 23: Comparing the indexed sine functions and the library sine functions for different FFT orders. The dashed line is the error deviations of $\sin(x)^2 + \cos(x)^2 - 1$ for the indexed sine functions. Its y-axis is on the left. The two solid lines are the maximal absolute value errors of the library $\sin(x)$ and $\cos(x)/\sin(x)$ when compared with the indexed $\sin(x)$ and $\cos(x)/\sin(x)$. Their y-axis is on the right.

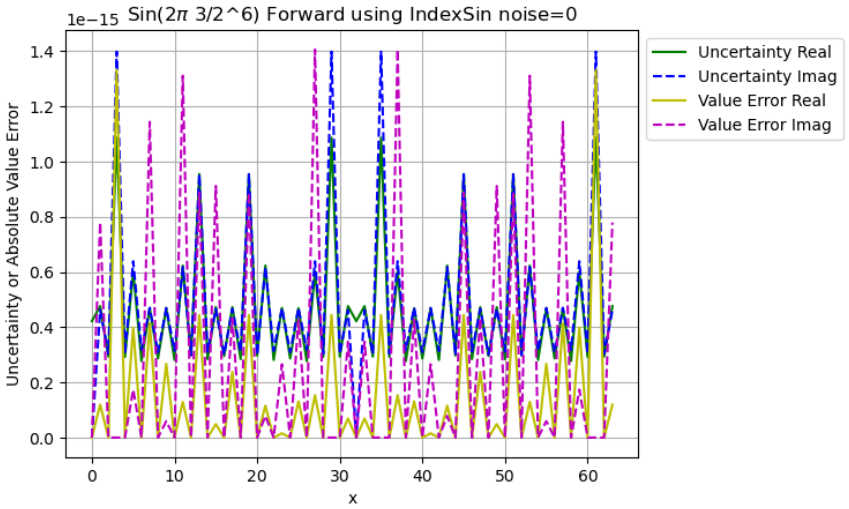

Figure 24: The FFT spectrum of $\sin(j3/2^6\pi)$ using the indexed sine functions after the forward transformation calculated by the variance arithmetic, with the uncertainty and the value errors shown in the legend. The x-axis shows the scale for index frequency. The y-axis shows the scale for uncertainty and absolute value errors.

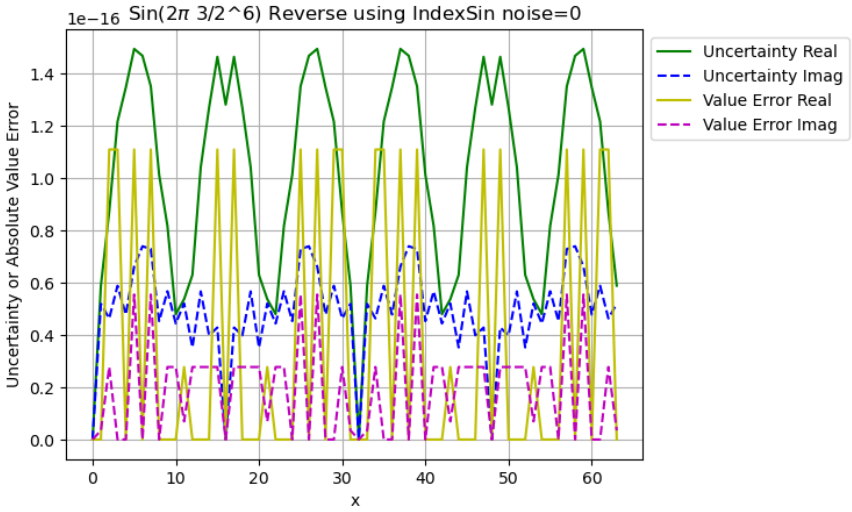

Figure 25: The FFT waveform of $\sin(j3/2^6\pi)$ using the indexed sine functions after the reverse transformation calculated by the variance arithmetic, with the uncertainty and the value errors shown in the legend. The x-axis shows the scale for index time. The y-axis shows the scale for uncertainty and absolute value errors.

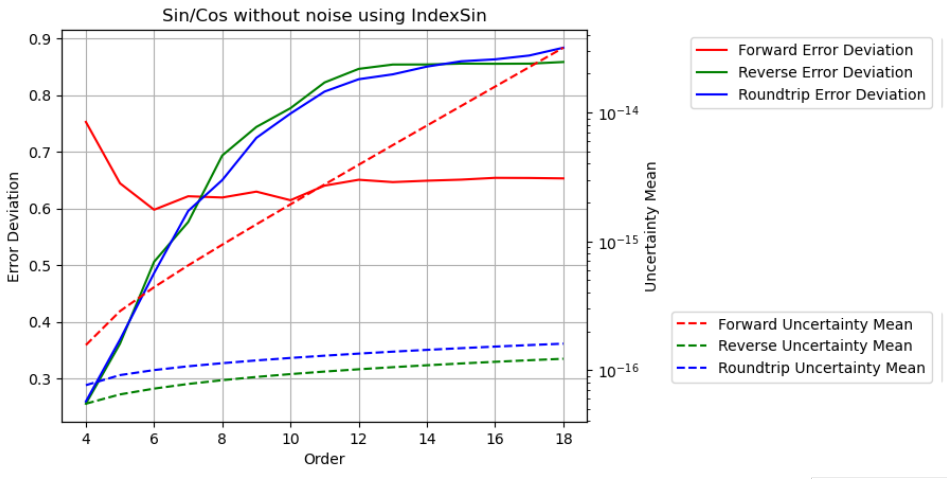

Figure 26: The result error deviations and uncertainty means of Sin/Cos signals vs. FFT order using the indexed sine functions for forward, reverse and roundtrip FFT transformations, as shown in the legend. The error deviations are scaled to the linear y-axis on the left, while the uncertainty means are scaled to the logarithmic log y-axis on the right.

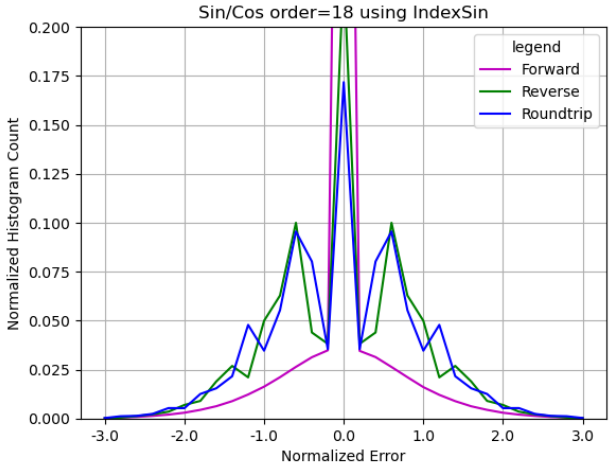

Figure 27: The histograms of the normalized errors of Sin/Cos signals without input noises using the indexed sine functions for forward, reverse and roundtrip FFT transformations, as shown in the legend. The FFT order is 18.

6.5 Using the Indexed Sine Functions for Sin/Cos Signals

Using the indexed sine functions, for the waveform of $\sin(2\pi k3/2^6)$ in which k is the index time:

- Figure 24 shows that for the forward transformation, the result value errors are comparable to the result uncertainties, with an error deviation of 0.37 for the real part, and 0.93 for the imaginary part.
- Figure 25 shows that for the reverse transformation, the result value errors are comparable to the result uncertainties, with an error deviation of 0.54 for the real part, and 0.52 for the imaginary part.

Figure 26 shows both the result uncertainty means and the error deviations vs. FFT order of FFT transformations of Sin/Cos signals using the indexed sine functions. Figure 27 shows the corresponding histograms at FFT order 17.

- As expected, the uncertainties grow much faster with the increasing FFT order for the forward transformation than those for the reverse transformation. Both are faster enough to achieve proper coverage.
- The faster growth of the uncertainties of the forward transformation in Figure 26 results in almost Gaussian distribution of the normalized errors for FFT order 18 in Figure 27. Thus, the forward transformation is expected to reach ideal coverage quicker with any added noise, and it is less sensitive to numerical calculation errors.
- The slower growth of the uncertainties of the reverse transformation in Figure 26 results in structured distribution on top of a Gaussian distribution of the normalized errors for FFT order 18 in Figure 27. Thus, the reverse transformation is expected to reach ideal coverage slower with added input noises, and it is more sensitive to numerical calculation errors.
- With the increasing FFT order, all histograms become more Gaussian like, and the error deviations for the real and the imaginary parts become more equal in value. The error deviations for the forward transformation are relatively stable. The error deviations for the reverse transformation increase with the increasing FFT orders until become stable when FFT orders is 12 or more. When the FFT order is about 12, statistical stability is reach.

Thus, proper coverage is achieved using the indexed sine functions for Sin/Cos signals.

6.6 Using the Library Sine Functions for Sin/Cos Signals

Because the least significant values are the only source of input uncertainties for variance arithmetic, the result uncertainties using either sine functions are almost identical.

The library sine functions contributes to the numerical calculation errors which is not specified by the input uncertainties of the variance arithmetic. The question is whether the variance arithmetic can track this large amount of numerical errors effectively or not.

Using the library sine functions, for the waveform of $\sin(2\pi k3/2^6)$ in which k is the index time:

- Figure 28 shows that for the forward transformation, the result value errors are noticeably larger than the result uncertainties, with an error deviation of 7.0 for

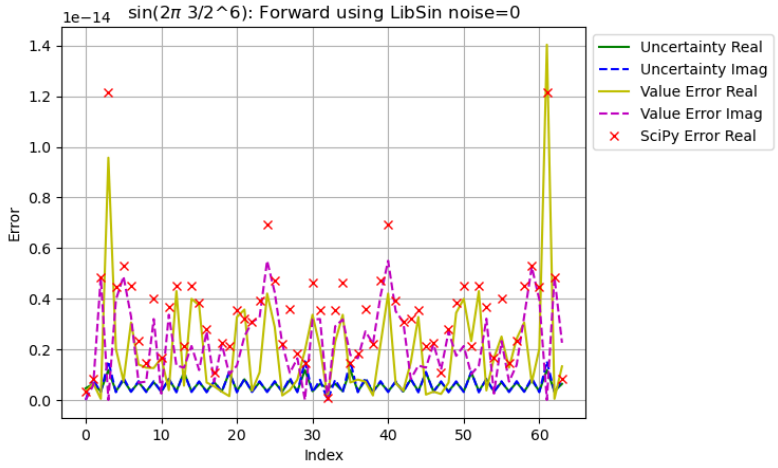

Figure 28: The FFT spectrum of $\sin(j3/2^6\pi)$ using the library sine functions after the forward transformation calculated by the variance arithmetic, with the uncertainties and the value errors shown in the legend. Also included is the corresponding result errors using *SciPy*. The x-axis shows the scale for index frequency. The y-axis shows uncertainties or absolute value errors.

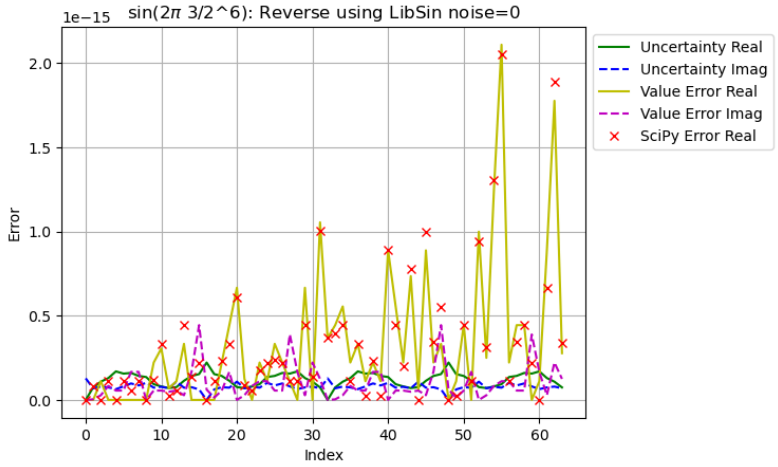

Figure 29: The FFT waveform of $\sin(j3/2^6\pi)$ using the indexed sine functions after the reverse transformation calculated by the variance arithmetic, with the uncertainties and the value errors shown in the legend. Also included is the corresponding result errors using *SciPy*. The x-axis shows the scale for index time. The y-axis shows the uncertainties or absolute value errors. No input noise is added to the Sin signal.

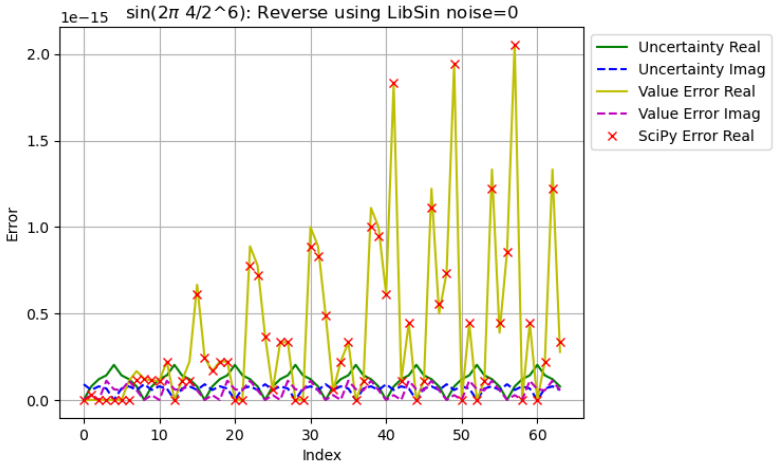

Figure 30: The FFT waveform of $\sin(j4/2^6\pi)$ using the indexed sine functions after the reverse transformation calculated by the variance arithmetic, with the uncertainties and the value errors shown in the legend. Also included is the corresponding result errors using *SciPy*. The x-axis shows the scale for index time. The y-axis shows the uncertainties or absolute value errors. No input noise is added to the Sin signal.

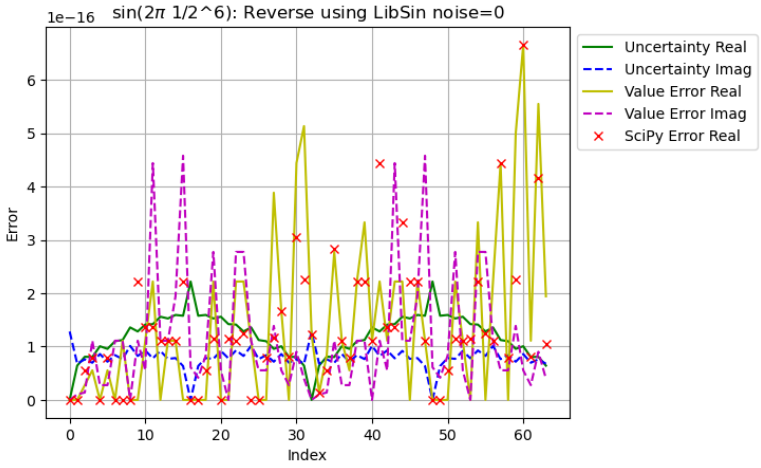

Figure 31: The FFT waveform of $\sin(j1/2^6\pi)$ using the indexed sine functions after the reverse transformation calculated by the variance arithmetic, with the uncertainties and the value errors shown in the legend. Also included is the corresponding result errors using *SciPy*. The x-axis shows the scale for index time. The y-axis shows the uncertainties or absolute value errors. No input noise is added to the Sin signal.

