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 predict the result uncertainty variance for analytic functions, or for numerical functions with known imprecise derivatives. It uses Taylor expansion to predict the result value bias and uncertainty variance due to input uncertainty variances, and such prediction is precise if the input uncertainty variances are all precise. For floating-point rounding errors, it predicts the result uncertainty variance numerically within 100-fold of the actual result uncertainty variance. Regardless of input uncertainty distribution, if the expected result is a check of zero, the result uncertainty distribution is delta; otherwise, it is Gaussian due to central limit theorem.

The statistical requirement for the variance arithmetic is that any input uncertainty correlates with neither any input value nor any other input uncertainty. Variance arithmetic tracks the correlations of input within analytic expression using standard statistics. Due to this dependency tracing, variance arithmetic allows no numerical execution freedom, and it has no dependency on such freedoms. It rejects certain calculations such as inverting an input near zero on the mathematical ground of the divergence of the result uncertainty variance, which is also the statistical ground that the result value to be within the bounding range of ± 5-fold of the predicted uncertainty.

Variance arithmetic has been shown to be widely applicable, so as for analytic calculations, progressive calculation, regressive generalization, polynomial expansion, statistical sampling, and transformations.

It seems necessary to reexamine traditional numerical executions to adopt variance arithmetic. It seems necessary to recalculate all math library functions to attach the uncertainty variance to each corresponding result value.

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

*

Contents

1 Introduction

1.1 Measurement Uncertainty [38]

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 Conventional Floating-Point Arithmetic [38]

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. 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 favor 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. Each existing uncertainty-bearing arithmetic is reviewed below, *with the improvements by variance arithmetic pointed out.*

1.3 Interval Arithmetic [38]

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

[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].

to be outside a bounding range. If a bounding range is defined using a statistical rule on bounding leakage, such as the $6\sigma - 10^{-9}$ rule for Gaussian distribution [4] (which says that the bounding leakage is about 10^{-9} for a bounding range of mean $\pm$ 6-fold of standard deviations), there is no guarantee that the calculation result will also obey the $6\sigma - 10^{-9}$ rule using interval arithmetic, since interval arithmetic has no statistical foundation [3].

Another problem is that interval arithmetic only provides the worst case of uncertainty propagation, so that it tends to over-estimate uncertainty. 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 variance arithmetic, as show later in this paper.

As shown previously [38], 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, variance arithmetic provides stable bounding for a given bounding leakage, e.g., $\pm$5-fold of the result deviation

[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.

from the mean for a bounding leakage of 5.73×10^{-7}*. Variance arithmetic is based on statistics. It has no dependency problem.*

1.4 Statistical Propagation of Uncertainty [38]

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, variance arithmetic is based on a statistical assumption much more lenient than knowing the correlation between imprecise values.*

1.5 Significance Arithmetic [38]

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, 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 specify the uncertainty [38]. For example, if the least significand bit of significand is used to

specify uncertainty, then the representation has very coarse precision, such as $1 \pm 10^{-3} = 1024 \times 2^{-10}$ *[38]. Introducing limited bits calculated inside uncertainty cannot avoid this problem completely. Thus, the resolution of the conventional floating-point representation is desired. This is the reason why variance arithmetic has abandoned the significance arithmetic nature of its predecessor [38].*

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 [38]:

- The introduction section 1 remains almost unchanged, except it quotes the negative results from [38] for interval arithmetic and significance arithmetic.
- Variance arithmetic also has uncorrelated uncertainty assumption as its foundation. The theoretical foundation section 2 has the following significant differences:
    1. Variance arithmetic no longer requires the precision principle. Instead, it is a pure statistical approach.
    2. Variance arithmetic has different digital presentation and uses the standard floating-point arithmetic to calculate value and uncertainty, so that it is completely compatible with conventional floating-point arithmetic for precise floating-point values. It abandons the embedded significance arithmetic, and the rounding error bits to minimize rounding errors in the precision arithmetic.
    3. The Taylor expansion in variance arithmetic is centered around statistical means instead of the calculation results assuming no uncertainty, so variance arithmetic is strict statistically, and when the input uncertainties are specified precisely, the result uncertainties cover the result errors precisely.
    4. The theoretical convergence and statistical binding leakage of the result uncertainties of variance arithmetic are provided. Zeros and pole of the analytic functions for uncertainty calculations are identified. When the calculation is too close to a pole or zero, the result will diverge, such as an inversion near 0.
    5. Variance arithmetic tracks the correlations of input within analytic expression using standard statistics. Due to this dependency tracing, variance arithmetic allows no numerical execution freedom, and it has no dependency on such freedoms.
- Variance arithmetic improves validation methods greatly, to result in a new and much simpler analysis. It introduces ideal coverage and proper coverage, to quantify its applicable ranges and result reliability. The existence of ideal coverage is a necessary condition for the result to be statistically correct. External errors as a new import source of errors is identified. The sub section which categorizes algorithms is unchanged.
- Section 4 is completely different except for using the same function for Taylor expansion. It shows in detail how rounding errors are generated in the expansion process and why variance arithmetic can only provide proper coverage for rounding errors. It also presents a new criterion to truncate an expansion.

- Section 5 is completely different except for the deduction of the equations for the determinant variance. Adjugate matrices are analyzed using completely new and much simpler methods. For the stability of matrix inversion, the determinant precision is shown to be equivalent to the condition number of a matrix. The first order approximation for calculating matrix determinant is introduced.
- Section 6 is completely different, except for the deduction of the equations for the progressive linear fit. Previously, it showed the effect of embedded significance arithmetic. Now, with the embedded significance arithmetic removed, it shows the effect of having unspecified input uncertainty in addition to the new analysis.
- Section 7 is new, to apply variance arithmetic to the most common math library functions.
- Section 8 is completely different, provided the section for the modeling errors. It now focuses on the effect of numerical library errors and shows that such library errors can be very significant, especially when they resonate with input data.
- Section 9 is different by using new analysis.
- Section 10 is completely different. It provides a summary and some discussion for variance arithmetic.

2 Variance arithmetic

2.1 Foundation for 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 obey the *uncorrelated uncertainty assumption* [38], 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 obey the *uncorrelated uncertainty assumption*, in the intermediate steps, variance arithmetic uses statistics to account for the correlation between uncertainties through their associated variables. Therefore, variance arithmetic has no dependency problem.

2.2 The Uncorrelated Uncertainty Assumption [38]

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

One special application of Formula (2.2) is the correlation between a measured signal and its true signal, in which noise is the uncorrelated part between the two.

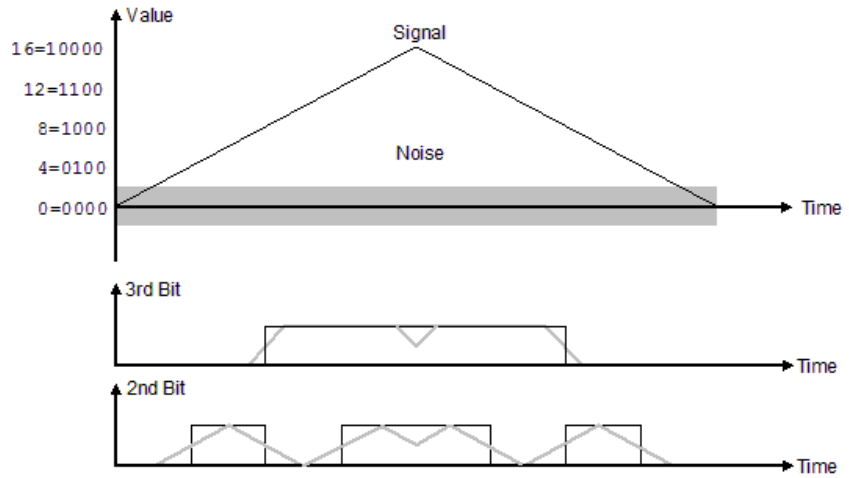

Figure 1: Effect of noise on bit values of a measured value. The triangular wave signal and the added white noise are shown at top using the thin black line and the grey area, respectively. The values are measured by a theoretical 4-bit Digital-to-Analog Converter in ideal condition, assuming LSB is 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 [38].

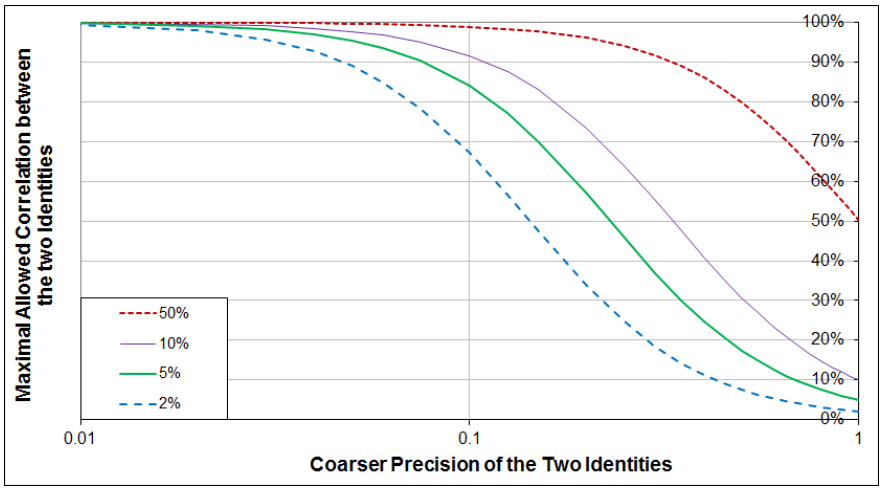

Figure 2: Allowed maximal correlation between two values vs. input precisions and independence standard (as shown in legend) for the independence uncertainty assumption of variance arithmetic to be true. This figure has been published previously [38].

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

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

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

Similarly, while the correlated portion between two values has 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 (2.2), if two significant values are overall correlated with γ, at the level of uncertainty the correlation between the two values decreases to γ_P according to Formula (2.4).

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

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

Figure 2 plots the relation of γ vs. P for each given γ_P in Formula (2.4). When γ_P is less than a predefined maximal threshold (e.g., 2%, 5% or 10%), the two values can be deemed uncorrelated to each other at the level of uncertainty. For each independence standard γ_P, there is a maximal allowed correlation between two values below which the uncorrelated uncertainty assumption of variance arithmetic holds. The maximal allowed correlation is a function of the larger precision of the two values according to Formula (2.4). Figure 2 shows that for two precisely measured values, their correlation γ is allowed to be quite high. Thus, the uncertainty assumption uncertainty assumption has much weaker statistical requirement than the assumption for independence arithmetic, which requires the two values to be independent of each other.

It is tempting to add noise to otherwise unqualified values to make them satisfy the uncertainties uncertainty assumption. As an extreme case of this approach, if two values are constructed by adding noise to the same signal, they are 50% correlated

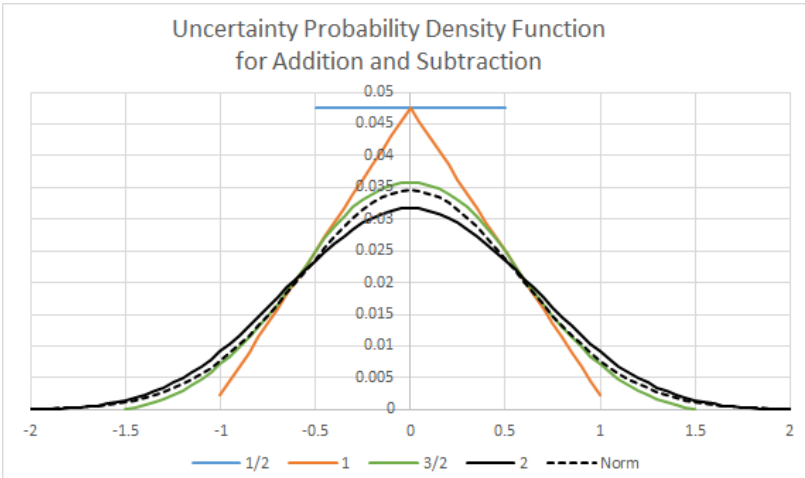

Figure 3: The probability distribution $P_{\frac{n}{2}}(x)$ of Formula (2.6), assuming $P_{\frac{1}{2}}(x)$ is uniformly distributed between $(-1/2, +1/2)$. The legend shows n. $P_2(x)$ is already close to the corresponding Gaussian distribution with the same mean and deviation, which is plotted in dash line of the same color. This figure has been published previously [38].

at the uncertainty level so that they will not satisfy the uncorrelated uncertainty assumption, which defines the upper bound for γ_P. [4]

2.3 Pure Rounding

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 [38], 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 Addition and Subtraction [38]

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].

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

[4]The 50% curve in Figure 2 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) gives 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 x and the variance $(\delta x)^2$ of an imprecise value $x \pm \delta x$. When the uncertainty δx is not specified initially:

- A conventional floating-point value is assumed to be imprecise in the least significant bit of its significand, and the error is assumed uniformly distributed within the bit. The equivalent value of the bit is defined as the *least significant value* [5], so that the uncertainty is $1/\sqrt{3}$-fold of the least significant value.
- 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.

The term least significant value can be abbreviated as LSV for simplicity.

[5] Together with least significant bit, it is a more indicative name for unit in the last place.

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 [6].

Otherwise, the standard z-method [4] is used to judge whether they are equal, 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 equivalent bounding range for z can be used, such as $|z| \leq 0.67448975$ for the equivalent threshold probability of 50%.

2.7 Multiplication

The result variance of multiplying $x \pm \delta x$ by a precise value y is $y^2\delta^2 x$ [4]. 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) [38].

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

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

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 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 (2.12) is the uncertainty distribution density function [1]. In Formula (2.12) [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 (2.12), Formula (2.13), (2.14), and (2.15) give the $\rho(\tilde{y}, y, \delta y)$ for e^x, $\ln(x)$, and x^c, respectively.

$$N(\tilde{z})d\tilde{z} = \rho(\tilde{x}, x, \delta x)d\tilde{x} = \rho(f^{-1}(\tilde{y}), x, \delta x)\frac{d\tilde{x}}{d\tilde{y}}d\tilde{y} = \rho(\tilde{y}, y, \delta y)d\tilde{y}; \tag{2.12}$$

$$y = e^x : \rho(\tilde{y}, y, \delta y) = \frac{1}{\tilde{y}}\frac{1}{\delta x}N(\frac{\log(\tilde{y}) - x}{\delta x}); \tag{2.13}$$

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

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

[6] Such definition of equality deviates from statistics. In statistics, such two imprecise values has 50% possibility to be less than or greater to each other but no chance to be equal to each other.

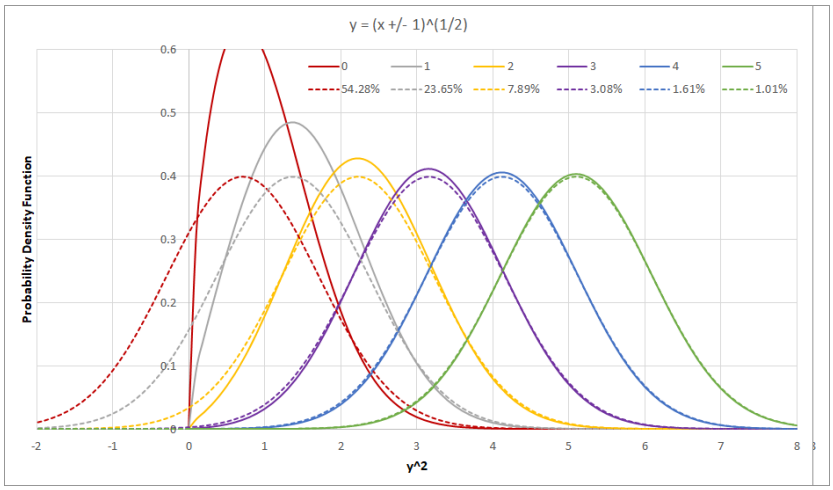

Figure 4: The probability density function for $\tilde{y} = \sqrt{x \pm 1}$, for different x as shown in the legend. The x-axis is scaled as $\tilde{y}^2$. Each probability density function $\rho(\tilde{x})$ in solid line is compared with a Gaussian distribution $\varrho(\tilde{x})$ of the same mode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

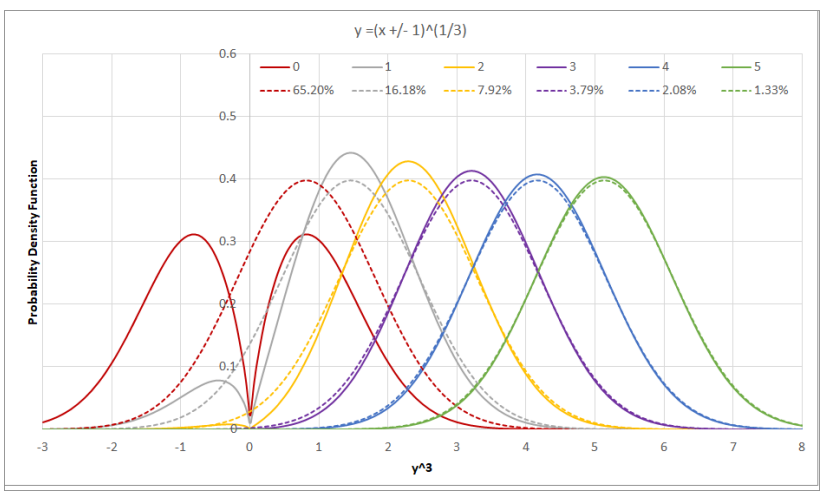

Figure 5: The probability density function for $\sqrt[3]{x \pm 1}$, for different x as shown in the legend. The y-axis is scaled as $\tilde{y}^3$. Each probability density function $\rho(\tilde{x})$ in solid line is compared with a Gaussian distribution $\varrho(\tilde{x})$ of the same mode and the same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

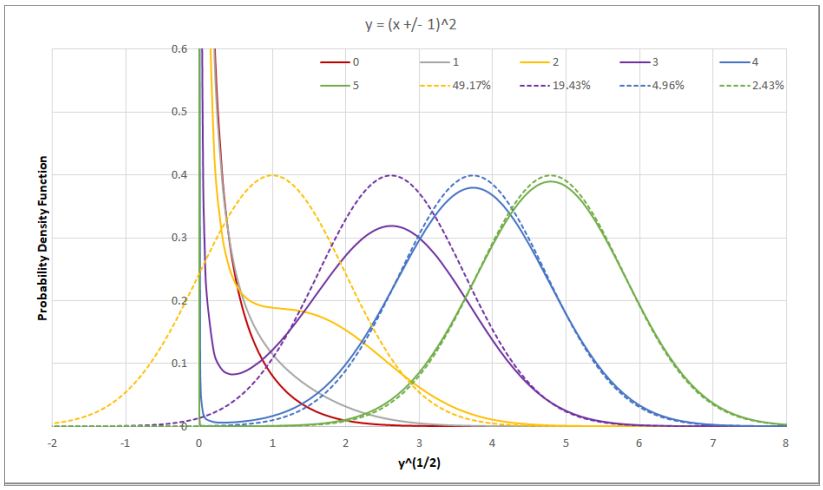

Figure 6: The probability density function for $(x \pm 1)^2$, for different x as shown in the legend. The y-axis is scaled as $\sqrt{\tilde{y}}$. Some probability density function $\rho(\tilde{x})$ in solid line is compared with a Gaussian distribution $\varrho(\tilde{x})$ of the same mode and the same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

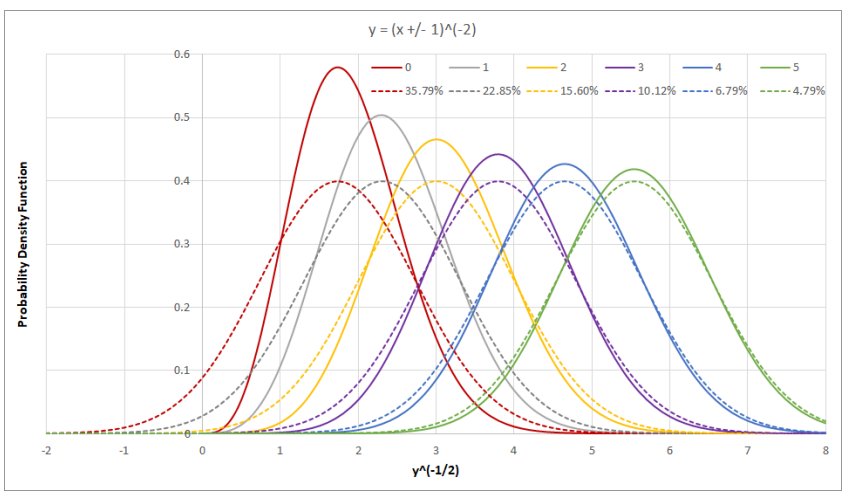

Figure 7: The probability density function for $1/(x \pm 1)^2$, for different x as shown in the legend. The y-axis is scaled as $1/\sqrt{\tilde{y}}$. The probability density function $\rho(\tilde{x})$ in solid line is compared with a Gaussian distribution $\varrho(\tilde{x})$ of the same mode and the 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 (2.12), Formula (2.16) 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 (2.17), (2.18), and (2.19) respectively.

$$\tilde{z} = \frac{f^{-1}(\tilde{y}) - x}{\delta x} : \quad \frac{d^2\tilde{z}}{d\tilde{y}^2} = \frac{d\tilde{z}}{d\tilde{y}}\tilde{z}; \tag{2.16}$$

$$\tilde{y} = e^{\tilde{x}} : \quad \ln(\tilde{y}_m) = x - (\delta x)^2; \tag{2.17}$$

$$\tilde{y} = \ln(\tilde{x}) : \quad e^{\tilde{y}_m} = \frac{x + \sqrt{x^2 + 4(\delta x)^2}}{2}; \tag{2.18}$$

$$\tilde{y} = \tilde{x}^c : \quad (\tilde{y}_m)^{1/c} = \frac{x + \sqrt{x^2 - 4(c-1)(\delta x)^2}}{2}; \tag{2.19}$$

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 perspective, a linear transformation generates neither zero nor pole according to Formula (2.12).

Figure 4 shows the probability density function for $\sqrt{x \pm 1}$ according to Formula (2.15), 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 (2.15), 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 (2.15), 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 (2.19), both $(0 \pm 1)^2$ and $(1 \pm 1)^2$ have no mode. $(0 \pm 1)^2$ is the χ^2 distribution [1].