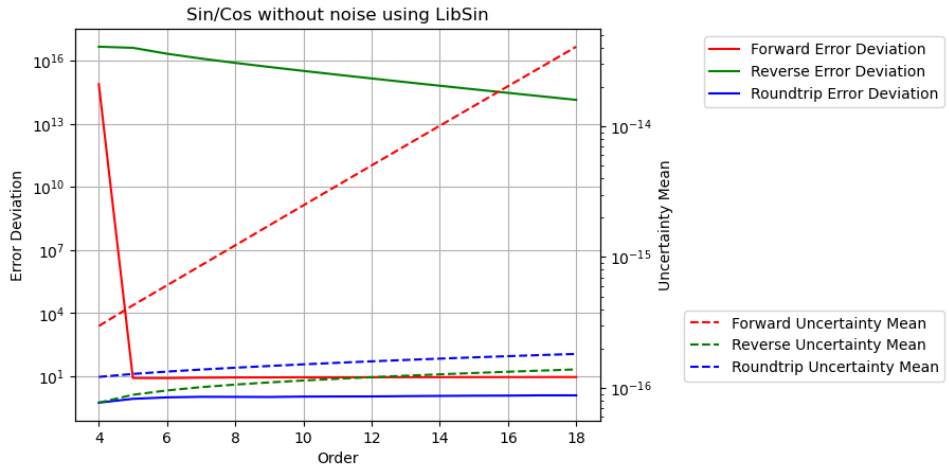

Figure 32: The result error deviations and uncertainty means of Sin/Cos signals vs. FFT order using the library sine functions for forward, reverse and roundtrip FFT transformations, as shown in the legend. The error deviations are scaled to the linear y-axis on the left, while the uncertainty means are scaled to the logarithmic log y-axis on the right. No input noise is added to the Sin signal.

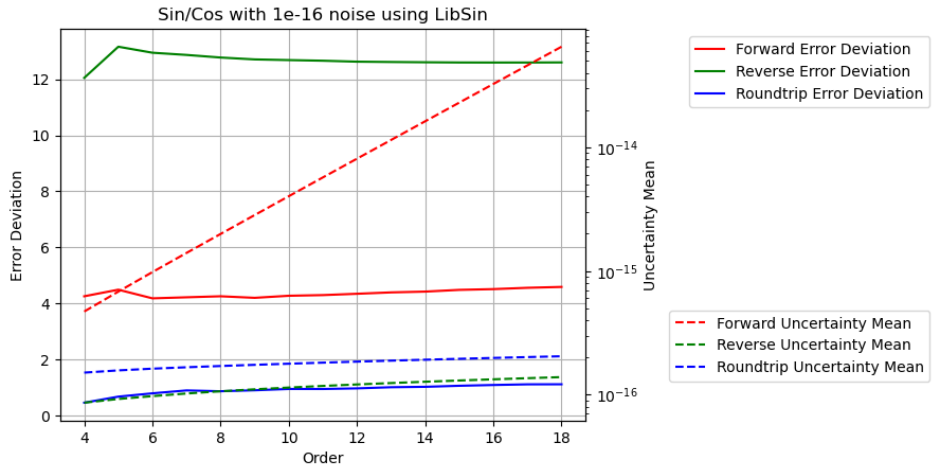

Figure 33: The result error deviations and uncertainty means of Sin/Cos signals vs. FFT order using the library sine functions for forward, reverse and roundtrip FFT transformations, as shown in the legend. The error deviations are scaled to the linear y-axis on the left, while the uncertainty means are scaled to the logarithmic log y-axis on the right. 10^{-16} input noises are added to the Sin/Cos signals.

the real part, and 5.6 for the imaginary part. As expected, the error deviations are noticeably larger than their counterparts in Figure 24.

- Figure **??** shows that for the reverse transformation, the result value errors are noticeably larger than the result uncertainties, with an error deviation of 5.7 for the real part, and 0.83 10^{15} for the imaginary part. The surprisingly large imaginary error deviation is caused by the small uncertainty at the index time where the value error is at a local maximum.
- As the result of of the huge error deviation, Figure 32 shows that the result of the reverse transformation no longer has proper coverage for all FFT orders. When a small noise with deviation of 10^{16} is added to the input, the result uncertainty of the reverse transformation at the index time of 8 is no longer near 0, so that the result error deviations achieve proper coverage again, as show in Figure 33.

To validate the FFT implementation in the variance arithmetic, the results are compared with the corresponding calculations using the python numerical library *SciPy* in Figure 28 and **??**. The results are quite comparable, except that in *SciPy*, the data waveform is always real rather than complex. In Figure 28, the *SciPy* results have been made identical for the frequency indexes f and $-f$, and the matching to the corresponding results of the variance arithmetic is indicative. In Figure **??**, the *SciPy* results matches the corresponding results of the variance arithmetic well.

In Figure **??**, the value errors tend to increase with the indexes, which is due to the periodic increase of the numerical errors of $\sin(x)$ with x as shown in Figure 20. When the signal frequency increases, the increase of the value errors with the index frequency in the reverse transformation becomes stronger, as shown in Figure 31, **??**, and 30. On the other hand, Figure 25 show no such increase, because it contains no library errors as described in Figure 20.

The variance arithmetic reveals the reason why to add small noises to the numerically generated input data, which is already a common practice [12]. How much noises to add to input to achieve proper coverage remains an art. When the proper tracking of the variance arithmetic is not achieved, the result may contain large amount of errors.

6.7 Linear Signal

In addition to the numerical errors of $\sin(\pi j/2^L), j = 0, 1, 2...2^{L-1}$, Linear signals introduce the numerical errors of $1/\tan(\pi(j-1)/2^L), j = 1, 2...2^{L-1}$ to the result. The question is whether the variance arithmetic can track this additional numerical errors or not.

Using the indexed sine functions:

- Figure 34 shows that proper coverage can be achieved for all the FFT transformations for all FFT orders.
- Figure 35 shows that for each transformation, the histogram of the Linear signal are quite quite similar to the counterpart of the Sin/Cos signals in Figure 27.

Using the library sine functions:

- Figure 36 shows that proper coverage can not be achieved any FFT transformation for all FFT orders, because the value errors outpace the uncertainties with the increasing FFT order. Because of this, adding noise to the input can not achieve proper coverage.

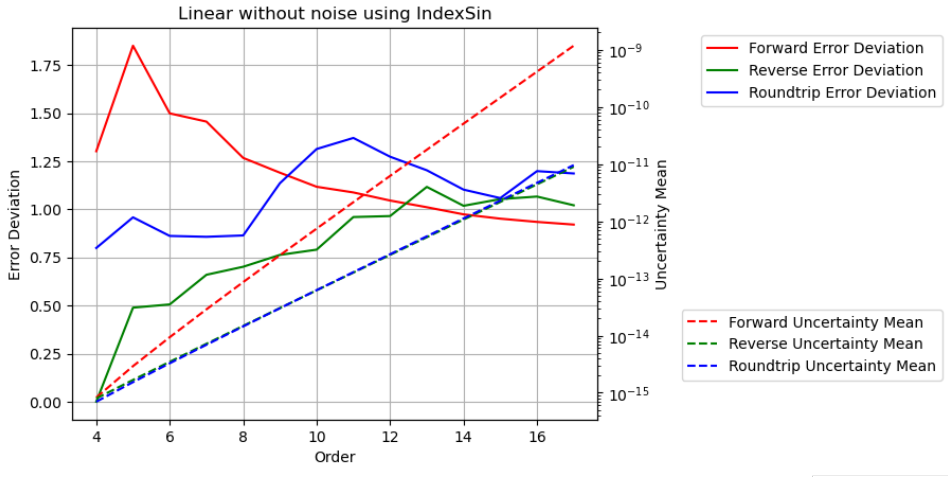

Figure 34: The result error deviations and uncertainty means of Linear signal vs. FFT order using the library sine functions for forward, reverse and roundtrip FFT transformations, as shown in the legend. The error deviations are scaled to the linear y-axis on the left, while the uncertainty means are scaled to the logarithmic log y-axis on the right. No input noise is added to the Linear signal.

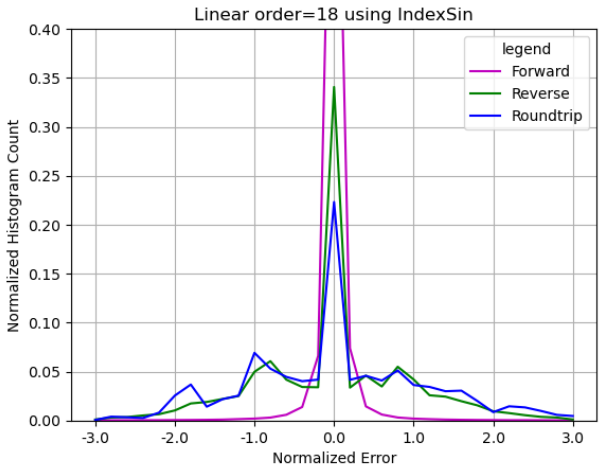

Figure 35: The histograms of the normalized errors of Sin/Cos signals without input noises using the indexed sine functions for forward, reverse and roundtrip FFT transformations, as shown in the legend. No input noise is added to the Linear signal.

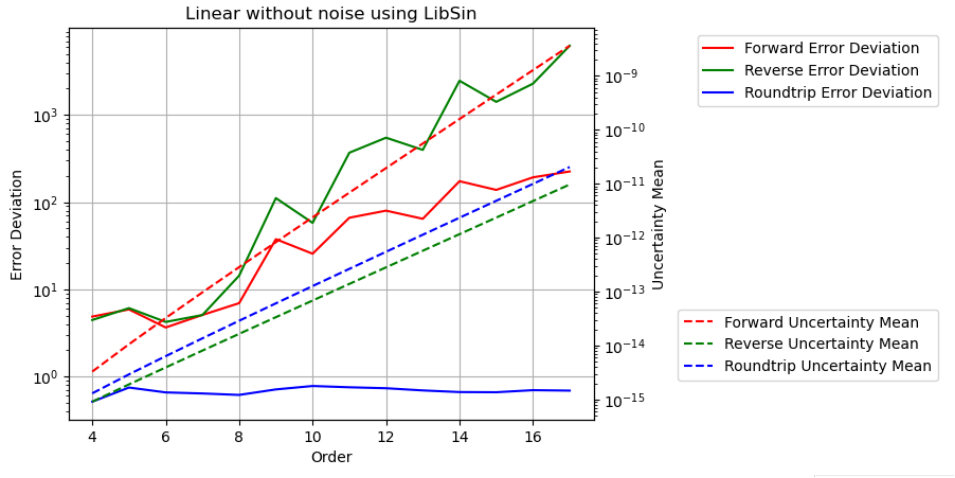

Figure 36: The result error deviations and uncertainty means of Linear signal vs. FFT order using the library sine functions for forward, reverse and roundtrip FFT transformations, as shown in the legend. The error deviations are scaled to the linear y-axis on the left, while the uncertainty means are scaled to the logarithmic log y-axis on the right. No input noise is added to the Linear signal.

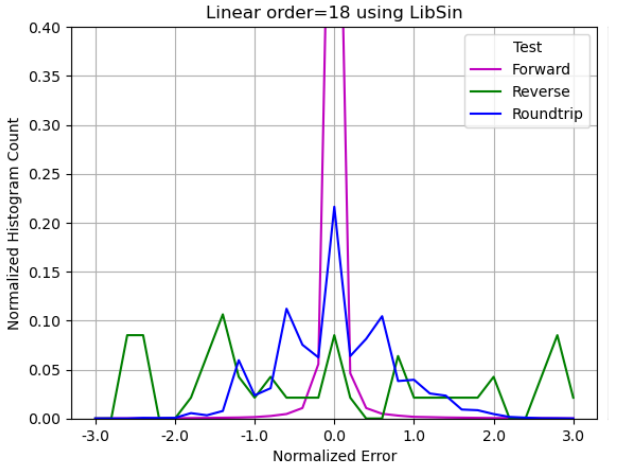

Figure 37: The histograms of the normalized errors of Sin/Cos signals without input noises using the indexed sine functions for forward, reverse and roundtrip FFT transformations, as shown in the legend. No input noise is added to the Linear signal.

- Figure 37 shows that the reverse histogram contains additional peaks in addition to its counterpart in Figure 36. The additional peaks extend beyond the range of $[-3, +3]$

When too much numerical errors are not specified by the input uncertainties, proper coverage can not be achieved, which is the case when the $\sin(x)$ numerical errors in Figure 20 is combined with the $1/\tan(x)$ numerical errors in Figure 22. The only solution seems to create a new sin library using the variance arithmetic so that all numerical calculation errors are accounted for.

6.8 Ideal Coverage

Even when proper coverage can not be achieved, adding enough noise to the input can overpower the numerical calculation errors, to achieve ideal coverage.

Figure 42 shows that with 10^{-3} input noise, the result normalized errors for the forward and reverse transformations are both normal distributed, while the result normalized errors for the roundtrip transformation is delta distributed, regardless of the types of input noises.

Figure 43 shows the result error deviations and uncertainty means vs. different input uncertainties for Linear signal of order 18 using the library sin/cos functions. When the deviation of the input noise is more than 10^{-3}, the input uncertainties become ideal coverage:

- As expected, the result uncertainty means for the forward transformations increase with the FFT order L as $\sqrt{2}^L$.
- As expected, the result uncertainty means for the reverse transformations decrease with the FFT order L as $\sqrt{1/2}^L$.
- As expected, the result uncertainty means for the roundtrip transformations always equal the corresponding input uncertainties.
- As expected, the result uncertainty means for both the forward and the reverse transformations are linear to the input uncertainties, respectively, because FFT transformations are linear. As expected, the result uncertainty means for the roundtrip transformation recover the corresponding input uncertainties perfectly.
- As expected, the normalized errors for the forward and the reverse transformations are normal distributed, even when the input noise is no longer Gaussian. The normalized errors for the roundtrip transformations are delta distributed at 0, meaning the input uncertainties are perfectly recovered.
- As expected, the result error deviations for the forward and reverse transformations are constant 1, while the result error deviations for the roundtrip transformation approaches 0 exponentially with the increasing FFT order.