Figure 7 shows the probability density function for $1/(x \pm 1)^2$ according to Formula (2.15), 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 (2.12) 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 precisely calculated in a range by the Taylor series as shown in Formula (2.20). Using Formula (2.9), Formula (2.22) and (2.23) [38] 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 (2.21), and σ is defined as the *bounding factor*.

$$\tilde{y} = f(x+\tilde{x}) = f(x+\tilde{z}\delta x) = \sum_{n=0}^{\infty} \frac{f_x^{(n)}}{n!} \tilde{z}^n (\delta x)^n; \tag{2.20}$$

$$\zeta(n) \equiv \int_{-\sigma}^{+\sigma} \tilde{z}^n N(\tilde{z}) d\tilde{z}; \quad \zeta(2n+1) = 0; \tag{2.21}$$

$$\overline{f(x)} = \int_{-\sigma}^{+\sigma} f(x+\tilde{z}\delta x) N(\tilde{z}) d\tilde{z} = f(x) + \sum_{n=1}^{\infty} (\delta x)^{2n} \frac{f_x^{(2n)} \zeta(2n)}{(2n)!}; \tag{2.22}$$

$$\delta^2 f(x) = \overline{f(x)^2} - \overline{f(x)}^2 = \sum_{n=0}^{\infty} (\delta x)^{2n} \tag{2.23}$$

$$\left(\zeta(2n) \sum_{j=1}^{2n-1} \frac{f_x^{(j)}}{j!} \frac{f_x^{(2n-j)}}{(2n-j)!} - \sum_{j=1}^{n-1} \frac{f_x^{(2j)} \zeta(2j)}{(2j)!} \frac{f_x^{(2n-2j)} \zeta(2n-2j)}{(2n-2j)!} \right);$$

$\overline{f(x)} - f(x)$ is defined as the *uncertainty bias* which is the effect of the input uncertainty to the result. When $f(x) \neq 0$, $\widehat{P(f(x))} \equiv \frac{\delta f(x)}{|f(x)|}$ is defined as the *normalized uncertainty*, which approximates the precision $P(f(x)) = \frac{\delta f(x)}{|\overline{f(x)}|}$ when the uncertainty bias is ignored.

2.9.2 Multiple Dimensional Expansion

Due to the uncorrelated uncertainty assumption, the Taylor expansion can be applied to multiple inputs, such as Formula (2.26) [38] .

$$f(x+\tilde{x}, y+\tilde{y}) = \sum_{m=0}^{\infty} \sum_{n=0}^{\infty} \frac{f_{(x,y)}^{(m,n)}}{m!n!} \tilde{x}^m \tilde{y}^n; \tag{2.24}$$

$$\overline{f(x,y)} = \int\int f(x+\tilde{x}, y+\tilde{y}) \rho(\tilde{x}, x, \delta x) \rho(\tilde{y}, y, \delta y)\, d\tilde{x} d\tilde{y} \tag{2.25}$$

$$= \sum_{m=0}^{\infty} \sum_{n=0}^{\infty} (\delta x)^{2m} (\delta y)^{2n} \frac{\zeta(2m)\zeta(2n) f_{(x,y)}^{(2m,2n)}}{(2m)!(2n)!};$$

$$\delta^2 f(x,y) = \overline{f(x,y)^2} - \overline{f(x,y)}^2 = \sum_{m=0}^{\infty} \sum_{n=0}^{\infty} (\delta x)^{2m} (\delta y)^{2n} \tag{2.26}$$

$$(\zeta(2m)\zeta(2n) \sum_{i=0}^{2m} \sum_{j=0}^{2n} \frac{f_{(x,y)}^{(i,j)}}{i!\, j!} \frac{f_{(x,y)}^{(2m-i,2n-j)}}{(2m-i)!\,(2n-j)!}$$

$$- \sum_{i=0}^{m} \sum_{j=0}^{n} \frac{\zeta(2i)\zeta(2j) f_{(x,y)}^{(2i,2j)}}{(2i)!(2j)!} \frac{\zeta(2m-2i)\zeta(2n-2j) f_{(x,y)}^{(2m-2i,2n-2j)}}{(2m-2i)!(2n-2j)!});$$

Formula (2.7) and (2.11) are two special cases of Formula (2.26). Although Formula (2.26) is only for 2-dimensional, it can be extended to any dimensional.

2.9.3 Bounding Factor and Bounding Leakage

The choice of the bounding factor σ needs careful considerations. If $\sigma = \infty$, $\zeta(2n) = (2n-1)!!$, which may cause Formula (2.23) to diverge in most cases. When σ is limited, Formula (2.21) becomes Formula (2.27):

- For small $2n$, $\zeta(2n)$ can be approximated by $\zeta(2n) = (2n-1)!!$ according to Formula (2.28).
- For large $2n$, Formula (2.27) reduces to Formula (2.29), which shows that $\zeta(2n)$ increases slower than σ^{2n} for increasing $2n$.

$$\zeta(2n) = 2N(\sigma)\sigma^{2n}\sum_{j=1}^{\infty}\sigma^{2j-1}\frac{(2n-1)!!}{(2n-1+2j)!!}; \tag{2.27}$$

$$= (2n-1)\zeta(\sigma, 2n-2) - 2N(\sigma)\sigma^{2n-1}; \tag{2.28}$$

$$\sigma^2 \ll 2n: \quad \zeta(2n) \simeq 2N(\sigma)\frac{\sigma^{2n+1}}{2n+1}; \tag{2.29}$$

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 (2.21) is replaced by a uniform distribution between $[-\sigma, +\sigma]$, Formula (2.30) is similar to Formula (2.29):

$$\zeta(2n) \equiv \int_{-\sigma}^{+\sigma}\frac{1}{2\sigma}\tilde{z}^{2n}d\tilde{z} = 2\frac{1}{2\sigma}\frac{\sigma^{2n+1}}{2n+1}; \tag{2.30}$$

The limited range of $\tilde{x} \in (-\sigma\delta x, +\sigma\delta x)$ causes a bounding leakage ϵ according to Formula (2.31), in which $\xi(z)$ is the cumulative density function for normal distribution [4]. When $\sigma = 5$, $\epsilon = 5.73 \times 10^{-7}$, which is small enough for most applications. $\sigma = 5$ is also a golden standard to assert a negative result [2].

$$\epsilon = 2 - 2\xi(\frac{1}{2} + \sigma); \tag{2.31}$$

Thus, $\sigma = 5$ in variance arithmetic. Figure 8 shows the value and uncertainties of the corresponding $\zeta(2n)$. When $2n < 20$, $\zeta(2n)$ can be well approximated by $(2n-1)!!$. Formula (2.23) and (2.23) can be approximated as Formula (2.32) and (2.32), respectively.

$$\begin{aligned}\delta^2 f(x) \simeq\ & (\delta x)^2(f_x^{(1)})^2 + (\delta x)^4\left(f_x^{(1)}f_x^{(3)} + \frac{1}{2}(f_x^{(2)})^2\right) \\ & + (\delta x)^6\left(\frac{1}{4}f_x^{(1)}f_x^{(5)} + \frac{1}{2}f_x^{(2)}f_x^{(4)} + \frac{5}{12}(f_x^{(3)})^2\right);\end{aligned} \tag{2.32}$$

$$\begin{aligned}\delta^2 f(x,y) \simeq\ & (\delta x)^2(f_{(x,y)}^{(1,0)})^2 + (\delta y)^2(f_{(x,y)}^{(0,1)})^2 \\ & + \left(\delta x)^4 f_{(x,y)}^{(1,0)}(\frac{1}{2}f_{(x,y)}^{(2,0)} + f_{(x,y)}^{(3,0)}\right) + (\delta y)^4\left(\frac{1}{2}f_{(x,y)}^{(0,2)} + f_{(x,y)}^{(0,1)}f_{(x,y)}^{(0,3)}\right) \\ & + (\delta x)^2(\delta y)^2\left((f_{(x,y)}^{(1,1)})^2 + f_{(x,y)}^{(0,1)}f_{(x,y)}^{(2,1)} + f_{(x,y)}^{(1,0)}f_{(x,y)}^{(1,2)}\right);\end{aligned} \tag{2.33}$$

2.9.4 One-dimensional Examples

Formula (2.36) gives the result variance for e^x, which always converge:

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

$$\overline{e^x} = e^x + e^x \sum_{n=1}^{\infty} (\delta x)^{2n} \zeta(2n) \frac{1}{(2n)!}; \tag{2.35}$$

$$\widehat{P(e^x)}^2 = \sum_{n=1}^{\infty} (\delta x)^{2n} \left(\sum_{j=1}^{2n-1} \frac{\zeta(2n)}{j!\,(2n-j)!} - \sum_{j=1}^{n-1} \frac{\zeta(2j)}{(2j)!} \frac{\zeta(2n-2j)}{(2n-2j)!} \right) \tag{2.36}$$

$$= (\delta x)^2 + \frac{3}{2}(\delta x)^4 + \frac{7}{6}(\delta x)^6 + \frac{5}{8}(\delta x)^8 + o((\delta x)^{10}); \tag{2.37}$$

Formula (2.40) gives the result variance for $\ln(x)$:

$$\ln(x+\tilde{x}) - \ln(x) = \ln(1+\frac{\tilde{x}}{x}) = \sum_{j=1}^{\infty} \frac{(-1)^{j+1}}{j} \frac{\tilde{x}^j}{x^j}; \tag{2.38}$$

$$\overline{\ln(x)} = \ln(x) - \sum_{n=1}^{+\infty} P(x)^{2n} \frac{\zeta(2n)}{2n}; \tag{2.39}$$

$$\delta^2 \ln(x) = \sum_{n=1}^{+\infty} P(x)^{2n} \left(\sum_{j=1}^{2n-1} \frac{\zeta(2n)}{j(2n-j)} - \sum_{j=1}^{n-1} \frac{\zeta(2j)}{2j} \frac{\zeta(2n-2j)}{2n-2j} \right) \tag{2.40}$$

$$= P(x)^2 + P(x)^4 \frac{9}{8} + P(x)^6 \frac{119}{24} + P(x)^8 \frac{991}{32} + o(P(x)^{10}); \tag{2.41}$$

Formula (2.44) gives the result variance for $\sin(x)$, which converge when $\delta x < 1$:

$$\sin(x+\tilde{x}) = \sum_{n=0}^{\infty} \sin(x)(-1)^n \frac{\tilde{x}^{2n}}{(2n)!} + \sum_{n=0}^{\infty} \cos(x)(-1)^n \frac{\tilde{x}^{2n+1}}{(2n+1)!}; \tag{2.42}$$

$$\overline{\sin(x)} = \sin(x) + \sin(x) \sum_{n=0}^{\infty} (\delta x)^{2n} (-1)^n \frac{\zeta(2n)}{(2n)!}; \tag{2.43}$$

$$\delta^2 \sin(x) = \sum_{n=1}^{\infty} (\delta x)^{2n} (-1)^{n-1} \tag{2.44}$$

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

$$= (\delta x)^2 \cos(x)^2 - (\delta x)^4 (\cos(x)^2 \frac{3}{2} - \frac{1}{2}) + (\delta x)^6 (\cos(x)^2 \frac{7}{6} - \frac{1}{2}) + o((\delta x)^8); \tag{2.45}$$

Formula (2.48) gives the result variance for x^c, in which $\binom{c}{n} = \frac{\prod_{j=0}^{n-1}(c-j)}{n!}$:

$$(x+\tilde{x})^c = x^c(1+\frac{\tilde{x}}{x})^c = x^c + x^c \sum_{n=1}^{\infty} \frac{\tilde{x}^n}{x^n} \binom{c}{n}; \tag{2.46}$$

$$\overline{x^c} = x^c + x^c \sum_{n=1}^{\infty} P(x)^{2n} \zeta(2n) \binom{c}{2n}; \tag{2.47}$$

$$\widehat{P(x^c)}^2 = \sum_{n=1}^{\infty} P(x)^{2n} \left(\sum_{j=1}^{2n-1} \zeta(2n) \binom{c}{j} \binom{c}{2n-j} - \sum_{j=1}^{n-1} \zeta(2j) \binom{c}{2j} \zeta(2n-2j) \binom{c}{2n-2j} \right) \tag{2.48}$$

$$= c^2 P(x)^2 + \frac{3}{2} c^2 (c-1)(c-\frac{5}{3}) P(x)^4 + \frac{7}{6} c^2 (c-1)(c-2)^2 (c-\frac{16}{7}) P(x)^6 + o(P(x)^8); \tag{2.49}$$

As some special cases for Formula (2.48), Formula (2.50) gives the result variance for x^2, Formula (2.51) gives the result variance for $\sqrt{x}$, Formula (2.52) gives the result variance for $1/x$, and Formula (2.53) gives the result variance for $1/x^2$:

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

$$\widehat{P(\sqrt{x})}^2 = \frac{1}{4} P(x)^2 + \frac{7}{32} P(x)^4 + \frac{75}{128} P(x)^6 + o(P(x)^8); \tag{2.51}$$

$$\widehat{P(1/x)}^2 = P(x)^2 + 8P(x)^4 + 69P(x)^6 + o(P(x)^8); \tag{2.52}$$

$$\widehat{P(1/x^2)}^2 = 4P(x)^2 + 66P(x)^4 + 960P(x)^6 + o(P(x)^8); \tag{2.53}$$

2.9.5 Convergence and Truncation of Taylor Expansion

If Formula (2.23) can be expressed as $\sum_{n=1}^{\infty} \upsilon(2n)\zeta(2n)P(x)^{2n}$, such as Formula (2.40) and (2.48), and if $|\upsilon(2n)| \sim \lambda^{2n}$, then Formula (2.23) 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 (2.23) in the $P(x)$-form to converge is defined as the *applicable precision threshold*, which can be estimated as $1/\sigma$ according to Formula (2.29).

Formula (2.23) 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 (2.48).

The variance formulas using Taylor expansion show 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 (2.22) and (2.23) 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 variance arithmetic.

Variance arithmetic may reject a calculation for the following reasons:

- 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 worse than $1/\sigma$, the result should be invalidated as *not reliable*.

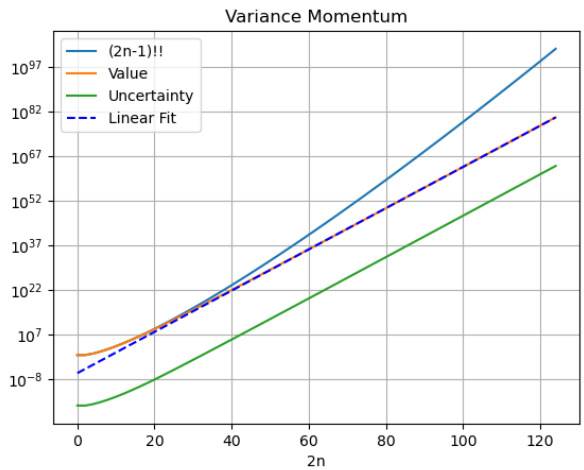

Figure 8: The value of variance momentum $\zeta(2n)$ vs the expansion order $2n$ when $\sigma = 5$. Also included are the uncertainty of the $\zeta(2n)$ due to rounding errors, $2(n-1)!!$, and the linear fit of the $\zeta(2n)$ value.

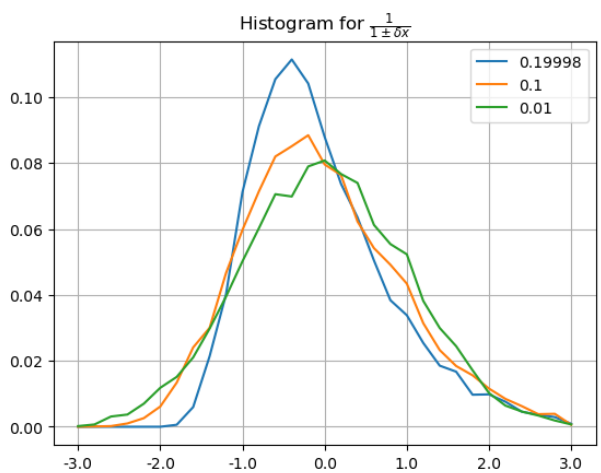

Figure 9: The histogram for $1/(1 \pm \delta x)$ using Gaussian input noises, with δx shown in the legend. The histogram has been normalized by the result mean and uncertainty calculated by variance arithmetic, so that 0 on the x-axis marks the result mean, and ± 1 on the x-axis marks the result uncertainty. The y-axis is scaled as the normalized count of the histogram.

- Due to digital limitation of conventional floating-point representation, $\zeta(2n)$ can only be calculated to limited $2n$. If at the limit, the expansion is still not stable, the result should be invalidated as *practical unstable*. Section 4 will define the stability standard for truncating an expansion.
- For the variance to converge, in Formula (2.23), the contribution from expansion order $2n$ should decrease with increasing $2n$. Otherwise, the result should be invalidated as *not monotonic*.

The rejection usually happens near a zero or a pole, such as when the possibility to calculate 1/0 becomes more than ϵ, so that $1/(1 \pm (0.2 - 1\ 10^{-5}))$ is rejected for not monotonic, while $1/(1 \pm (0.2 - 2\ 10^{-5})) = 1.046 \pm 0.250$. Figure 9 shows the histograms for $1/(1 \pm \delta x)$ when $\delta x = 0.01, 0.1, 0.2 - 2\ 10^{-5}$. When $\delta x = 0.01$, the result is normal distributed. When δx approaches 0.2, the result distribution deviates from normal more and more. The rejection also keeps the result distribution from deviating too much from normal distribution, so that the result value and uncertainty do not deviate too much from those of variance arithmetic. For the worst case of $1/(1 \pm (0.2 - 2\ 10^{-5}))$, the mean is about 0.4% smaller, while the deviation is about 1.3% smaller. Such error when the input precision is very close to applicable precision threshold is ignored in variance arithmetic.

2.10 Dependency Tracing

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

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

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

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

When all the inputs obey the uncorrelated uncertainty assumption, variance arithmetic uses statistics to trace dependency in the intermediate steps. By convention, variance arithmetic treats different input variables as imprecise values satisfying uncorrelated uncertainty assumption. For example:

- Formula (2.54) shows $\delta^2(f+g)$, whose dependence tracing can be demonstrated by $\delta^2(f-f) = 0$ and $\delta^2(f(x)+g(y)) = \delta^2 f + \delta^2 g$, with the latter case corresponding to Formula (2.7). Formula (2.58) shows that Formula (2.23) uses Formula (2.54) between any two terms in the Taylor expansion of Formula (2.20).
- Formula (2.55) shows $\delta^2(fg)$, whose dependence tracing can be demonstrated by $\delta^2(f/f) = 0$ and $\delta^2(f(x)g(y)) = \overline{f}^2(\delta^2 g) + (\delta^2 f)\overline{g}^2 + (\delta^2 f)(\delta^2 g)$, with the latter case corresponding to Formula (2.11).
- Formula (2.56) shows $\delta^2 f(g(x))$, whose dependence tracing can be demonstrated by $\delta^2(f^{-1}(f)) = \delta^2 x$.
- Formula (2.57) shows the variance of linear transformation of a function, which can be applied to Formula (2.54) and (2.55) for more generic dependency tracing.

$$\delta^2\left(\frac{f_x^{(m)}\tilde{x}^m}{m!} + \frac{f_x^{(n)}\tilde{x}^n}{n!}\right) = (\delta x)^{2m}(\frac{f_x^{(m)}}{m!})^2\eta(2m) + (\delta x)^{2n}(\frac{f_x^{(n)}}{n!})^2\eta(2n) \tag{2.58}$$
$$+ 2(\delta x)^{m+n}\left(\frac{f_x^{(m)} f_x^{(n)}}{m!\ n!}\eta(m+n) - \frac{f_x^{(m)}}{m!}\eta(m)\ \frac{f_x^{(n)}}{n!}\eta(n)\right);$$

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

Dependency trading comes with a price: variance calculation is generally much more complex than value calculation.

2.11 Traditional Execution and Dependency Problem

Dependency tracing requires an analytic solution to apply Taylor expansion for the result mean and variance, such as using Formula (2.23) and (2.26). It conflicts with traditional numerical executions on analytic functions:

- Traditionally, using intermediate variables is a common practice. But it upsets dependency tracing.
- Traditionally, different analytic expressions of the same function give the same result. But this is no longer true in variance arithmetic if Taylor expansion is not used. For example, $x^2 - x$ can be calculated equivalently as Formula (1.7), (1.8), and (1.9) when applying Formula (2.7), (2.11), and (2.50) as separated and independent arithmetic operations. Only Formula (1.7) gives the correct result because the input variable x appears only once, while the other two give wrong results for wrong independence assumptions, respectively, as shown in Table 1.
- Traditionally, conditional executions are used to optimize numerical executions, such as the Gaussian elimination to minimize floating-point rounding errors for matrix inversion [45]. Unless required by mathematics such as for the definition of identity matrix, conditional executions need to be removed for dependency tracing, e.g., Gaussian elimination needs to be replaced by the analytic equations as described in Section 5.
- Traditionally, approximations on the result values are carried out during executions. Using variance arithmetic, Formula (2.23) shows that the variance converges much slower than that of the value in Taylor expansion, so that the target of approximation should be more on the variance. Section 5 provides an example of first order approximation on calculating matrix determinant.
- Traditionally, floating point rounding errors, and numerical errors from library functions work largely in dark. Variance arithmetic can now illuminate these sources of result uncertainties. Section 4 and Section 8 show that the current traditional floating-point representation is deficient for both traditional Taylor expansion and for variance arithmetic, respectively.
- Traditionally, a large calculation may be broken into sequential steps, such as calculating $f(g(x))$ as $f(g(x)) = f(y)|_{y=g(x)}$. But this is no longer true in variance arithmetic due to dependency tracing of $g(x)$ inside $f(g(x))$. $\delta^2 f(g(x))$

is calculated as Formula (2.59), which is different from Formula (2.60) for $\delta^2 f(x)$ with $\delta^2 x = \delta^2 g(x)$. The result of Formula (2.60) depends on the sequence of the composite functions, such as the difference between Formula (2.61) and (2.62). In contrast, the result of Formula (2.59) is not sequence dependent.

$$\begin{aligned}\delta^2 f(g(x)) \simeq\ & (\delta x)^2 (f_x^{(1)} g_x^{(1)})^2 \\ & + (\delta x)^4 \left((f_x^{(1)})^2 (g_x^{(1)} g_x^{(3)} + \frac{1}{2}(g_x^{(2)})^2) + \frac{1}{2}(f_x^{(2)})^2 (g_x^{(1)})^4 \right) \\ & + (\delta x)^4 \left(f_x^{(1)} f_x^{(2)} (g_x^{(1)})^4 + 4 f_x^{(1)} f_x^{(2)} (g_x^{(1)})^2 g_x^{(2)} \right) + o((\delta x)^6);\end{aligned} \tag{2.59}$$

$$\begin{aligned}\delta^2 f(x)|_{\delta^2 x = \delta^2 g(x)} \simeq\ & (\delta x)^2 (f_x^{(1)} g_x^{(1)})^2 \\ & + (\delta x)^4 \left((f_x^{(1)})^2 (g_x^{(1)} g_x^{(3)} + \frac{1}{2}(g_x^{(2)})^2) + \frac{1}{2}(f_x^{(2)})^2 (g_x^{(1)})^4 \right) \\ & + (\delta x)^4 \left(f_x^{(1)} f_x^{(3)} (g_x^{(1)})^4 \right) + o((\delta x)^6);\end{aligned} \tag{2.60}$$

$$P(\widehat{y^2|_{y=\sqrt{x}}})^2 \simeq P(x)^2 + \frac{9}{2} P(x)^4; \tag{2.61}$$

$$P(\widehat{\sqrt{y}|_{y=x^2}})^2 \simeq P(x)^2 + \frac{9}{8} P(x)^4; \tag{2.62}$$

Many traditional numerical algorithms are pure numerical, such as numerical equation root finding and Monte Carlo simulation. Although they cannot be described analytically, their results usually have clear uncertainties, such as the uncertainty for the root. Variance arithmetic cannot apply to their execution but can ingest their results.