For Linear signals using the library sine functions, the the added noise vs FFT order regions for the ideal coverage are shown as the regions where the error deviations are 1, by Figure 44 and 45 for the forward and reverse transformations, respectively. In other regions, proper coverage is not achievable. Because the uncertainties grow slower in the reverse transformation than in the forward transformation, the reverse transformations have smaller ideal coverage region than that of the forward transformation. Because the amount of numerical errors increases with the amount of calculations, the input noise range reduces with the increasing FFT order. It is possible that ideal coverage is

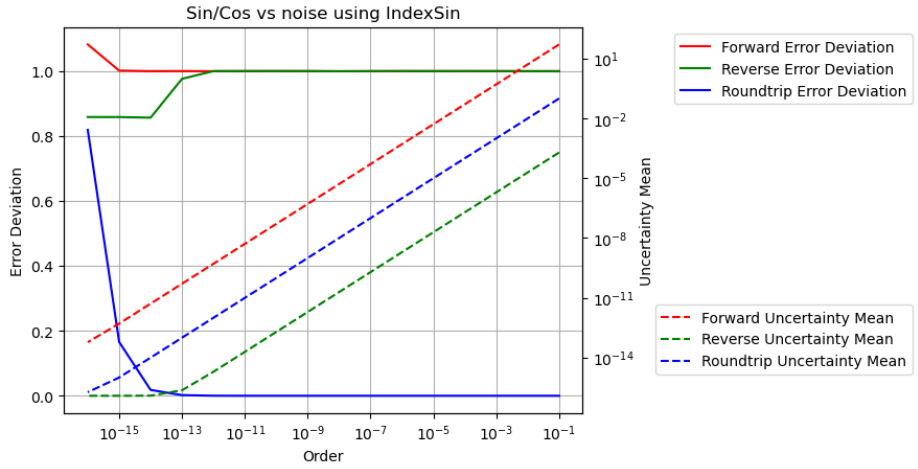

Figure 38: The result error deviations and uncertainty means for Sin/Cos signals of order 16 using indexed sin/cos functions vs. input uncertainties for forward, reverse and roundtrip transformations for FFT order 18. The error deviations are scaled to the linear y-axis on the right, while the uncertainty means are scaled to the logarithmic log y-axis on the right.

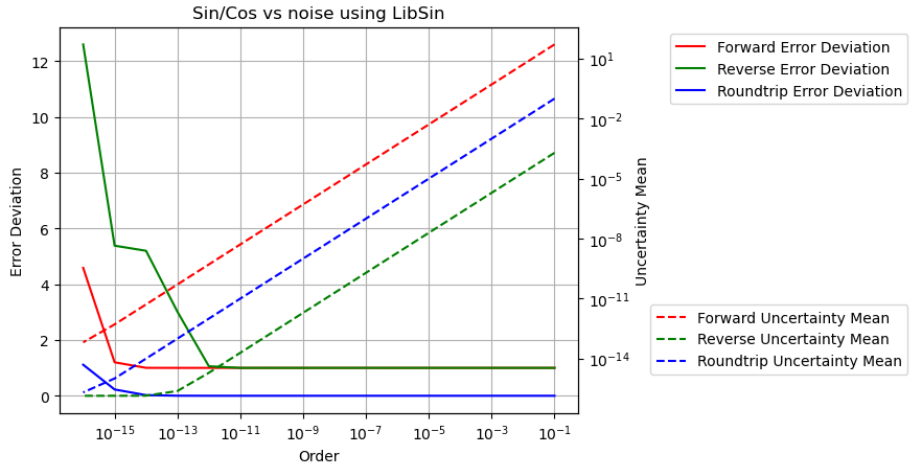

Figure 39: The result error deviations and uncertainty means for Sin/Cos signals of order 16 using library sin/cos functions vs. input uncertainties for forward, reverse and roundtrip transformations for FFT order 18. The error deviations are scaled to the linear y-axis on the right, while the uncertainty means are scaled to the logarithmic log y-axis on the right.

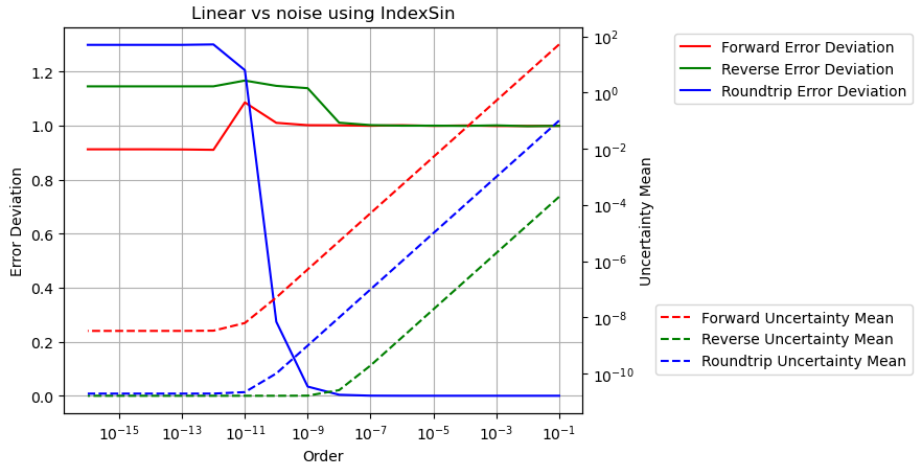

Figure 40: The result error deviations and uncertainty means using Linear signals of order 18 vs. input uncertainties for forward, reverse and roundtrip transformations for FFT order 18. The error deviations are scaled to the linear y-axis on the right, while the uncertainty means are scaled to the logarithmic log y-axis on the right.

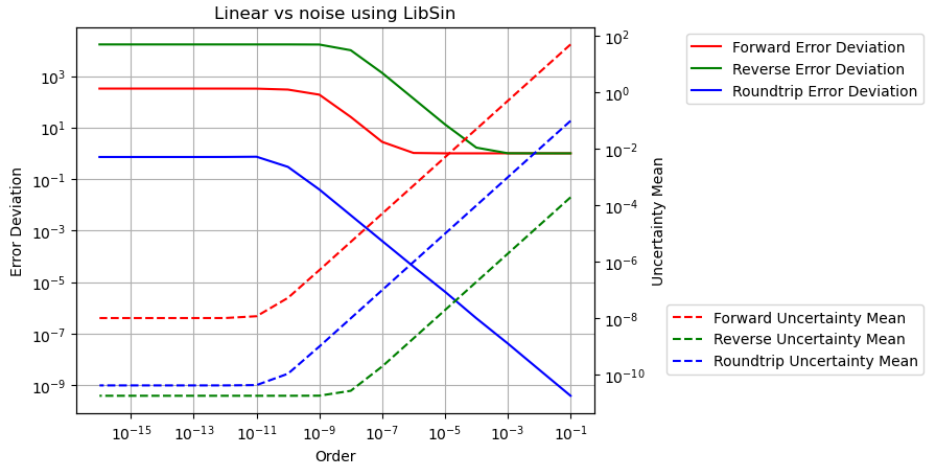

Figure 41: The result error deviations and uncertainty means using Linear signals of order 18 vs. input uncertainties for forward, reverse and roundtrip transformations for FFT order 18. The error deviations are scaled to the linear y-axis on the right, while the uncertainty means are scaled to the logarithmic log y-axis on the right.

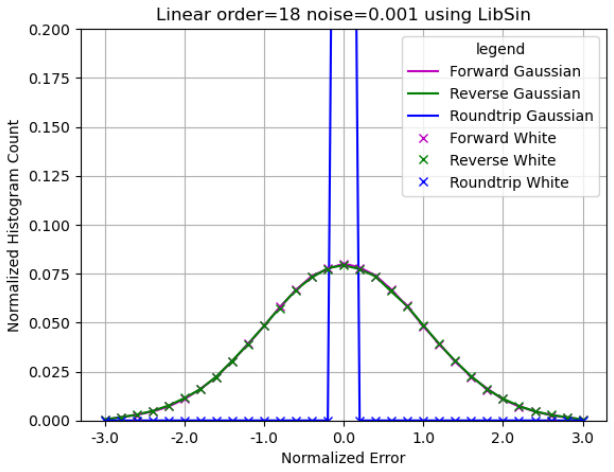

Figure 42: The histograms of the normalized errors of Linear signal without input noises using the library sine functions for forward, reverse and roundtrip FFT transformations, as shown in the legend. 10^{-3} input noises are added to the Sin/Cos signals.

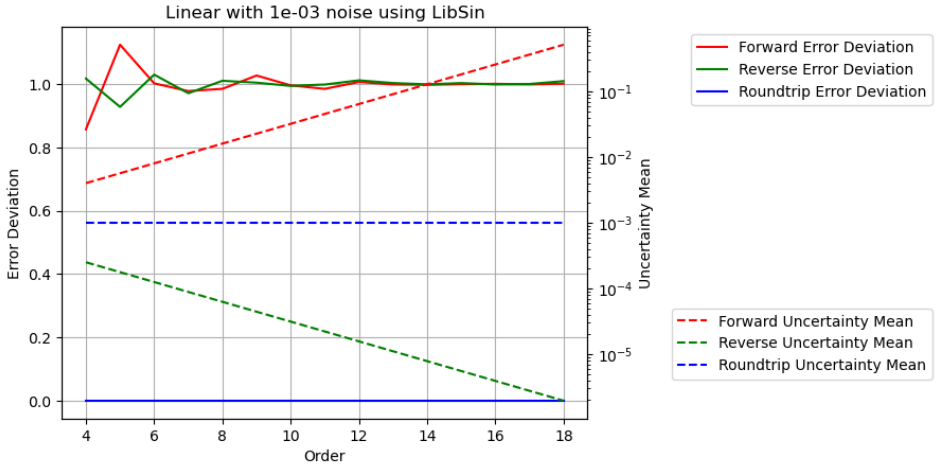

Figure 43: The result error deviations and uncertainty means using Linear signal with 10^{-3} input noises vs. FFT order for forward, reverse and roundtrip FFT transformations. The error deviations are scaled to the linear y-axis on the left, while the uncertainty means are scaled to the logarithmic log y-axis on the right.

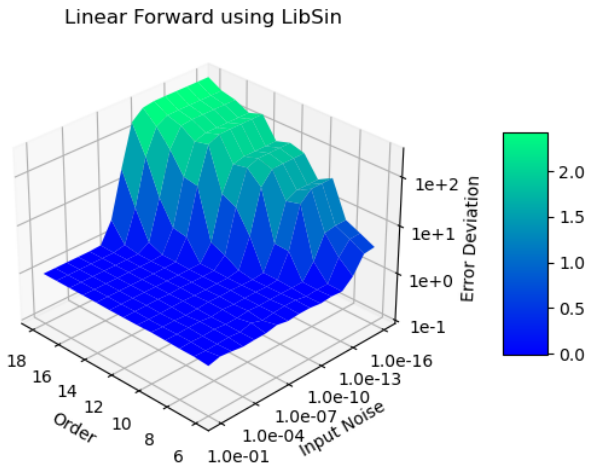

Figure 44: The result error deviations for Linear signals using library sine functions vs. input uncertainties and FFT orders for the forward transformations. The input uncertainties run from 10^{-16} to 10^{-1}, while the FFT Order runs from 6 to 18.

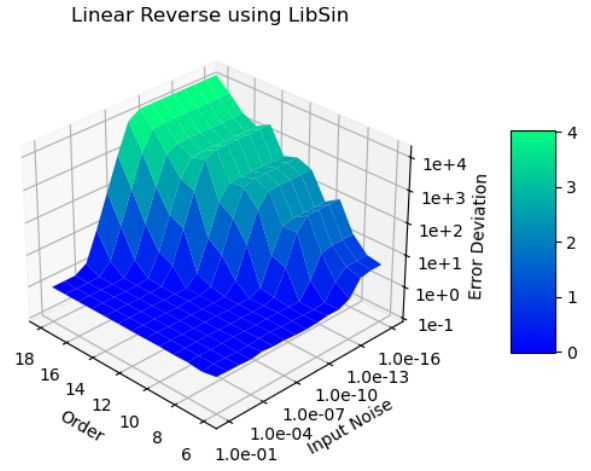

Figure 45: The result error deviations for Linear signals using library sine functions vs. input uncertainties and FFT orders for the reverse transformations. The input uncertainties run from 10^{-16} to 10^{-1}, while the FFT Order runs from 6 to 18.

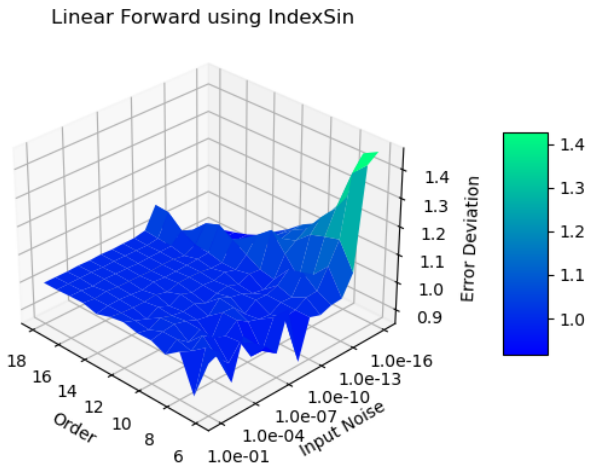

Figure 46: The result error deviations for Linear signals using indexed sine functions vs. input uncertainties and FFT orders for the forward transformations. The input uncertainties run from 10^{-16} to 10^{-1}, while the FFT Order runs from 6 to 18.

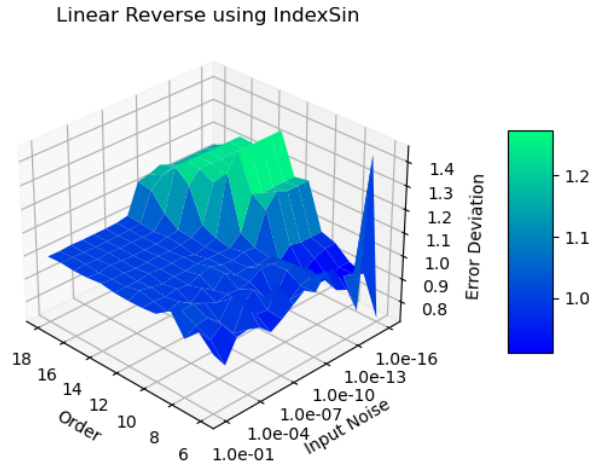

Figure 47: The result error deviations for Linear signals using indexed sine functions vs. input uncertainties and FFT orders for the reverse transformations. The input uncertainties run from 10^{-16} to 10^{-1}, while the FFT Order runs from 6 to 18.

not achievable at all, e.g., visually, when the FFT order is larger than 25 for the reverse transformation. As one of the most robust numerical algorithm that is very insensitive to input errors, FFT breaks down when the FFT order is more than 25 due to the numerical errors in the library sine functions, and such deterioration of the calculation result is not easily detectable using the conventional floating-point calculation.

In contrast, for Linear signals using the indexed sine functions, the the added noise vs FFT order regions for the ideal coverage are shown as the regions where the error deviations are 1, by Figure 46 and 47 for the forward and reverse transformations, respectively. The ideal regions are much larger. In other regions, proper coverage are achieved.

6.9 Summary

Compared to its counterpart continuous Fourier Transformation (FT), the discrete Fourier transformation (DFT) has large modeling error for its implied assumption that any waveform is periodic outside its defined time range. This modeling error has not been addressed seriously.