When the dependency tracing is not applied strictly, variance arithmetic will have dependency problem. Dependency tracing gets rid of almost all freedoms in the traditional numerical executions, as well as the associated dependence problems. Because of dependency tracing, variance arithmetic does not suffer from the dependency problem [21] which has plagued interval arithmetic. All traditional numerical algorithms need to be reexamined or reinvented for variance arithmetic.

3 Verification of 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 numerical result and the corresponding known precise 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 of the result uncertainties.
- *uncertainty deviation*: as the deviation of the result uncertainties.
- *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 input errors from all sources, 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 result 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 when the error deviations are in the range of $[0.1, 10]$.

When an input contains unspecified errors, such as the numerical errors of library functions, Gaussian or white noises of increasing deviations can be added to the input, until ideal coverage is reached. The needed minimal noise deviation gives a good estimation of the amount of unspecified input uncertainties. The existence of ideal coverage is a necessary verification if variance arithmetic correctly tracks the result uncertainty. The input noise range for the idea coverage defines the ideal application range of 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

There are five sources of result uncertainty for a calculation [1][12]:

- input uncertainties,
- rounding errors,

- truncation errors,
- external errors,
- modeling errors.

The examples in this paper show that when the input uncertainty precision is 10^{-15} or larger, variance arithmetic provides ideal coverage for input uncertainties.

Section 4 shows the reason why variance arithmetic can only provide proper coverage for floating-point rounding errors, which is confirmed by all other sections.

When the analytic expression is a sum of infinitive diminishing terms, such as a Taylor expression, its numerical implementation needs to truncate the sum to finite terms, possibly causing *truncation errors*. Section 4 reevaluates this process for variance arithmetic.

The *external errors* are the value errors which are not specified by the input uncertainties, such as the numerical errors in the library math functions. Section 8 studies the effect of sin and tan numerical errors. It demonstrates that when the external errors are large enough, neither ideal coverage nor proper coverage can be achieved. It shows that the library functions need to be recalculated to have corresponding uncertainty attached to each value.

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 8 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 [38]

Algorithms of completely different nature with each representative for its category are needed to test the generic applicability of variance arithmetic. An algorithm can be categorized by comparing the amount of its input and output data:

- application,
- transformation,
- generation,
- reduction.

An *application* algorithm calculates numerical values from an analytic formula. Section 4 and 7 show two applications. In addition to the specified input uncertainties, floating-point rounding errors are hidden input uncertainties.

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 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. For reversible transformations, a unique requirement for variance 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 should be delta distributed with 0 error deviation [7]. A test of variance arithmetic using FFT algorithms is provided in Section 8.

[7] When noises are added to the original input signal, the value errors should be the results minus the overall input data. The previous approaches [38] 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. Some generating algorithms are theoretical calculations which involve no imprecise input so that all result uncertainty is due to rounding errors. Section 9 demonstrates such an algorithm, which calculates a table of the sine function using trigonometric relations and two precise input data, $sin(0) = 0$ and $sin(\pi/2) = 1$.

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]. A test of statistical characterization through sampling is provided in Section 7.

4 Polynomial and Taylor Expansion

4.1 Polynomial

For a polynomial $f(x) \equiv \sum_{j=0}^{N} c_j x^j$, Formula (4.1) gives the corresponding Taylor expansion:

$$f(x+\tilde{x}) = \sum_{j=0}^{N} c_j (x+\tilde{x})^j = \sum_{j=0}^{N} \tilde{x}^j \sum_{k=0}^{N-j} x^{k-j} c_{j+k} \binom{j+k}{j}; \tag{4.1}$$

When $f(x) = (ax+b)^N$, the result of applying Formula (4.1) to Formula (2.23) is Formula (2.48) with $c = N$ and $x = ax+b$. Unlike Formula (2.48), Formula (4.1) is no longer limited by the applicable precision threshold, because the exponent for power is limited to positive integers.

4.2 Truncation Error

Formula (2.58) shows that Taylor expansion such as Formula (2.22) and (2.32) can be treated as polynomial with $N \to \infty$.

A numerical calculation can only calculate limited terms in a Taylor expansion, and the count of terms of the Taylor expansion to approximate a function $f(x)$ is defined as the *Taylor expansion order*. The needed Taylor expansion order can be obtained by theoretical reminder estimation: If the estimation is less than LSV of the current expansion, the Taylor expansion can stop [12]. However, this method has not considered floating-point rounding errors, so that it cannot guarantee that the result is correct for all significant bits beyond the least significant bit of signficand.

By treating LSV of each conventional floating-point value as imprecise, variance arithmetic can trace floating-point rounding errors. Formula (4.2) gives the Taylor expansion of $1/(1-x)$. Taylor expansion for $1/(1-x)$ is used to test the ability of variance arithmetic to predict the effect of floating-point rounding errors. In Formula (4.2), each $|x|^n$ generates the same rounding error, but the difference in sign means that the same rounding error will have different effects for the summation.

$$e(N) = \sum_{n=0}^{N} x^n = \begin{cases} x > 0: & 1 + |x| + |x|^2 + |x|^3 + \dots \\ x < 0: & 1 - |x| + |x|^2 - |x|^3 + \dots \end{cases}; \tag{4.2}$$

$$r(N) = \left| e(N) - \frac{1}{1-x} \right|; \tag{4.3}$$

Instead of a reminder estimator, the true reminder $r(N)$ is calculated as Formula (4.3).

4.3 Rounding Error

Figure 10 shows $r(N)$ and the corresponding uncertainties for $x = \pm 0.6, \pm 0.7$. For $x = \pm 0.6, 0.7$, after $r(N)$ reaches LSV of $1/(1-x)$, the LSV equals the corresponding expansion uncertainties, respectively. For $x = -0.7$, $r(N)$ does not reach LSV, leaving the least two significant bits incorrect, which shows that traditional numerical Taylor expansion can be less accurate than its theoretical claim.

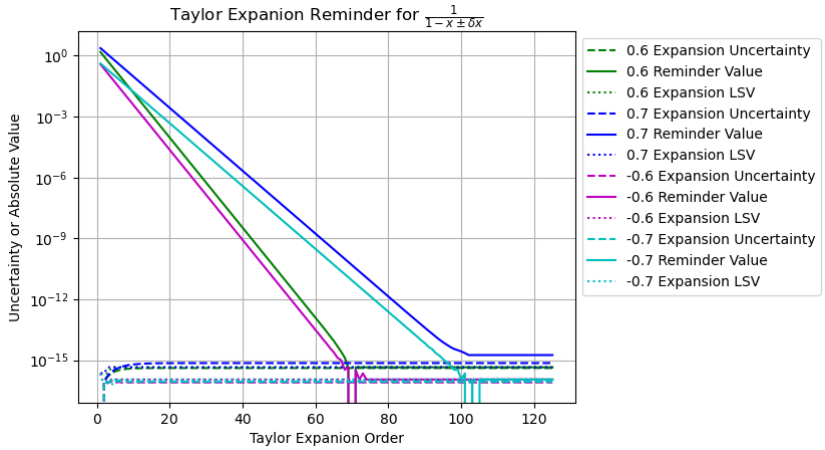

Figure 10: The reminders and result uncertainties for Taylor expansion of $1/(1-x)$, for different Taylor expansion order N as shown in the x-axis, and different x as shown in the legend. The y-axis is for either absolute reminder or expansion uncertainty. In the legend, *Reminder Value* is the value of Formula (4.3), *Expansion Uncertainty* is the uncertainty of Formula (4.2), and *Expansion LSV* is the *LSV* of the value of Formula (4.2).

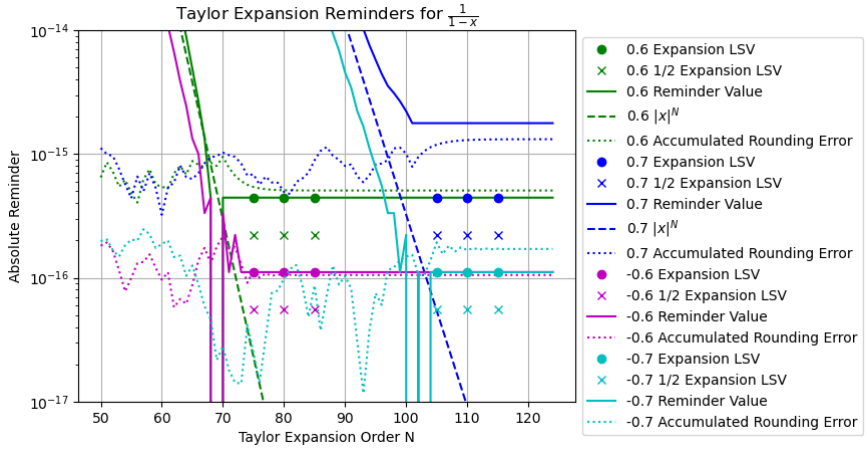

Figure 11: The reminders for Taylor expansion of $1/(1-x)$, for different Taylor expansion order N as shown in the x-axis, and different x as shown in the legend. The y-axis is for absolute reminder. In the legend, *Reminder Value* is the value of Formula (4.3), $|x|^N$ is the absolute expansion term at Taylor expansion order N, *Expansion LSV* is the *LSV* of the value of Formula (4.2), *1/2 Expansion LSV* is half of *Expansion LSV*, and *Accumulated Rounding Error* is the sum of rounding errors of $x^n, n = 1 \ldots N$.

8+	4+	2+	1+	1^-	0+
	8−	4−	2−	1^-	0+
	7+	4−	2−	1−	0+
	7−	3+	2−	1−	0+
	6+	3+	2−	1−	0+
	6−	3−	1+	1−	0+
	5+	3−	2+	1−	0+
	5^-	2+	1+	1−	0+

Table 2: The rounding pattern of positive integers when they are repeatedly divided by 2 and after each division the result is rounded to the nearest. Each row contains such a process from left to right. Each cell in the table contains an integer P with its rounding error sign bit r [38], as Pr. r indicates the sign of the current rounding error, to achieve round to nearest in each step, such as $7 + /2 \rightarrow 4-$ and $7 - /2 \rightarrow 3+$. If the rounding error is no more than half of LSV initially, it will never exceed half of LSV of P in the above process.

In addition to $r(N)$, Figure 11 shows the expansion term $|x|^N$ and the accumulated rounding errors, zooming in at where the expansions terminate.

1. When x^N is added to $e(N)$ in Formula (4.2), it is digitized by the LSV of $e(N)$, to generate rounding error at N. For example, $0.7^{100} = 3.23\ 10^{-16}$ is digitized by the $LSV = 4.44\ 10^{-16}$ as $4.44\ 10^{-16}$ for $1/(1 - 0.7)$. Such digitization is observed to be round to the nearest.
2. When x^N reaches below half of LSV of $e(N)$, it vanishes so that $e(N)$ reaches its final value.
3. Figure 10 shows that while the accumulated rounding errors fluctuate around 0, the last few accumulated rounding errors when x^N approaches LSV of $e(N)$ determine the final value error of $e(N)$, such as the accumulation of 2-bit errors for $1/(1 - 0.7)$.
4. Such accumulation of rounding errors is probably not by chance. When an integer is repeatedly divided by 2 and after each division the result is rounded to the nearest, the value of the integer before it becomes 0 is not randomly distributed, e.g., it is always 1 in the last step, and predominantly 2 in the step before, and more randomly distributed in the step before, as shown in Table 2.
5. x^N approaches LSV faster for $x = 0.6$ than that for $x = 0.7$, to prevent the pattern in Table 2 from forming, so that the accumulated rounding errors does not cause the result value error beyond the least significant bit of significand.
6. The negative sign in Formula (4.2) for $x = -0.7$ prevents the pattern in Table 2 from forming, so that the accumulated rounding error for $x = -0.7$ is smaller than that for $x = +0.7$.
7. When the $LSV \ll |x|^N$, the value pattern in Table 2 is random, so that rounding errors cannot accumulate steadily before $|x|^N$ reaches near the LSV.

Floating-point rounding error is a major source of value errors for Taylor expansion. To be free from such rounding errors, the input range for an expansion has to be smaller than the convergence input range, e.g., for $1/(1 - |x|)$, $|x|$ has to be smaller than 0.7.

In Figure 10, although the expansion uncertainty has the correct trend, it does not reach $r(N)$ at the end for $1/(1-0.7)$. The reason is due to the digital limitation of the conventional floating-point representation. The smallest positive floating-point value is about $5\ 10^{-324}$. The LSV for 0.7 is about $1\ 10^{-16}$. At $n = 100$, applying Formula (4.1) to Formula (2.23) has 100 terms from $\sim (\delta x)^2$ to $\sim (\delta x)^{200}$, but only the first 10 terms can be calculated before $(\delta x)^{2n}$ vanishes below $5\ 10^{-324}$. Thus, variance arithmetic can only provide proper coverage to floating-point rounding errors.

4.4 Stability Truncation

The input to an expansion can also be imprecise. Figure 12 shows that the absolute reminder values decrease exponentially with increasing expansion order, while the expansion uncertainty stabilizes quickly above the input uncertainty 0.01. The expansion can terminate when the reminder value equals the expansion value LSV for $x = \pm 0.6 \pm 0.01, -0.7 \pm 0.01$, but not for $x = 0.7 \pm 0.01$. Figure 12 is very similar to Figure 10 except that the expansion uncertainty is much larger.

When input uncertainty increases from 0.01 to 0.04, Figure 13 shows that the absolute reminder value decreases slower than exponentially with increasing expansion order, while the expansion uncertainty remains about the same with increasing Taylor expansion order. For example, the expansion of $1/(1 + 0.7 \pm 0.04)$ now terminates at Taylor expansion order 105 and instead of 70 in Figure 12. The reason for such changes is due to the increase of uncertainty bias in Formula (4.1).

It is questionable to use the expansion truncation standard for precise input to imprecise input. Specifically, it is questionable to always calculate the result value to 10^{-16} precision while the result precision is only about 0.16 for the case of $1/(1 + 0.6 \pm 0.04) = 2.526 \pm 0.26$. Because variance arithmetic allows statistical bounding leakage of $\epsilon = 5.73 \times 10^{-7}$ according to Formula (2.31), the tolerance on the result value should be likewise statistically. Formula (4.4) gives the *required value precision* τ, in which $\xi(z)$ is the cumulative density function for normal distribution [4].

$$2\xi(\tau) = \epsilon = 2 - 2\xi(\frac{1}{2} + \sigma): \quad \tau \simeq \epsilon\frac{\sqrt{2\pi}}{2} = 7.18 \times 10^{-7}; \tag{4.4}$$

Using τ as the tolerance, the rule for *stability truncation* for truncating an expansion $e(N)$ at the expansion order N is the following:

- The absolute uncertainty change from $e(N-1)$ to $e(N)$ is less than τ-fold of the uncertainty of $e(N)$, and
- The absolute value change from $e(N-1)$ to $e(N)$ is less than either τ-fold of the uncertainty of $e(N)$, or the LSV of the value of $e(N)$.

The stability truncation is free from reminder estimator, and it is based on the stability of $e(N)$ at and after the expansion order N. The true reminder $r(N)$ at the truncation is renamed as the truncation error.

Figure 14 shows the stability truncation orders for Taylor expansion of $1/(1-x\pm\delta x$, while Figure 15 shows the corresponding truncation errors. To validate stability truncation in this case, the truncation marked as *Reminder Error* is calculated using the same rule for stability truncation but using the uncertainty of $1/(1-x)$ instead of the uncertainty of $e(N)$. Figure 14 and 15 shows that both truncation method have very similar results, so that truncation without reminder estimator is validated in this case.

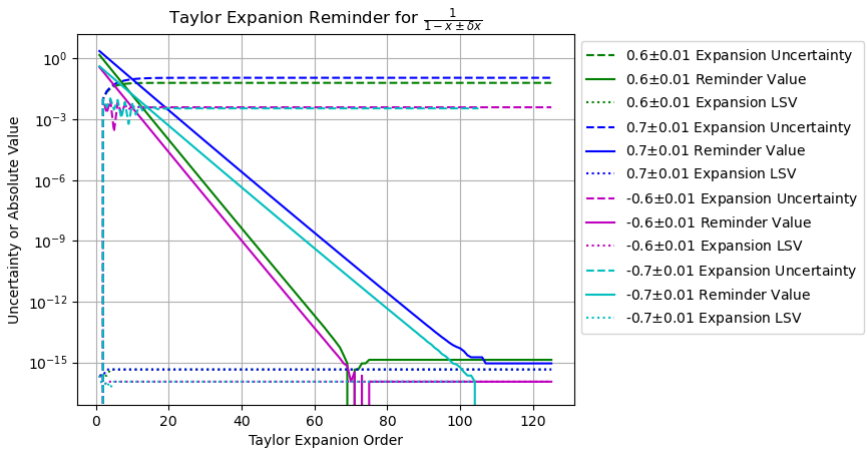

Figure 12: The reminder values and the expansion uncertainties for Taylor expansion of $\frac{1}{1-x\pm\delta x}$, for different Taylor expansion order N as shown in the x-axis, and different input $x \pm \delta x$ as shown in the legend. In the legend, *Reminder Value* is the value of Formula (4.3), *Expansion LSV* is the *LSV* of the value of Formula (4.2), *Expansion Uncertainty* is the uncertainty of Formula (4.2).

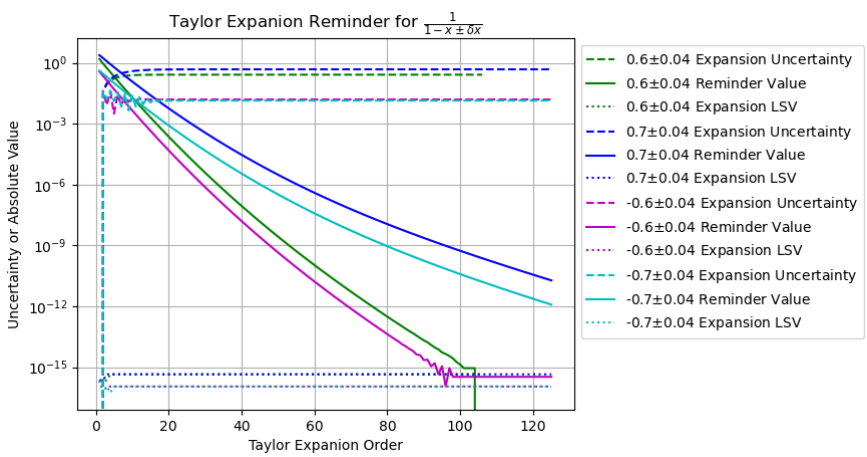

Figure 13: The reminder values and the expansion uncertainties for Taylor expansion of $\frac{1}{1-x\pm\delta x}$, for different Taylor expansion order N as shown in the x-axis, and different input $x \pm \delta x$ as shown in the legend. In the legend, *Reminder Value* is the value of Formula (4.3), *Expansion LSV* is the *LSV* of the value of Formula (4.2), *Expansion Uncertainty* is the uncertainty of Formula (4.2).

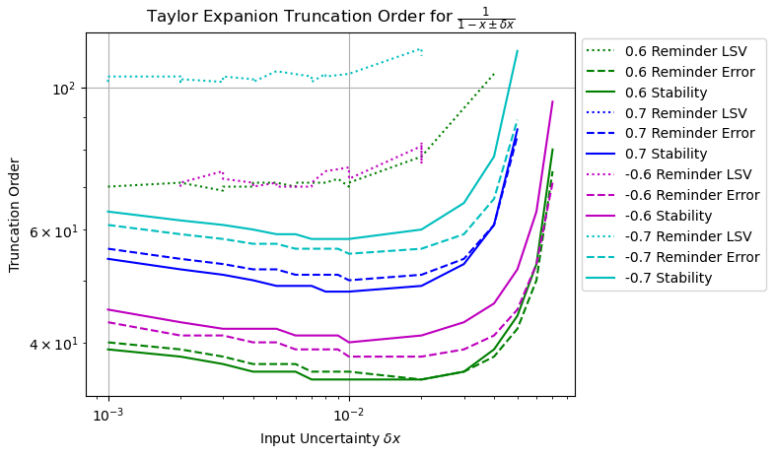

Figure 14: The truncation orders for Taylor expansion of $\frac{1}{1-x\pm\delta x}$, for different input uncertainty δx as shown in the x-axis, and different input value x and truncation rule as shown in the legend. In the legend, *Reminder LSV* means when the value of $r(N)$ is less than the *LSV* of the value of $\frac{1}{1-x\pm\delta x}$, *Reminder Error* means when the absolute value change from $e(N-1)$ to $e(N)$ is less than τ-fold of the uncertainty of $1(1-x)$, *Stability* means to use the stability criterion to decide maximal N for $e(N)$.

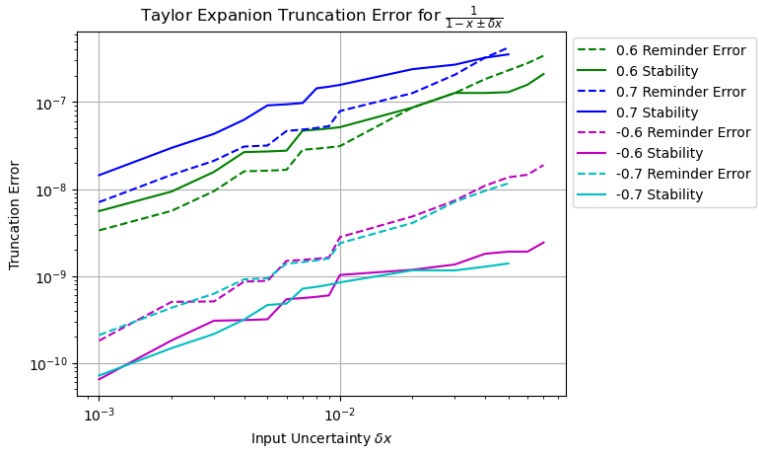

Figure 15: The truncation errors for Taylor expansion of $\frac{1}{1-x\pm\delta x}$, for different input uncertainty δx as shown in the x-axis, and different input value x and truncation rule as shown in the legend. In the legend, *Reminder Error* means when the absolute value change from $e(N-1)$ to $e(N)$ is less than τ-fold of the uncertainty of $1(1-x)$, *Stability* means to use the stability criterion to decide maximal N for $e(N)$.

As an comparison, the truncation marked as *Reminder LSV* is the traditional truncation method, whose truncation orders generally double the corresponding stability truncation orders, showing that stability truncation is more efficient.