The library sine functions using conventional floating-point arithmetic have been shown to contain numerical errors as large as equivalently 10^{-3} of input accuracy for FFT transformations. The library $\sin(x)$ errors increase periodically with x, causing noticeable increase of the result errors with the increasing index frequency in the reverse transformation. The dependency of the result numerical errors on the amount of calculation and input data means that a small scale test can not properly qualify the result of a large scale calculation. The effect of numerical errors inside math library has not been addressed seriously.

The variance arithmetic should be used, whose values largely reproduce the corresponding results using conventional floating-point arithmetic, whose uncertainties trace all input errors, and whose result error deviations qualify the calculation quality as either ideal, or proper, or suspicious. If the library functions also have proper coverage, the calculation result will probably have proper coverage as well.

7 Comparison Using Matrix Inversion

7.1 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 (7.1) [12] and let the sub-determinant at index (i, j) be defined as Formula (7.2) [44]:

$$|M| \equiv \sum_{[p_1 \dots p_n]_n} \$[p_1 \dots p_n]_n \prod_k x_{k,p_k}; \tag{7.1}$$

$$|M|_{i,j} \equiv \sum_{[p_1 \dots p_n]_n}^{p_i = j} \$[p_1 \dots p_n]_n \prod_k^{k \neq i} x_{k,p_k}; \tag{7.2}$$

$(-1)^{i+j}|M_{(i,j)}|$ 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 (7.3) holds for the arbitrary row index i or the arbitrary column index j [12]:

$$|M| = \sum_{j=1}^{n} |M_{i,j}| x_{i,j} = \sum_{i=1}^{n} |M_{i,j}| x_{i,j}; \tag{7.3}$$

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 (7.3) can be applied to $M_{i,j}$, as:

$$|M_{<i_1,i_2>_n[j_1,j_2]_n}| \equiv \sum_{[p_1 \dots p_n]_n}^{p_{i_1}=j_1, p_{i_2}=j_2} \$[p_1 \dots p_n]_n \prod_k^{k \neq i_1, k \neq i_2} x_{k,p_k}; \tag{7.4}$$

$$|M| = \sum_{j_1} x_{i_1,j_1} |M_{i_1,j_1}| = \sum_{j_1} \sum_{j_2}^{i_2 \neq i_1, j_2 \neq j_1} x_{i_1,j_1} x_{i_2,j_2} |M_{<i_1,i_2>_n[j_1,j_2]_n}|; \tag{7.5}$$

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 (7.6).

$$||M_{<i_1,i_2>_n[j_1,j_2]_n}|| = ||M|_{<i_1,i_2>_n[j_2,j_1]_n}||; \tag{7.6}$$

The definition of a sub-determinant can be extended to Formula (7.7), in which $m \in \{1, 2 \dots n\}$. Formula (7.5) can be generalized as Formula (7.8), in which $m \in \{1, 2 \dots n\}$ and $< i_1 \dots i_m >_n$ is an arbitrary ordered permutation. Formula (7.8) can be viewed as the extension for both Formula (7.3) and Formula (7.1).

$$|M_{<i_1 \dots i_m>_n[j_1 \dots j_m]_n}| \equiv \sum_{[p_1 \dots p_n]_n}^{p_{i_k}=j_k, k \in \{1 \dots m\}} \$[p_1 \dots p_n]_n \prod_{k \in \{1 \dots n\}}^{k \notin \{i_1 \dots i_m\}} x_{k,p_k}; \tag{7.7}$$

$$|M| = \sum_{[j_1 \dots j_m]_n} |M_{<i_1 \dots i_m>_n[j_1 \dots j_m]_n}| \prod_{k=1}^{m} x_{i_k,j_k}; \tag{7.8}$$

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 (7.8) results in Formula (7.9), which results in Formula (7.10) after applying Formula (**??**):

$$\widetilde{|M|} - |M| = \sum_{m=1}^{n} \sum_{<i_1 \dots i_m>_n} \sum_{[j_1 \dots j_m]_n} |M_{<i_1 \dots i_m>_n [j_1 \dots j_m]_n}| \prod_{k=1}^{m} \widetilde{y}_{i_k,j_k}; \tag{7.9}$$

$$\delta|M|^2 = \sum_{m=1}^{n} \sum_{<i_1 \dots i_m>_n} \sum_{[j_1 \dots j_m]_n} |M_{<i_1 \dots i_m>_n [j_1 \dots j_m]_n}|^2 \prod_{k=1}^{m} \delta x_{i_k,j_k}^2; \tag{7.10}$$

Defining $|M_{<>_n<>_n}| \equiv |M|$, Formula (7.11) is an recursive form of Formula (7.10):

$$\delta|M_{<p_1 \dots p_k>_n<q_1 \dots q_k>_n}|^2 = \sum_{p_i} \sum_{q_j} \delta x_{p_i,q_j}^2$$
$$(|M_{<p_1 \dots p_i \dots p_k>_n<q_1 \dots q_j \dots q_k>_n}|^2 + \delta|M_{<p_1 \dots p_i \dots p_k>_n<q_1 \dots q_j \dots q_k>_n}|^2); \tag{7.11}$$

When using Formula (7.3) 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 (7.11) shows that the result uncertainty is the aggregation of uncertainties from all possible path of Formula (7.3). To accounts for all such uncertainties, Formula (7.11) 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]:

$$z_{i,j} = \frac{|M_{j,i}|}{|M|}; \tag{7.12}$$

Formula (7.12) 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].

7.2 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 (7.12). 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.

7.3 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 (7.1) 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 (7.1) 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 (7.12). This hypothesis is confirmed by Figure 64, 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 65 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 (7.12) 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 66 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 (7.10) contains the result sensitivity to any matrix element, any combination of matrix elements, as well as the aggregated result uncertainty deviation. Therefore, Formula (7.10) and Formula (7.11) may be better than the condition numbers for describing matrix stability.

7.4 Testing Uncertainty Propagation in Adjugate Matrix

When the adjugate matrix is calculated using variance arithmetic, Figure 67 shows that the average output deviations for the adjugate matrix increase linearly with the input deviation, which is in good agreement with Formula (**??**). Such relation is also true for maximal and average output values errors. Formula (**??**) 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 67 suggests that L increases with N^2 for the average output precision and average output error[8].

Figure 68 shows that the average output tracking ratio of the adjugate matrix using variance arithmetic is approximately a constant of 0.8. Figure 68 is very similar to Figure 55. Similar to the maximal output bounding ratios of FFT algorithms, the maximal output bounding ratios for the adjugate matrix using precision also obey Formula (**??**) 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 69, which is very similar to Figure 51. 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.

7.5 Calibration

Because $|M_{j,i}|$ and $|M|$ are not independent of each other, M^{-1} calculated by Formula (7.12) contains the dependency problem. Figure 69 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 69 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 70. Figure 71 shows that average tracking ratios

[8]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 (**??**) [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 (7.11) 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 70.

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.

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 (2.21) or Formula (2.47) is no longer subjective. Formula (**??**) is decomposed into the contribution of each successive term for Tylor expansion, as Formula (8.1):

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

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 72 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 72 repeats the above process with 4-bit calculated inside uncertainty, which only differs from Figure 72 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 74 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 75:

- 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]	$1.32\text{x}10^{-14}$	$-0.705\text{x}10^{-14}$
3	[25, 47]	$2.52\text{x}10^{-14}$	$-1.42\text{x}10^{-14}$
4	[26, 47]	$1.16\text{x}10^{-13}$	$-1.13\text{x}10^{-13}$
5	[26, 48]	$5.08\text{x}10^{-13}$	$-6.82\text{x}10^{-13}$
6	[26, 48]	$1.92\text{x}10^{-12}$	$-2.72\text{x}10^{-12}$

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 76, 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 77 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 78 shows the output deviation and the value errors vs. time while Figure 79 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 78 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 79.

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 78. This mechanism of error tracking in variance arithmetic is also demonstrated in Figure 80 and Figure 81. Figure 80 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 80 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 81 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 81, revealing similar underline error-tracking mechanism in both cases. Figure 81 also shows that for the same bits calculated inside uncertainty, the average tracking ratios deviate from independence at exactly the same time. Figure 80 and Figure 81 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 82 and 84, the input deviation is also increased by 10-fold to simulate an increase in measured uncertainty.
- *Defective*: In Figure 83 and 85, 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 84 while they are not affected at all in Figure 85. 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 82 and Figure 86 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 88 shows that the result tracking ratios of the progressive linear regression is exponentially distributed, while Figure 89 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 (2.24) and its multi-dimension extensions such as Formula (**??**).

Formula (**??**) 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 (7.11), 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 (7.11), 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 (7.11), 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 (**??**) 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

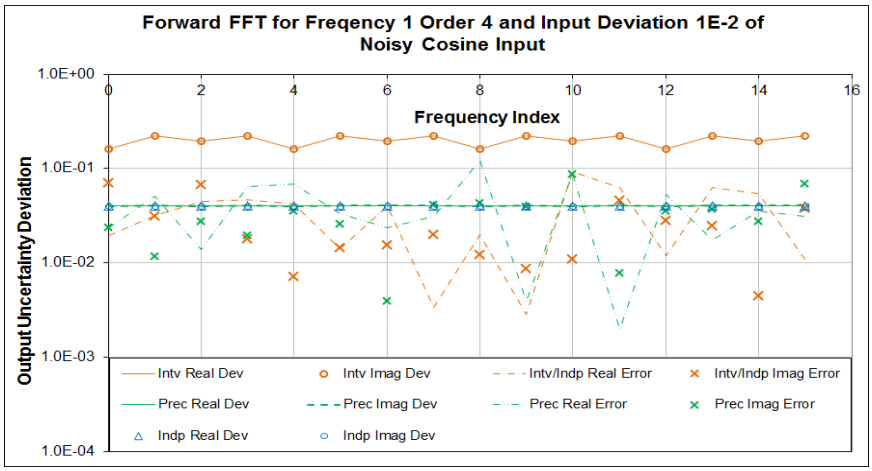

Figure 48: The output deviations and value errors of the forward FFT on a noisy sine signal of FFT order 4, index frequency 1 and input deviation 10^{-2}. In the legend, "Intv" means interval arithmetic, "Indp" means independence arithmetic, "Prec" means variance arithmetic, "Dev" means output uncertainty 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.

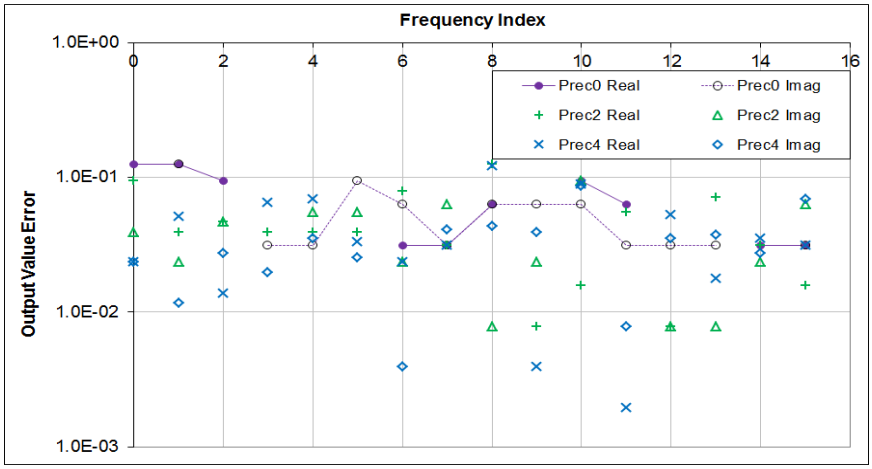

Figure 49: The output value errors of the forward FFT on a noisy sine signal of index frequency 1 and input deviation 10^{-2} using variance arithmetic with different bit inside uncertainty. In the legend, "Prec0" means variance arithmetic with 0-bit calculated inside uncertainty, "Prec2" means variance arithmetic with 2-bit calculated inside uncertainty, and "Prec4" means variance arithmetic with 4-bit calculated inside uncertainty.

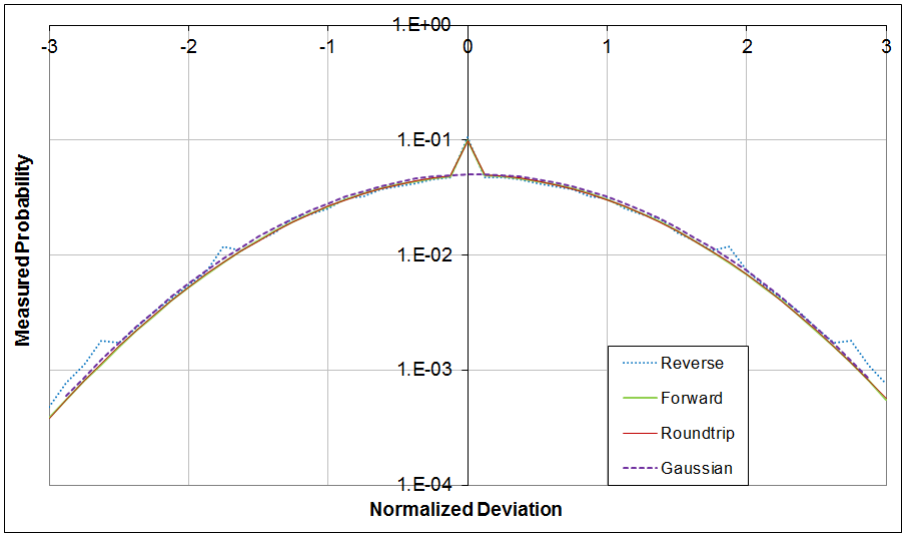

Figure 50: The measured tracking ratio distributions using independence arithmetic for FFT algorithms (as shown in legend). They are best fitted by a Gaussian distribution with the mean of 0.06 and deviation of 0.98.

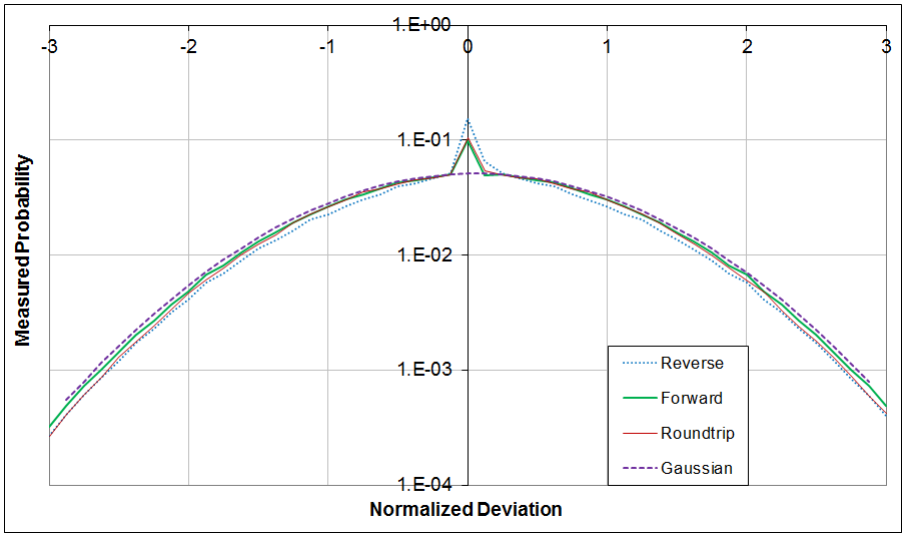

Figure 51: The measured tracking ratio distributions using variance arithmetic for FFT algorithms (as shown in legend). They are best fitted by a Gaussian distribution with the mean of 0.06 and deviation of 0.97.