As expected, Figure 15 shows that truncation errors are linearly proportional to input uncertainties. Figure 14 shows that for each case, the truncation order has a global minimum, and on each side, either increases slowly with decreasing input uncertainty, or increases fast with increasing input uncertainty. Even when the input uncertainty is comparable to the floating-point rounding error, stability truncation still works by comparing the expansion value change with the LSV of the value after the expansion uncertainty has been stable.

Due to the digital limitation of the 64-bit conventional floating-point representation, $\zeta(2n)$ can only be calculated up to $2n = 252$, so that Formula (4.1) can only be calculated for $n < 126$. If after 126 orders of expansion, stability truncation can still not be achieved, the expansion is deemed to be *practically unstable*. Figure 14 shows that when the input uncertainty is too large, the Taylor expansion is practically unstable, such as for $1/(1 - 0.6 \pm 0.06)$. However, using Formula (2.48), $1/(1-0.6\pm0.06) = 2.561\pm0.415$. Because Formula (4.1) expands by $(\delta x)^2$, it converges slow than Formula (2.48) which expands by $P(x)^2$. For $1/(1 - 0.6 \pm 0.06)$, Formula (4.1) is practically unstable, while Formula (2.48) is not. If $\zeta(2n)$ were not limited by $n < 126$, these two formulas should have given identical result.

Stability truncation is applied numerically in variance arithmetic for all Taylor expansion related calculations.

5 Matrix Inversion

5.1 Uncertainty Propagation in Matrix Determinant

Let vector $[p_1, p_2 \dots p_n]_n$ denote a permutation of the vector $(1, 2 \dots n)$ [45]. Let $\$[p_1, p_2 \dots p_n]_n$ denote the permutation sign of $[p_1, p_2 \dots p_n]_n$ [45]. Formula (5.1) [45] defines the determinant of a n-by-n square matrix $\boldsymbol{M}$ with the element $x_{i,j}, i, j = 1 \dots n$. The sub-determinant [45] $M_{i,j}$ at index (i, j) is formed by deleting the row i and column j of M, whose determinant is given by Formula (5.2) [45]. Formula (5.3) holds for the arbitrary row index i or the arbitrary column index j [45].

$$|\boldsymbol{M}| \equiv \sum_{[p_1 \dots p_n]_n} \$[p_1 \dots p_n]_n \prod_{k=1 \dots n} x_{k,p_k}; \tag{5.1}$$

$$|\boldsymbol{M}|_{i,j} \equiv \sum_{[p_1 \dots p_n]_n}^{p_i = j} \$[p_1 \dots p_n]_n \prod_{k=1 \dots n}^{k \neq i} x_{k,p_k}; \tag{5.2}$$

$$|\boldsymbol{M}| = \sum_{j=1 \dots n} |\boldsymbol{M}_{i,j}| x_{i,j} = \sum_{i=1 \dots n} |\boldsymbol{M}_{i,j}| x_{i,j}; \tag{5.3}$$

Assuming $p_1, p_2 \in \{1 \dots n\}$, 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 $p_1 < p_2$, and let $< i_1, i_2 >_n$ be an arbitrary ordered permutation [8]. Formula (5.3) can be applied to $M_{i,j}$, as:

$$|\boldsymbol{M}_{<i_1,i_2>_n,[j_1,j_2]_n}| \equiv \sum_{[p_1 \dots p_n]_n}^{p_{i_1}=j_1, p_{i_2}=j_2} \$[p_1 \dots p_n]_n \prod_{k=1 \dots n}^{k \notin \{i_1,i_2\}} x_{k,p_k}; \tag{5.4}$$

$$|\boldsymbol{M}| = \sum_{j_1=1 \dots n} x_{i_1,j_1} |\boldsymbol{M}_{i_1,j_1}| = \sum_{j_1=1 \dots n} \sum_{j_2=1 \dots n}^{j_2 \neq j_1} |\boldsymbol{M}_{<i_1,i_2>_n,[j_1,j_2]_n}| x_{i_1,j_1} x_{i_2,j_2}; \tag{5.5}$$

The definition of a sub-determinant can be extended to Formula (5.6), in which $m = 1 \dots n$. Formula (5.5) can be generalized as Formula (5.7), in which $< i_1 \dots i_m >_n$ is an arbitrary ordered permutation. The $(n-m)$-by-$(n-m)$ matrix in $|\boldsymbol{M}_{<i_1 \dots i_m>_n [j_1 \dots j_m]_n}|$ is obtained by deleting the rows in $\{i_1 \dots i_m\}$ and the columns in $\{j_1 \dots j_m\}$. This leads to Formula (5.8) [9].

$$|\boldsymbol{M}_{<i_1 \dots i_m>_n,[j_1 \dots j_m]_n}| \equiv \sum_{[p_1 \dots p_n]_n}^{k \in \{i_1 \dots i_m\}: p_k = j_k} \$[p_1 \dots p_n]_n \prod_{k=1 \dots n}^{k \notin \{i_1 \dots i_m\}} x_{k,p_k}; \tag{5.6}$$

$$|\boldsymbol{M}| = \sum_{[j_1 \dots j_m]_n} |\boldsymbol{M}_{<i_1 \dots i_m>_n,[j_1 \dots j_m]_n}| \prod_{k=1}^{m} x_{i_k,j_k}; \tag{5.7}$$

$$\left| |\boldsymbol{M}_{<i_1 \dots i_m>_n,[j_1 \dots j_m]_n}| \right| = ||\boldsymbol{M}_{<i_1 \dots i_m>_n,<j_1 \dots j_m>_n}||; \tag{5.8}$$

A matrix contains up to n^2 elements, so the direct application of the n^2-extension of Formula (2.25) and (2.26) is not convenient. Let $\tilde{x}_{i,j}$ be the uncertain part for $x_{i,j}$. Formula (5.9) give the Taylor expansion $|\widetilde{\boldsymbol{M}}|$ of $|\boldsymbol{M}|$, which is linear for each $x_{i,j}$.

[8] An ordered permutation is a sorted combination.

[9] $||\boldsymbol{M}||$ in Formula (5.8) means absolute value of determinant.

$\overline{|\boldsymbol{M}|}$ can be deducted from $|\widetilde{\boldsymbol{M}}|$, while $\delta^2|\boldsymbol{M}|$ can be deducted from $|\widetilde{\boldsymbol{M}}|^2 - \overline{|\boldsymbol{M}|}^2$. In $|\widetilde{\boldsymbol{M}}|$ or $|\widetilde{\boldsymbol{M}}|^2$, when all $\tilde{x}_{i,j}$ are independent of each other, $\tilde{x}_{i,j}^{2k+1} \Rightarrow 0$, while $x_{i,j}^{2k} \Rightarrow \zeta(2k)(\delta x_{i,j})^{2k}$. Such deduction leads to Formula (5.10) and (5.11).

$$|\widetilde{\boldsymbol{M}}| = \sum_{[p_1 \ldots p_n]_n} \$[p_1 \ldots p_n]_n \prod_{i=1\ldots n} (x_{i,p_i} + \tilde{x}_{i,p_i}) \tag{5.9}$$

$$= \sum_{m=0\ldots n} \sum_{<i_1 \ldots i_m>_n} \sum_{[j_1 \ldots j_m]_n} M_{<i_1 \ldots i_m>_n, [j_1 \ldots j_m]_n} \prod_{i=1\ldots m}^{i \in \{i_1 \ldots i_m\}} \tilde{x}_{i,p_i};$$

$$\overline{|\boldsymbol{M}|} = |\boldsymbol{M}|; \tag{5.10}$$

$$\delta^2|\boldsymbol{M}| = \sum_{m=1}^{n} \sum_{<i_1 \ldots i_m>_n} \sum_{[j_1 \ldots j_m]_n} |\boldsymbol{M}_{<i_1 \ldots i_m>_n, <j_1 \ldots j_m>_n}|^2 \prod_{k=1\ldots n}^{i_k \in \{i_1 \ldots i_m\}} (\delta x_{i_k, j_k})^2; \tag{5.11}$$

Formula (5.10) and (5.11) assume that the uncertainties of matrix elements are independent of each other. In contrast, for the special matrix in Formula (5.12), after applying Formula (2.25) and (2.26), Formula (5.13) and (5.14) give the result mean and variance, respectively.

$$|\boldsymbol{M}| = \begin{vmatrix} x, z, y \\ z, y, x \\ y, x, z \end{vmatrix} = 3xyz - x^3 - y^3 - z^3; \tag{5.12}$$

$$\overline{|\boldsymbol{M}|} = |\boldsymbol{M}| - 3x(\delta x)^2 - 3y(\delta y)^2 - 3z(\delta z)^2; \tag{5.13}$$

$$\begin{aligned} \delta^2|\boldsymbol{M}| = {} & 3(5x^4 - 12x^2yz + 2xy^3 + 2xz^3 + 3y^2z^2)(\delta x)^2 \\ & + 3(5y^4 - 12y^2xz + 2yx^3 + 2yz^3 + 3x^2z^2)(\delta y)^2 \\ & + 3(5z^4 - 12z^2xy + 2zx^3 + 2zy^3 + 3x^2y^2)(\delta z)^2 \\ & + 9(z^2 + 2xy)(\delta x)^2(\delta y)^2 + 9(y^2 + xz)(\delta x)^2(\delta z)^2 + 9(x^2 + 2yx)(\delta y)^2(\delta z)^2 \\ & + 9(5x^2 - 2yz)(\delta x)^4 + 9(5y^2 - 2xz)(\delta y)^4 + 9(5z^4 - 2xy)(\delta z)^4 \\ & + 15(\delta x)^6 + 15(\delta y)^6 + 15(\delta z)^6; \end{aligned} \tag{5.14}$$

5.2 Adjugate Matrix

The square matrix whose element is $(-1)^{i+j}|\boldsymbol{M}_{j,i}|$ is defined as the *adjugate matrix* [45] $\boldsymbol{M}^A$ to the original square matrix $\boldsymbol{M}$. Let $\boldsymbol{I}$ be the identical matrix for $\boldsymbol{M}$. Formula (5.15) and (5.16) show the relation of $\boldsymbol{M}^A$ and $\boldsymbol{M}$ [45], which only involve addition, subtraction and multiplication.

$$\boldsymbol{M} \times \boldsymbol{M}^A = \boldsymbol{M}^A \times \boldsymbol{M} = |\boldsymbol{M}|\boldsymbol{I}; \tag{5.15}$$

$$|\boldsymbol{M}|\boldsymbol{M}^A = |\boldsymbol{M}^A|\boldsymbol{M}; \tag{5.16}$$

To test Formula (5.11):

1. A matrix $\boldsymbol{M}$ is constructed using random integers uniformly distributed in the range of $[-2^8, +2^8]$, which has a distribution deviation of $2^8/\sqrt{3}$. The integer arithmetic grantees that $|\boldsymbol{M}|$, $\boldsymbol{M}^A$, and $|\boldsymbol{M}^A|$ are precise.

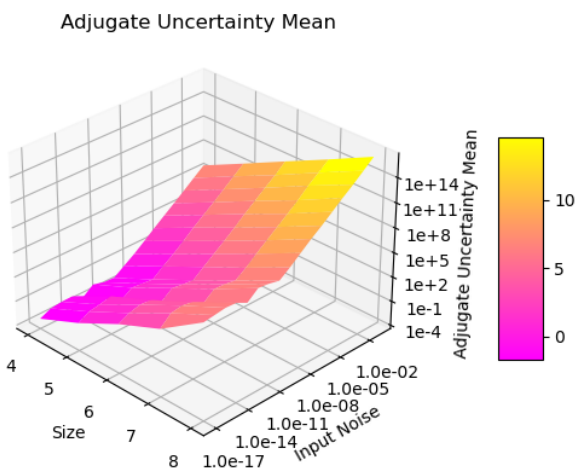

Figure 16: The result uncertainty means vs. input noise precision and matrix size for the difference between $\widetilde{\boldsymbol{M}}^A$ and $\boldsymbol{M}^A$. The input noise precision runs from 10^{-17} to 10^{-1}, while the matrix size runs from 4 to 8.