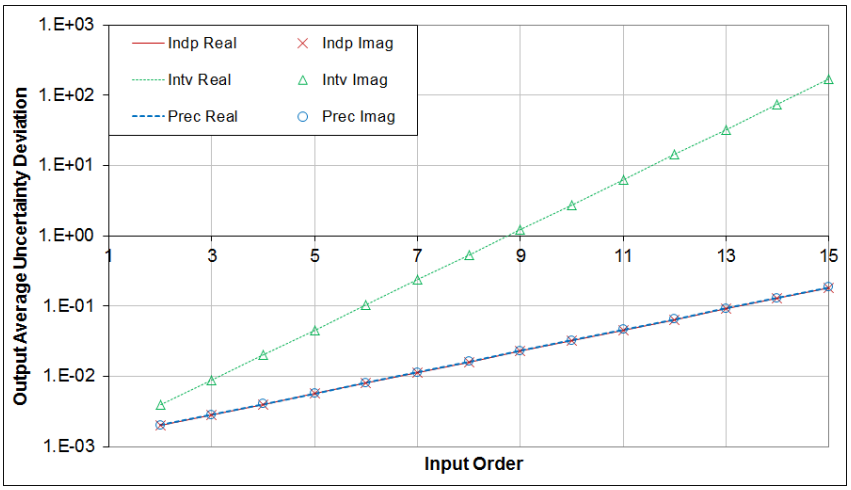

Figure 52: For the same input deviation of 10^{-3}, the empirical average output deviations of the forward FFT increase exponentially with the FFT order for all uncertainty-bearing arithmetics. In the legend, "Intv" means interval arithmetic, "Indp" means independence arithmetic, "Prec" means variance arithmetic, "Real" means real part, and "Imag" means imaginary part.

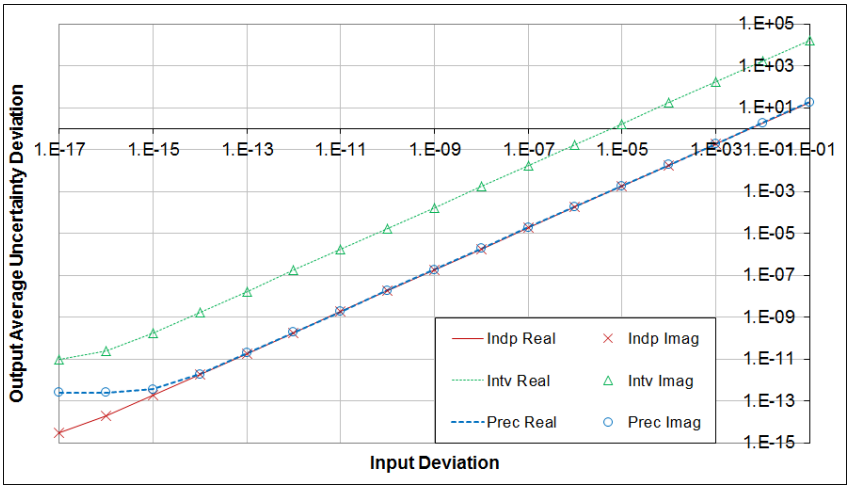

Figure 53: For the same order of the FFT calculation of 15, the empirical average output deviations of the forward FFT increases linearly with the input deviation for all uncertainty-bearing arithmetics. In the legend, "Intv" means interval arithmetic, "Indp" means independence arithmetic, "Prec" means variance arithmetic, "Real" means real part, and "Imag" means imaginary part.

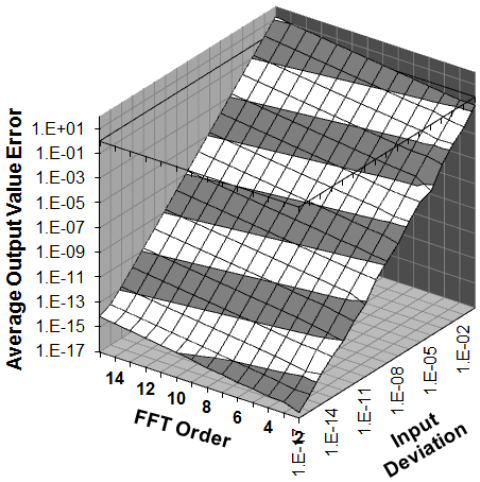

Figure 54: The empirical average output value errors using variance arithmetic increase exponentially with the FFT order and linearly with the input deviation, respectively.

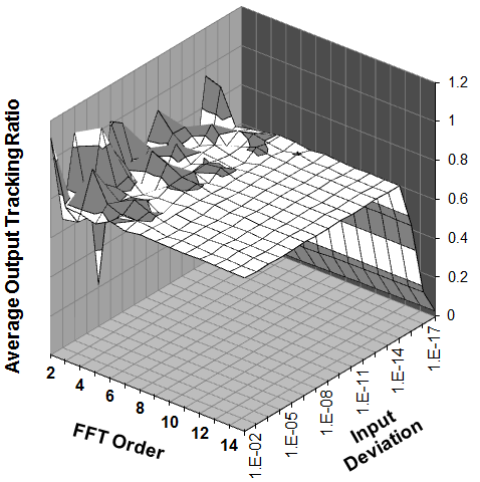

Figure 55: The empirical average output tracking ratios using variance arithmetic is a constant when the input deviation is larger than 10^{-14} and the FFT order is more than 5 for forward FFT algorithms. Because the precision of conventional floating-point representation is at 10^{-16}, adding Gaussian 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.

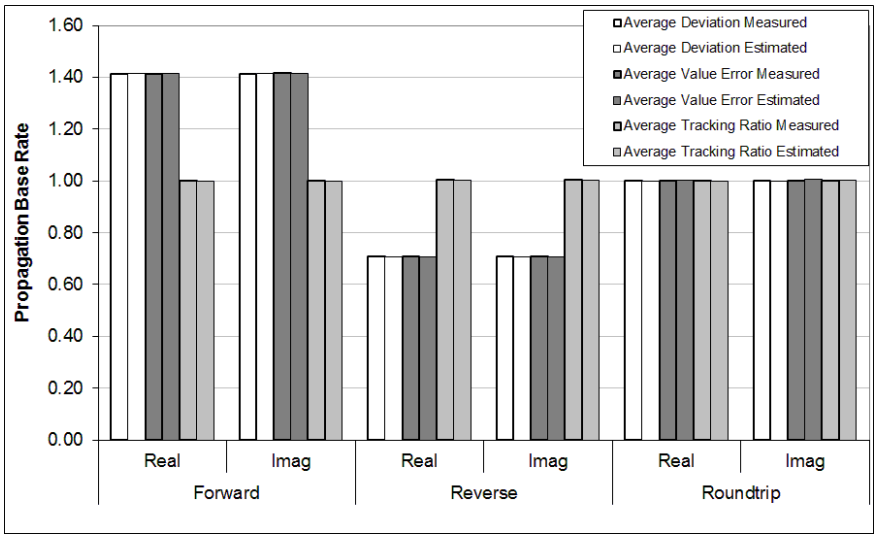

Figure 56: Empirical and theoretical β for fitting average output deviations, value errors and tracking ratios for forward, reverse and roundtrip FFT using independence arithmetic on noisy sine signals. In the chart, "Real" means real part, and "Imag" means imaginary part.

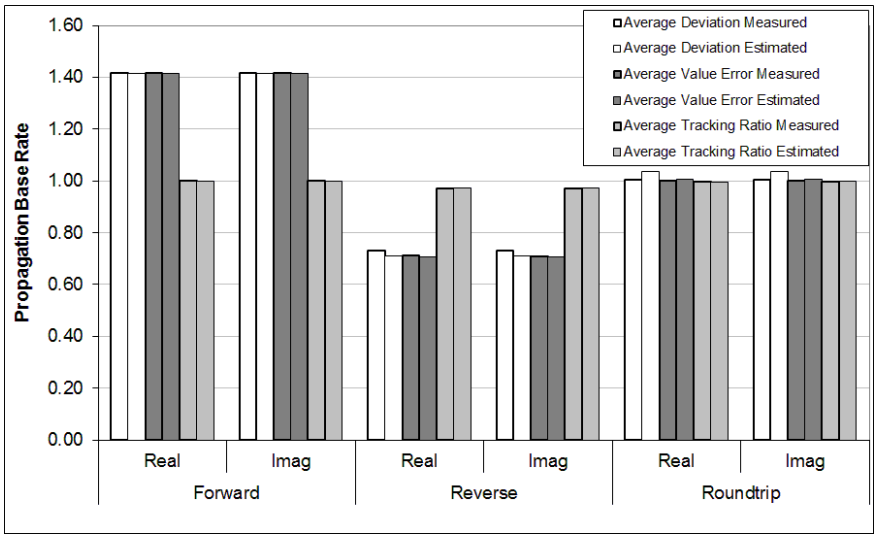

Figure 57: Empirical and theoretical β for fitting average output deviations, value errors and tracking ratios for forward, reverse and roundtrip FFT using variance arithmetic on noisy sine signals. In the chart, "Real" means real part, and "Imag" means imaginary part.

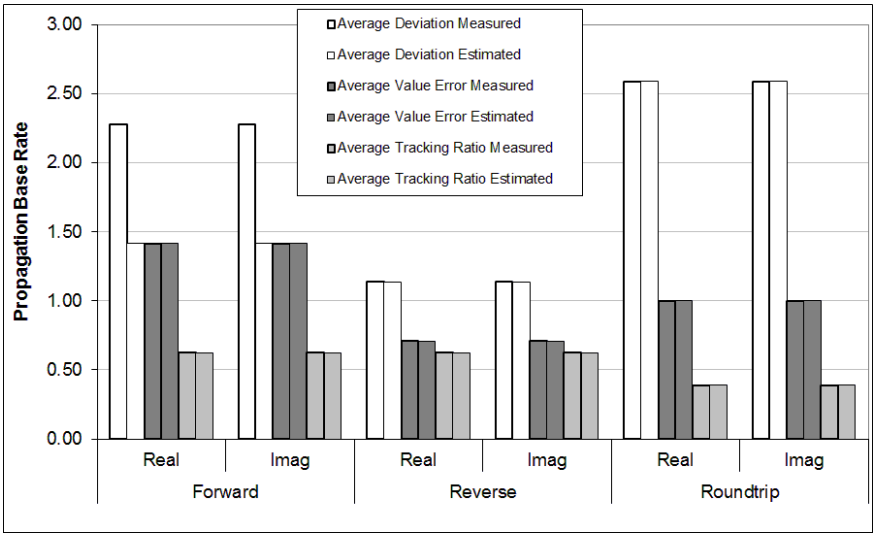

Figure 58: Empirical and theoretical β for fitting average output deviations, value errors and tracking ratios for forward, reverse and roundtrip FFT using interval arithmetic on noisy sine signals. In the chart, "Real" means real part, and "Imag" means imaginary part.

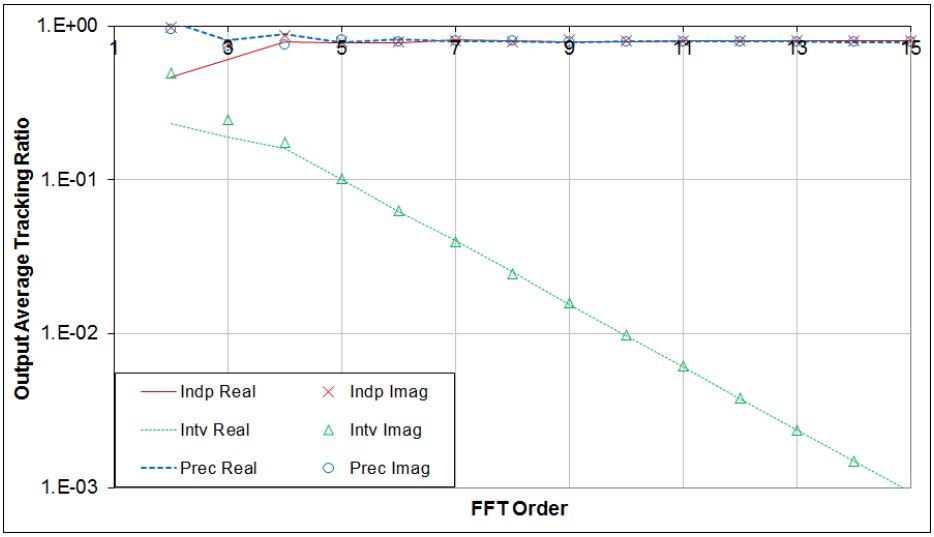

Figure 59: The empirical output average tracking ratios vs. the FFT order of the forward FFT for all three arithmetics when the input uncertainty deviation is 10^{-3}. In the legend, "Intv" means interval arithmetic, "Indp" means independence arithmetic, "Prec" means variance arithmetic, "Real" means real part, and "Imag" means imaginary part.

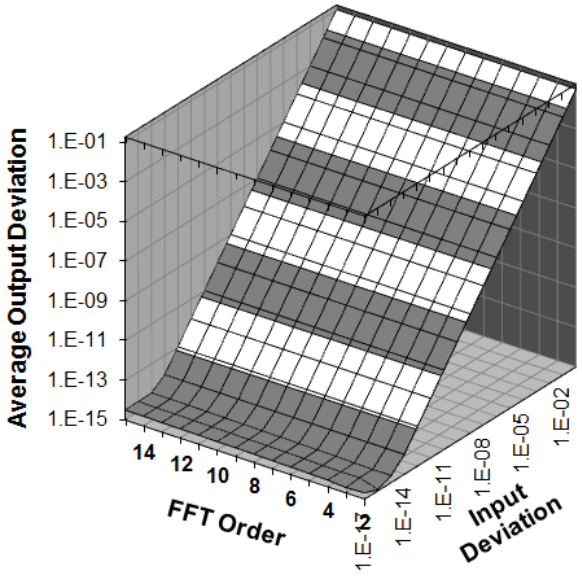

Figure 60: The empirical average output deviations vs. the FFT order and input deviations using variance arithmetic for the round-trip FFT algorithm.

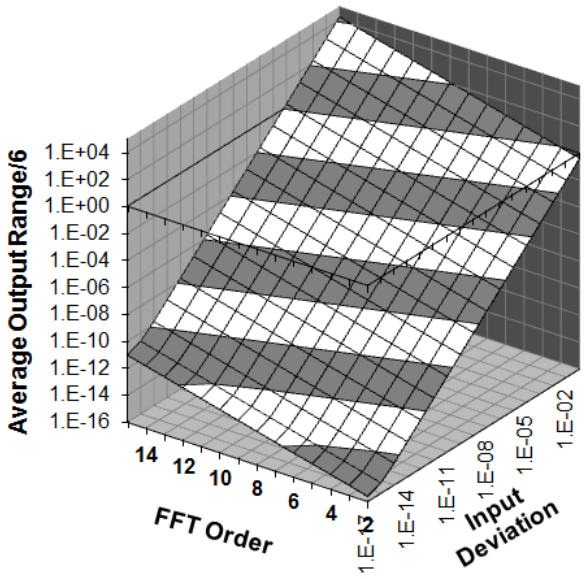

Figure 61: The empirical average output deviations vs. the FFT order and input deviations using interval arithmetic for the round-trip FFT algorithm.

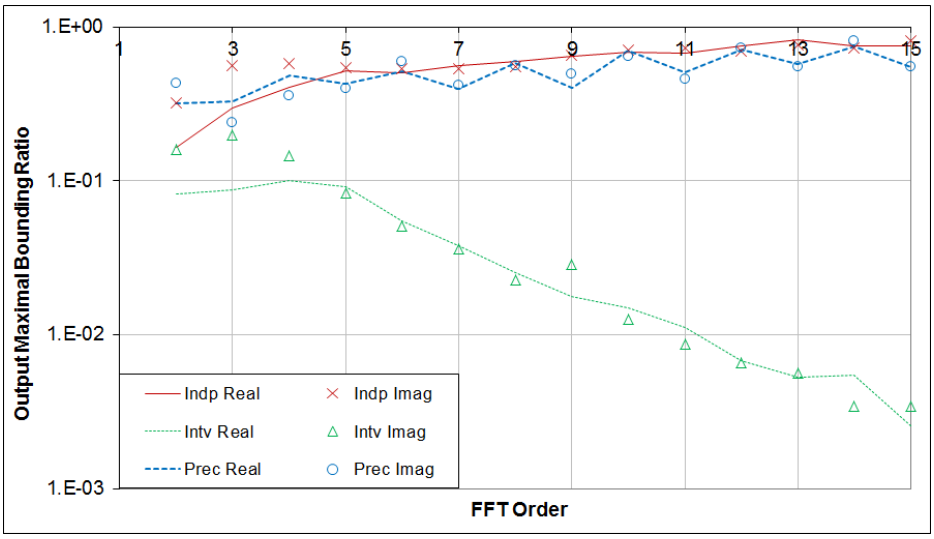

Figure 62: The empirical maximal output bounding ratios vs. the FFT order of the forward FFT for all three arithmetics. In the legend, "Intv" means interval arithmetic, "Indp" means independence arithmetic, "Prec" means variance arithmetic, "Real" means real part, and "Imag" means imaginary part.

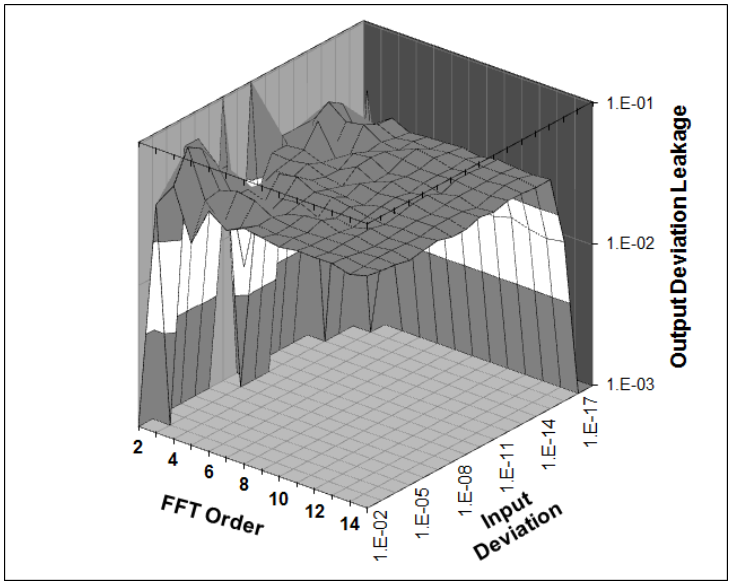

Figure 63: The empirical deviation leakages vs. the FFT order and input deviations using variance arithmetic for the forward FFT algorithm.

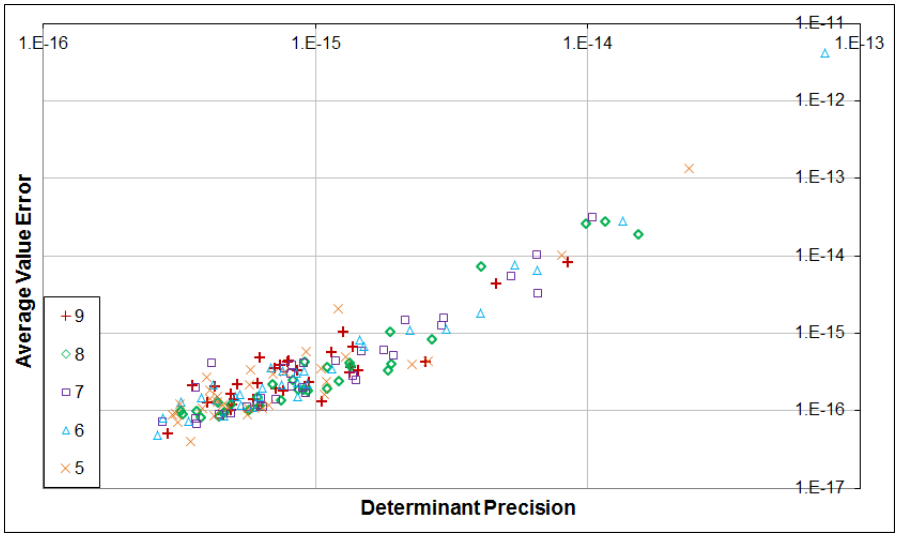

Figure 64: The empirical average value errors of the inverted matrix using conventional floating-point arithmetic vs. matrix determinant precision using variance arithmetic for clean matrices of different sizes (as shown in legend).

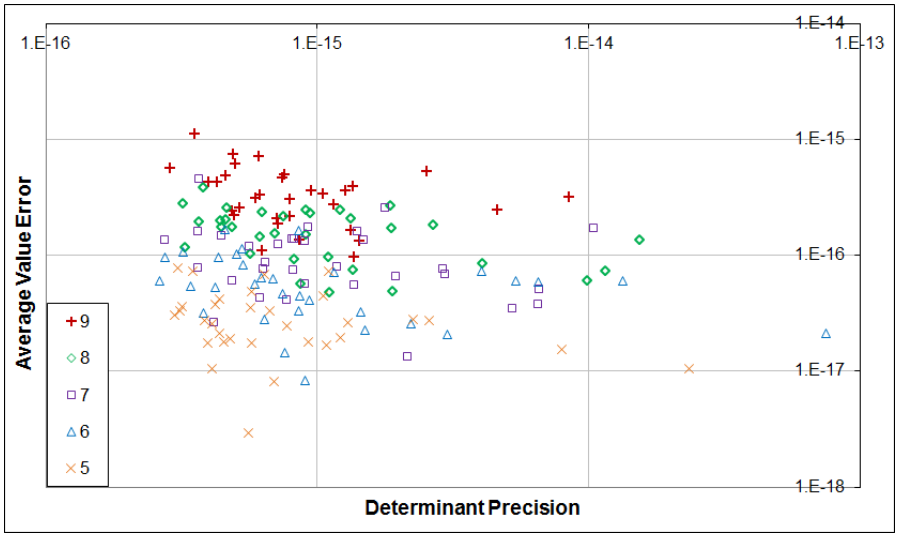

Figure 65: The empirical average value errors of the adjugate matrix using conventional floating-point arithmetic vs. matrix determinant precision using variance arithmetic for clean matrices of different sizes (as shown in legend).

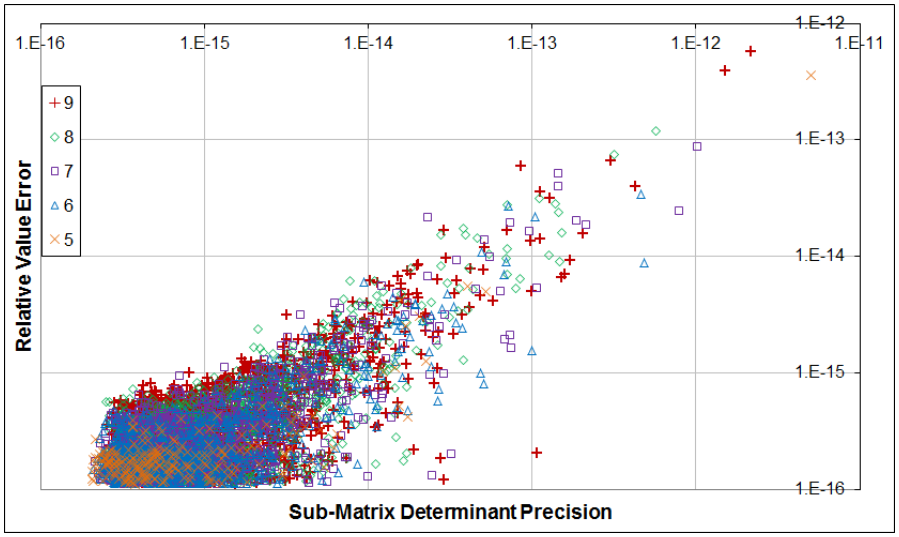

Figure 66: Empirical relative value errors of the adjugate matrix using conventional floating-point arithmetic vs. corresponding sub-matrix determinant precision using variance arithmetic for clean matrices of different sizes (as shown in legend).

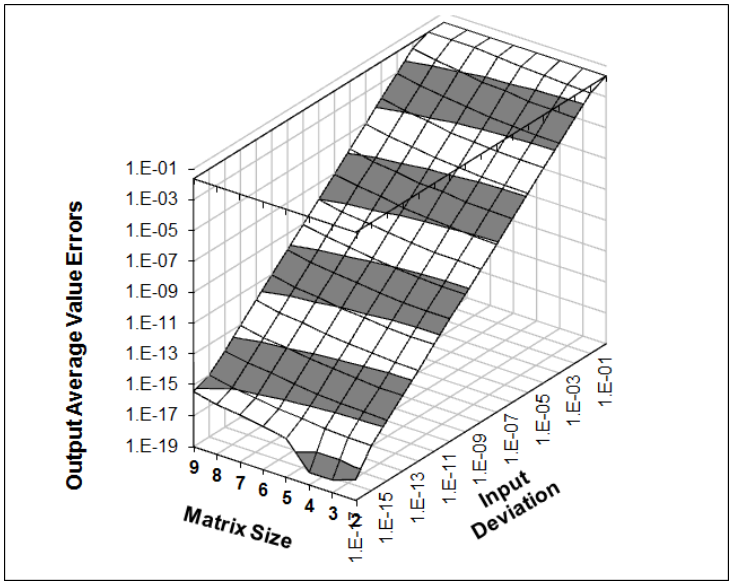

Figure 67: Using variance arithmetic, the average output deviations of the adjugate matrix vs. input precision and the matrix size.

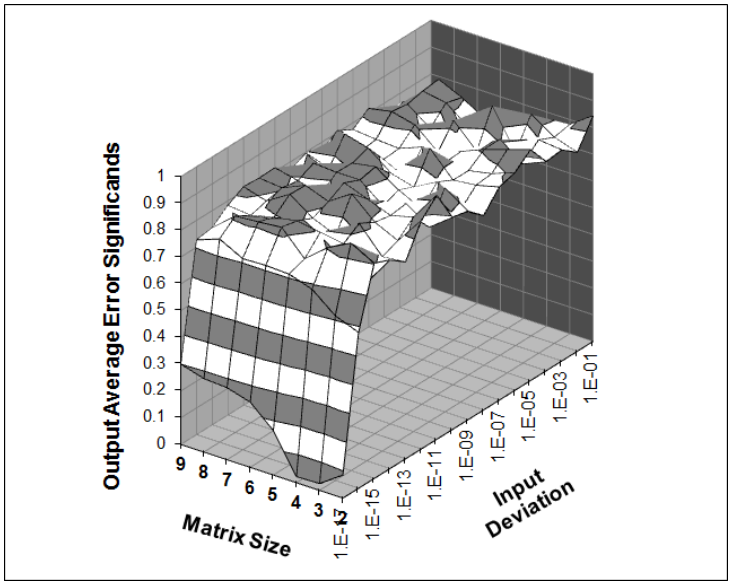

Figure 68: Using variance arithmetic, the average output tracking ratios of the adjugate matrix vs. input precision and the matrix size.

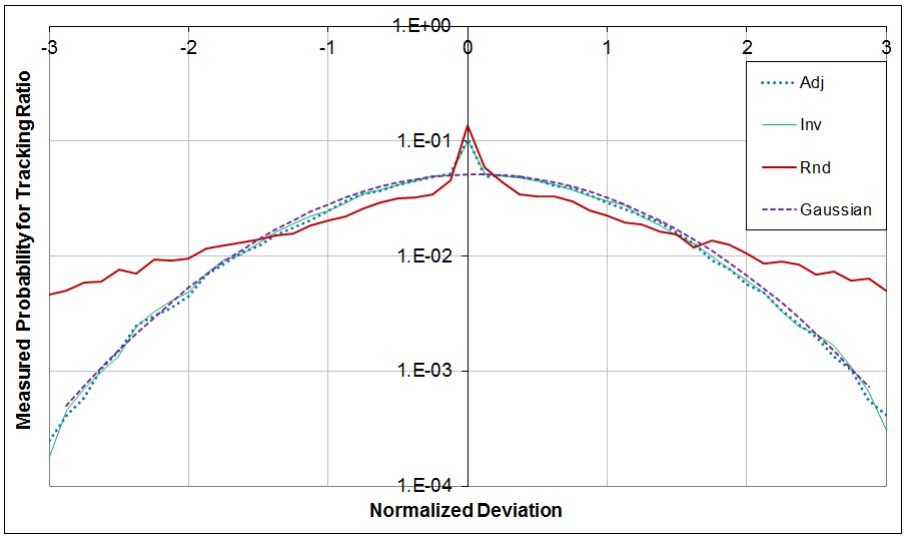

Figure 69: The measured tracking ratio distributions using variance arithmetic for matrix calculations of matrix size 9. They are best fitted by a Gaussian distribution with the mean of 0.06 and deviation of 0.96. In the legend, "Adj" means calculating adjugate M^A, "Inv" means calculating inverted M^{-1}, and "Rnd" means calculating double inverted $(M^{-1})^{-1}$.

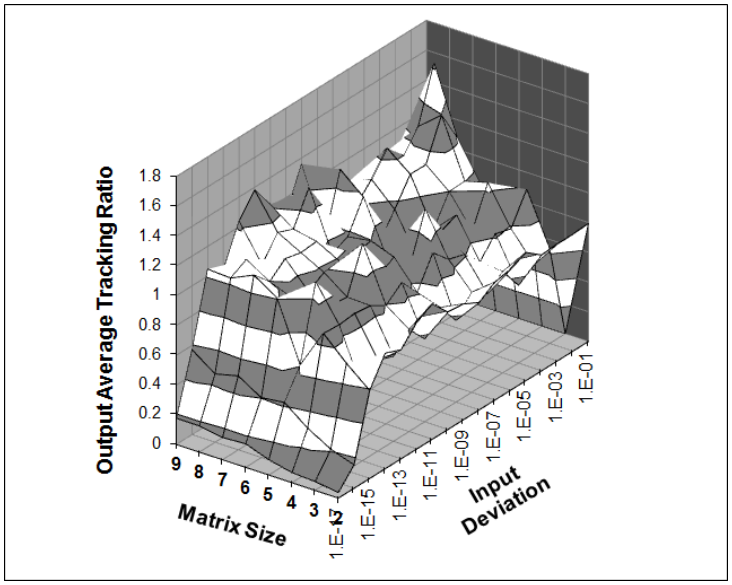

Figure 70: Using variance arithmetic, the average output tracking ratios of the double inverted matrix vs. input precision and the matrix size.

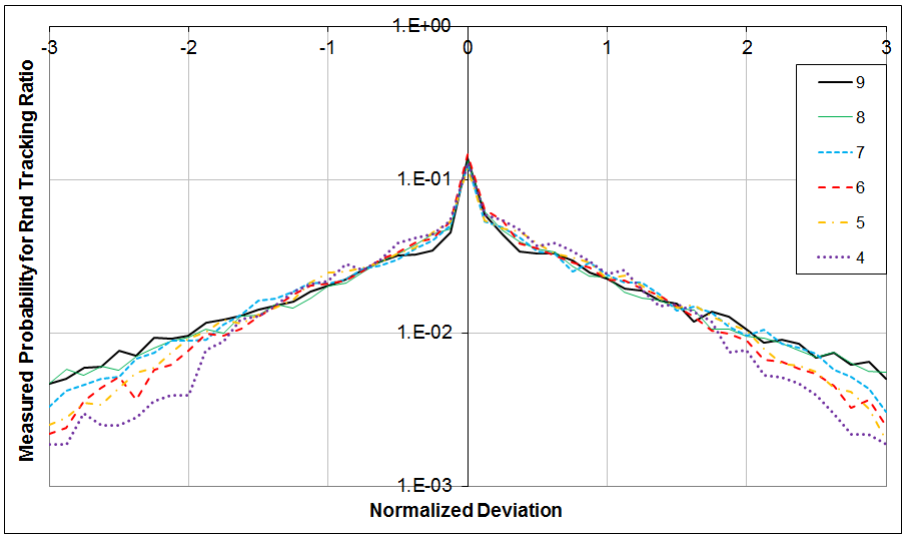

Figure 71: The measured tracking ratio distributions using variance arithmetic for matrix calculations of matrix size 9. They are best fitted by a Gaussian distribution with the mean of 0.06 and deviation of 0.96. In the legend, "Adj" means calculating adjugate M^A, "Inv" means calculating inverted M^{-1}, and "Rnd" means calculating double inverted $(M^{-1})^{-1}$.

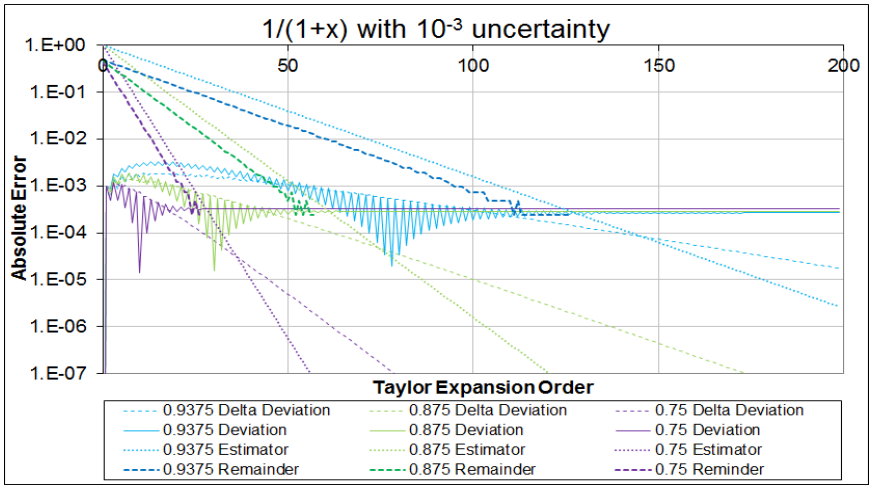

Figure 72: The delta deviation, deviations, Cauchy estimator and remainders of a Taylor expansion vs. the expansion orders for different input value with 10^{-3} input uncertainty using variance arithmetic with 0-bit calculated inside uncertainty. Different inputs are displayed using different color.

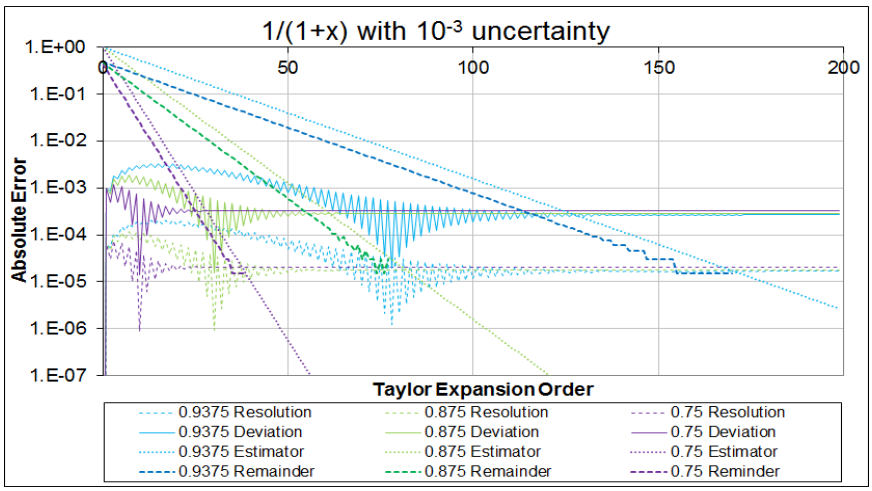

Figure 73: The deviations, resolutions, Cauchy estimator and remainders of a Taylor expansion vs. the expansion orders for different input value with 10^{-3} input uncertainty using variance arithmetic with 4-bit calculated inside uncertainty. Different inputs are displayed using different color.

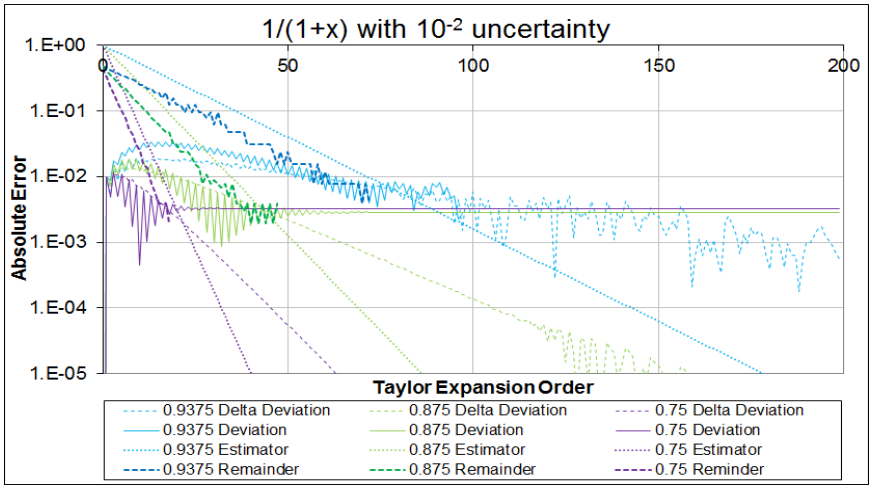

Figure 74: The delta deviation, deviations, Cauchy estimator and remainders of a Taylor expansion vs. the expansion orders for different input value with 10^{-2} input uncertainty using variance arithmetic with 0-bit calculated inside uncertainty. Different inputs are displayed using different color.

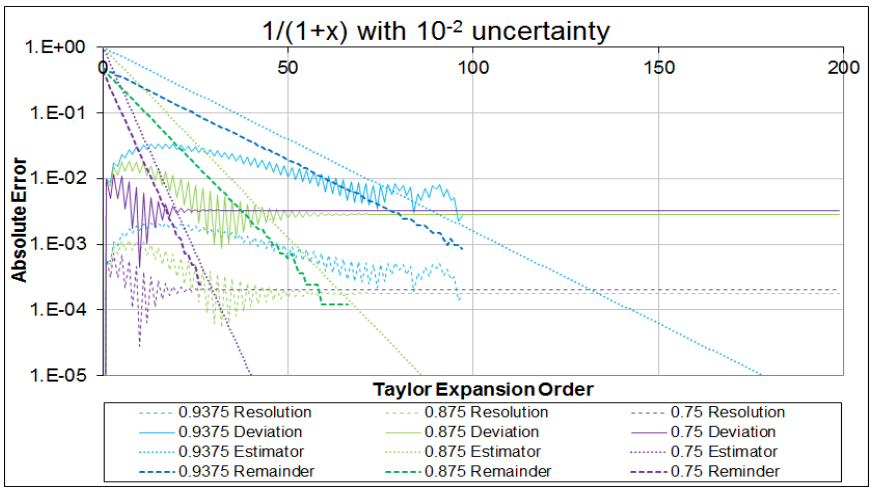

Figure 75: The deviations, resolutions, Cauchy estimator and remainders of a Taylor expansion vs. the expansion orders for different input value with 10^{-2} input uncertainty using variance arithmetic with 4-bit calculated inside uncertainty. Different inputs are displayed using different color.

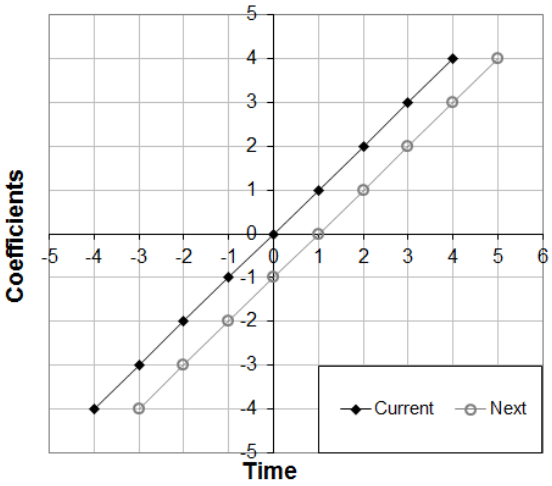

Figure 76: Coefficients of X in (10.2) at current and next position in a time series of the least square linear regression. Except the two end points at $X = -H$ and $X = H + 1$, respectively, the coefficient difference between the current and then next position in a time series are all by 1 in the overlapping region from $X = -H + 1$ to $X = H$, which results in (10.3).

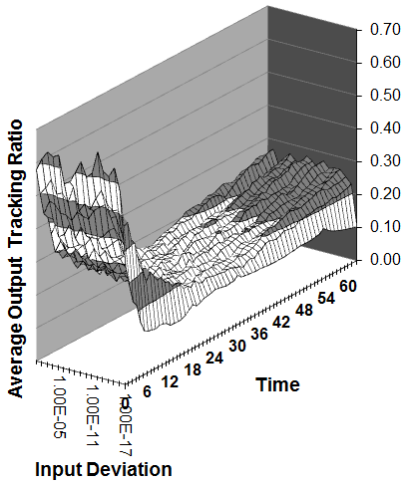

Figure 77: The average tracking ratio vs. time and the input uncertainty deviations for the progressive moving-window linear regression of a straight line using variance arithmetic with 4-bit calculated inside uncertainty.

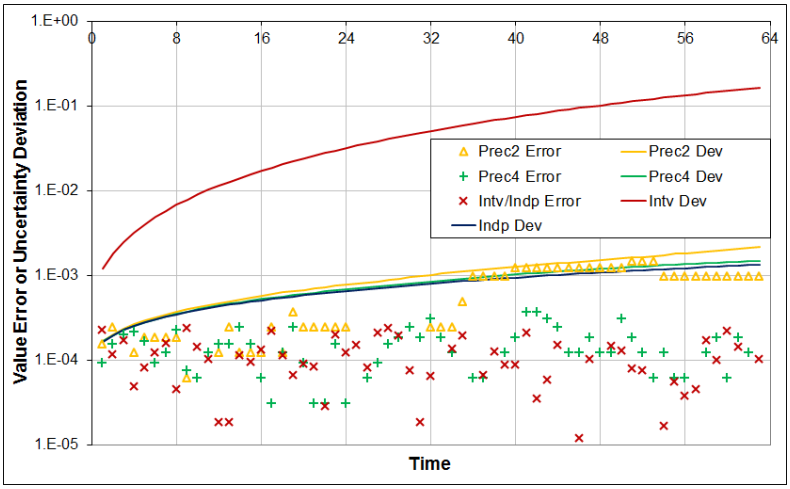

Figure 78: The output uncertainty deviations and the value errors vs. time for the progressive moving-window linear regression of a straight line. In the legend, "Indp" means independent arithmetic, "Intv" means interval arithmetic, "Prec4" and "Prec2" means the variance arithmetic with 4-bit and 2-bit calculated inside uncertainty, respectively.

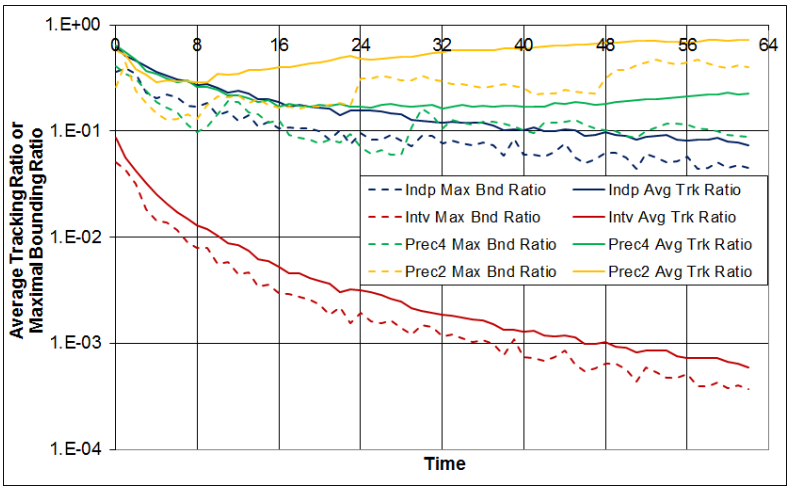

Figure 79: The average tracking ratios and the max bounding ratios vs. time for the progressive moving-window linear regression of a straight line. In the legend, "Indp" means independence arithmetic, "Intv" means interval arithmetic, "Prec4" and "Prec2" means the variance arithmetic with 4-bit and 2-bit 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.

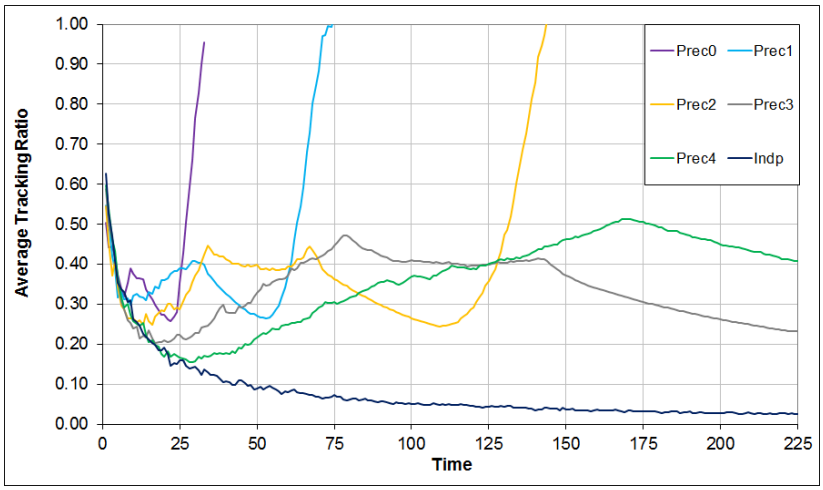

Figure 80: The average tracking ratios vs. time and the bits calculated inside uncertainty using variance arithmetic for the progressive moving-window linear regression of a straight line. In the legend, "Indp" means independence arithmetic, "PrecX" means the variance arithmetic with X-bit calculated inside uncertainty.

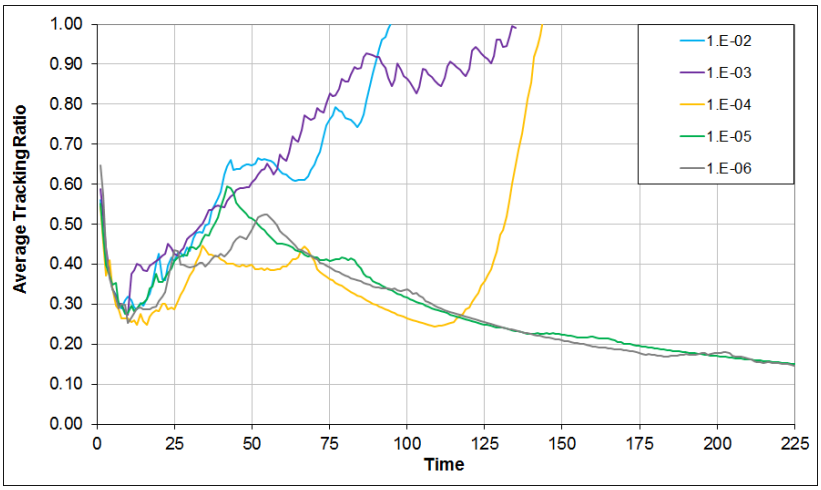

Figure 81: The average tracking ratios vs. time and the input precision using variance arithmetic with 2-bit calculated inside uncertainty for the progressive moving-window linear regression of a straight line for different input uncertainty deviations.

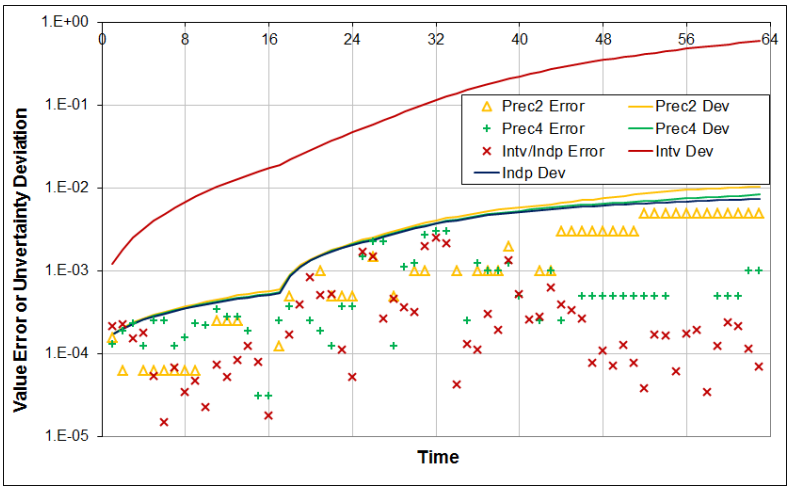

Figure 82: The output deviations and the value errors vs. time for the progressive moving-window linear regression of a straight line with 10-fold increase of both input noise and input uncertainty in the middle.

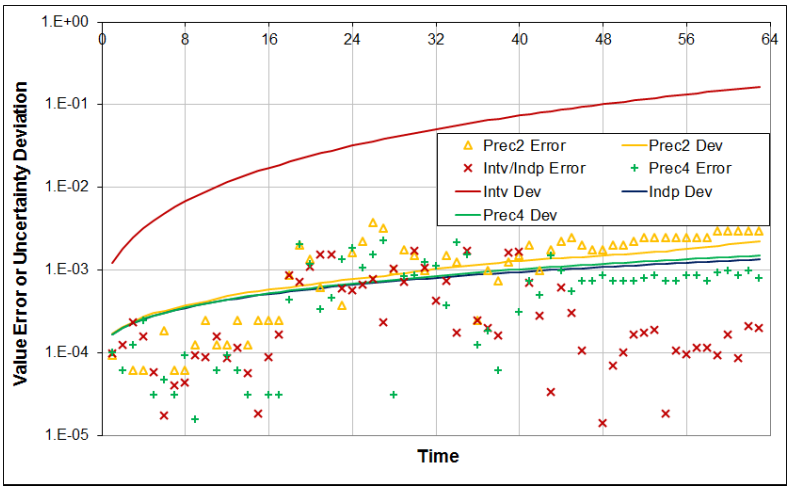

Figure 83: The output deviations and the value errors vs. time for the progressive moving-window linear regression of a straight line with only 10-fold increase of both input noise in the middle.

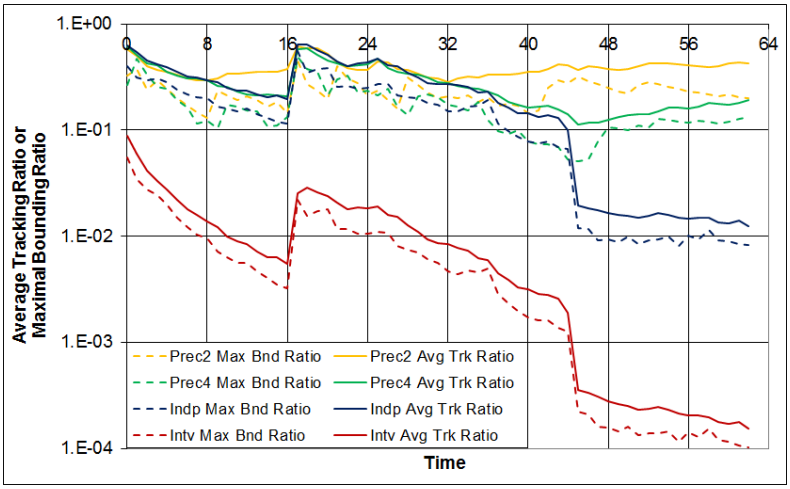

Figure 84: The average tracking ratios and the max bounding ratio vs. time for the progressive moving-window linear regression of a straight line with 10-fold larger input noise and deviation in the middle, to simulate larger noise following the straight line.

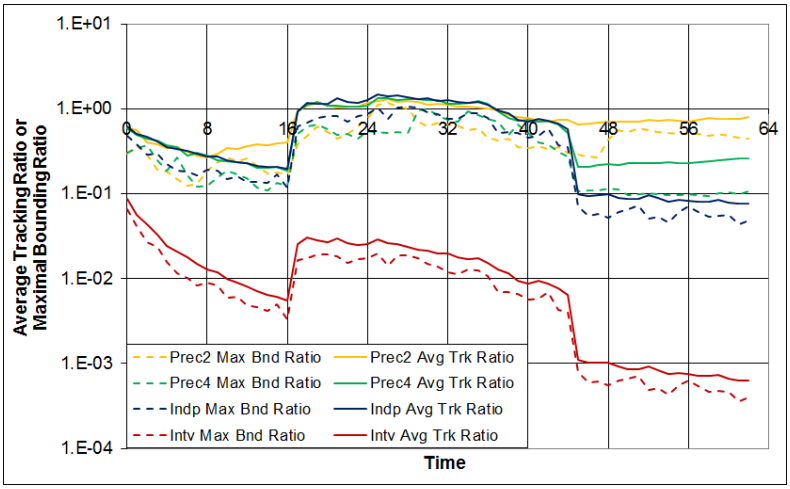

Figure 85: The average tracking ratios and the max bounding ratio vs. time for the progressive moving-window linear regression of a straight line with 10-fold larger input noise but same input deviation in middle, to simulate defects in obtaining the corresponding uncertainty deviations.

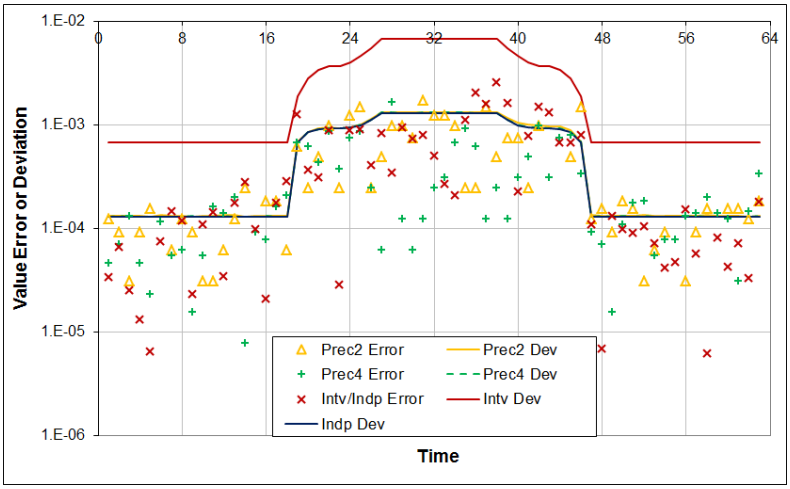

Figure 86: The output deviations and the value errors vs. time for the expressive moving-window linear regression of a straight line with 10-fold increase of both input noise and input uncertainty in the middle using variance arithmetic with 4-bit calculated inside uncertainty.

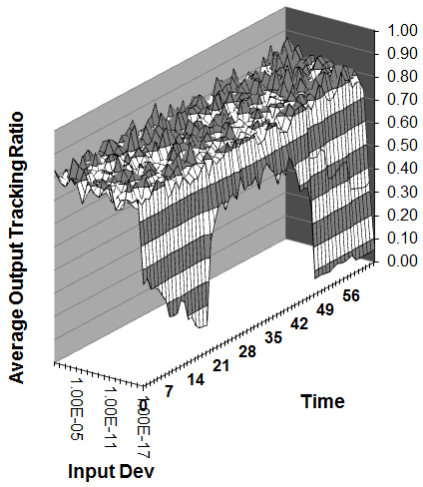

Figure 87: The average tracking ratio vs. time and the input uncertainty deviations for the expressive moving-window linear regression of a straight line with 10-fold increase of both input noise and input uncertainty in the middle using variance arithmetic with 4-bit calculated inside uncertainty.

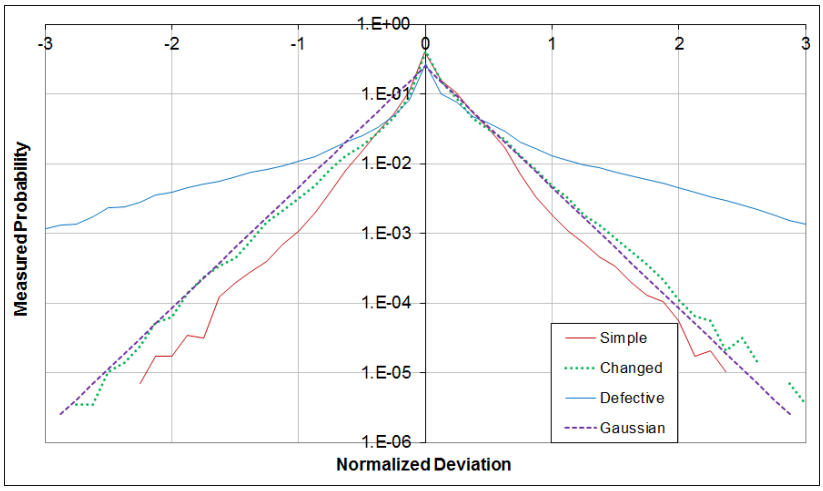

Figure 88: The measured tracking ratio distributions of the progressive moving-window linear regression for different cases (as shown in legend) using variance arithmetic with 4-bit calculated inside uncertainty. The case of "Changed" is best fitted by a exponential distribution with the mean of 0 and deviation of 0.25.

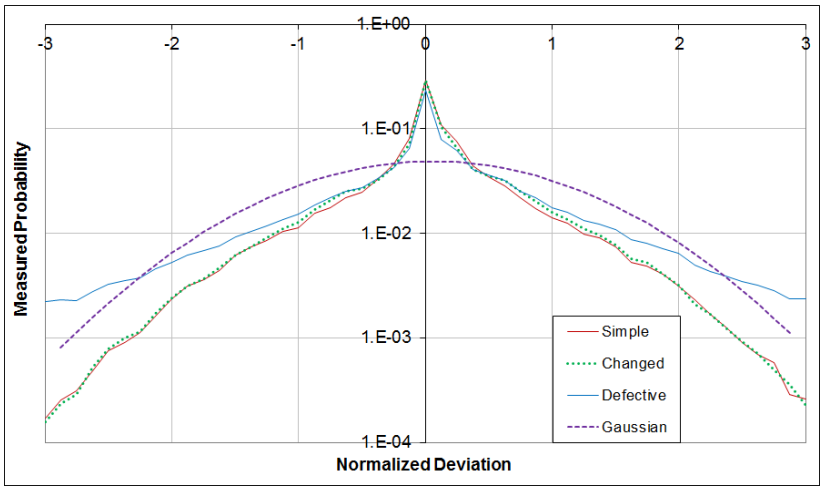

Figure 89: The measured tracking ratio distributions of the expressive moving-window linear regression using Formula (10.3) for different cases (as shown in legend) using variance arithmetic with 4-bit calculated inside uncertainty. The cases of "Simple" and "Changed" are best fitted by a Gaussian distribution with the mean of 0.06 and deviation of 0.97.