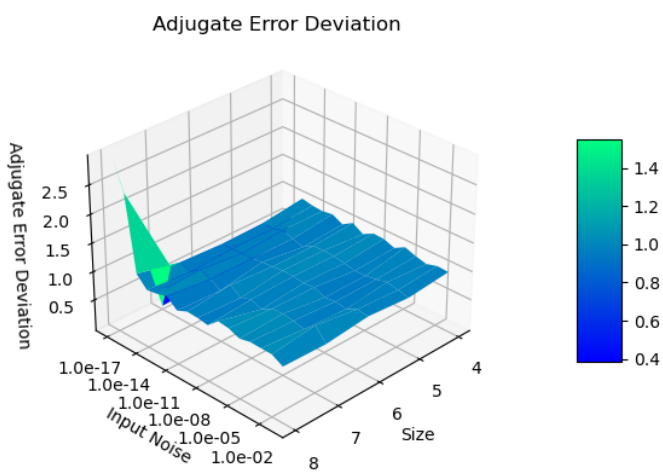

Figure 17: The result error deviations vs. input noise precision and matrix size for the difference between $\widetilde{\boldsymbol{M}}^A$ and $\boldsymbol{M}^A$. The input noise precision runs from 10^{-17} to 10^{-1}, while the matrix size runs from 4 to 8.

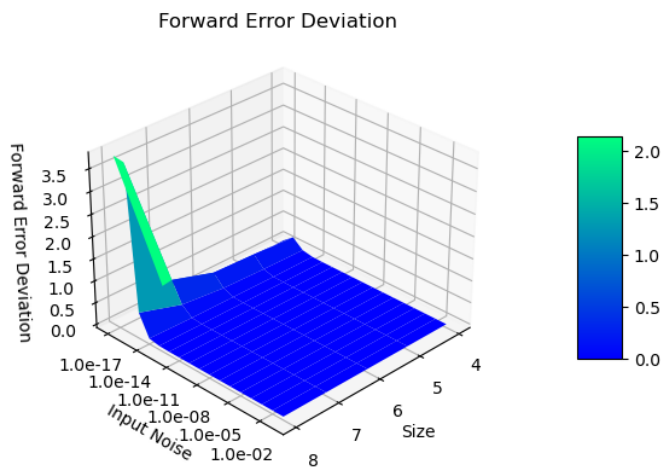

Figure 18: The result error deviations vs. input noise precision and matrix size for the difference of Formula (5.15). The input noise precision runs from 10^{-17} to 10^{-1}, while the matrix size runs from 4 to 8.

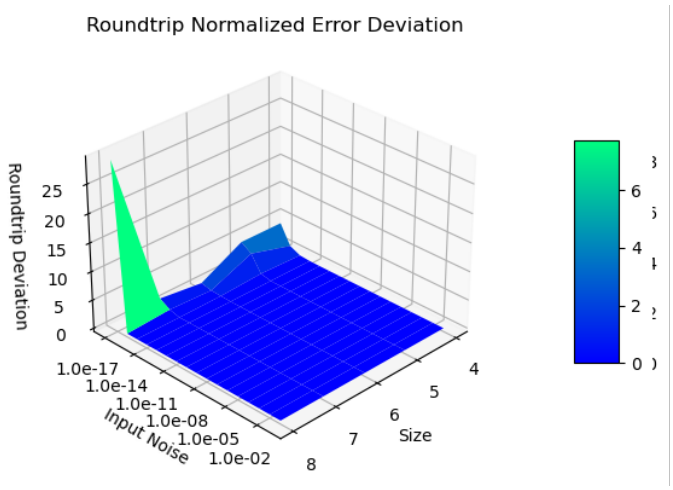

Figure 19: The result error deviations vs. input noise precision and matrix size for the difference of Formula (5.16). The input noise precision runs from 10^{-17} to 10^{-1}, while the matrix size runs from 4 to 8.

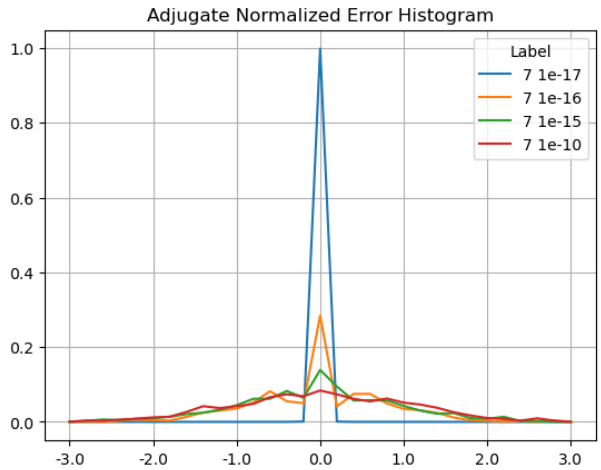

Figure 20: The result histograms for the normalized errors of the adjugate matrix for the same matrix size 7 and different input noise precision as shown in the legend.

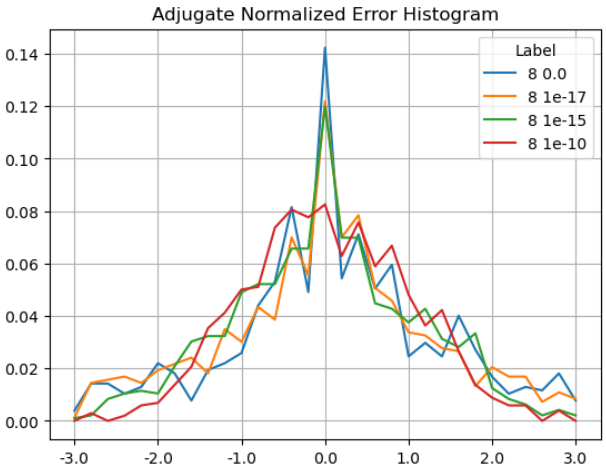

Figure 21: The result histograms for the normalized errors of the adjugate matrix for the same matrix size 8 and different input noise precision as shown in the legend.

Error Deviation	Input Noise Precision			
Matrix Size	0	10^{-17}	10^{-16}	10^{-15}
4	0	0	0.68	1.03
5	0	0.003	0.60	0.96
6	0	0.006	0.80	1.04
7	0	0.083	1.05	1.05
8	4.62	3.91	3.01	1.01

Table 3: The error deviation as a function of matrix size and input noise precision for $\boldsymbol{M}^A$ in which $\boldsymbol{M}$ is generated by random integers between $[-2^8, +2^8]$. The measured error deviations change slightly for different runs.

2. Gaussian noises of predefined levels are added to $\boldsymbol{M}$, to construct a $\widetilde{\boldsymbol{M}}$ which contains imprecise values. Variance arithmetic is used to calculate $|\widetilde{\boldsymbol{M}}|$, $\widetilde{\boldsymbol{M}}^A$, and $|\widetilde{\boldsymbol{M}}^A|$. For example, to construct a $\widetilde{\boldsymbol{M}}$ with 10^{-3} input noise precision, the intensity of the Gaussian noise is $10^{-3} \times 2^8/\sqrt{3}$.
3. The difference between $\widetilde{\boldsymbol{M}}^A$ and $\boldsymbol{M}^A$ defines the *Adjugate* Test. Figure 16 shows that the result uncertainties increase exponentially with both the input noises and the matrix size, while Figure 17 shows that the ideal coverage is achieved for the input precision greater than 10^{-16}.
4. Formula (5.15) defines the *Forward* Test. Figure 18 shows that Formula (5.15) is strictly observed for the idea coverage.
5. Formula (5.16) defines the *Roundtrip* Test. Figure 19 shows that Formula (5.16) is strictly observed for the idea coverage.

Figure 20 and 21 shows that the error values are normal distributed for the input noise precision of 10^{-10}, as a typical example of the histograms for ideal coverage.

The existence of ideal coverage validates Formula (5.11).

5.3 Floating Point Rounding Errors

The significand of the conventional floating-point representation [10] has 52-bit resolution, which is equivalent to 10^{-16}. Adding noise below this level will not change the value of a conventional floating-point value, so the result uncertainties are caused by the round errors in the floating-point calculations.

Figure 20 shows that the histograms of the normalized errors is delta-distributed for the input uncertainty 10^{-17}, because the adjugate matrix calculation involves about $8 \times 6 = 48$ significand bits for a matrix size 7. Such delta distribution also holds when the matrix size is less than 7, or when the input noise precision is 0. When the matrix size is 8, $8 \times 7 = 6$ significand bits are needed so rounding occurs. As a result, in Figure 21, the distribution becomes Gaussian with a hint of delta distribution. Such delta-like distribution persists until the input noise precision reaches 10^{-10}, which is also the transition of the two trends in Figure 16.

When the input noise precision increases from 10^{-17} to 10^{-10}, the distributions become less delta-like and more Gaussian, as shown in both Figure 20 and Figure 21. The non-ideal behaviors in Figure 17, 18, and 19 are all attributed to the round-

ing errors. Table 3 shows that variance arithmetic achieves proper coverage for the calculation rounding errors for adjugate matrix.

5.4 Matrix Inversion

A major usage of $|\boldsymbol{M}|$ and $\boldsymbol{M}^A$ is to define $\boldsymbol{M}^{-1}$ in Formula (5.17) which satisfies Formula (5.18). When $\boldsymbol{M}$ contains imprecise elements, Formula (5.19) shows that the direct application of Formula (5.17) numerically is wrong, which agrees with the common practice to avoid using Formula (5.17) directly [12]. Another reason is that according to Formula (2.52), the denominator uncertainty makes the result precision significantly worse.

$$\boldsymbol{M}^{-1} \equiv \boldsymbol{M}^A / |\boldsymbol{M}|; \tag{5.17}$$

$$(\boldsymbol{M}^{-1})^{-1} = \boldsymbol{M}; \tag{5.18}$$

$$\overline{\boldsymbol{M}^{-1}_{i,j}} \neq |\boldsymbol{M}_{j,i}| / |\boldsymbol{M}|; \tag{5.19}$$

Traditionally, the effect of uncertainty of $\boldsymbol{M}$ to $\boldsymbol{M}^{-1}$ is quantified by matrix condition number [45]. In Formula (5.17), $\boldsymbol{M}^{-1}$ is dominated by $1/|\boldsymbol{M}|$, suggesting that the precision of $\boldsymbol{M}^{-1}$ is largely determined by the precision of $|\boldsymbol{M}|$. Variance arithmetic treats the least significant value of a floating-point value to be imprecise, so it can use Formula (5.14) to calculate the determinant variance for a matrix containing floating-point elements, including a randomly generated matrix, or a Hilbert matrix [45] after it is converted to a floating-point matrix. Figure 22 shows that there is a strong linear correlation between the conditional number and the determinant precision of matrices, so that the determinant precision can replace the condition number, which is much more difficult to calculate.

In fact, condition number may not be the best tool to describe the response to input uncertainties of a matrix. Formula (5.20) shows an example matrix inversion using variance arithmetic representation. It clearly shows the overall precision dominance by the value of 2 ± 0.161 on the denominator, e.g., 1 ± 0.1 in the adjugate matrix becomes -0.510 ± 0.180 in the inverted matrix, while the direct calculation of $1 \pm 0.1/-2 \pm 0.161 = -0.503 \pm 0.065$ has 3-fold better result precision. In Formula (5.20), even though the input matrix has element precision ranging from 10^{-1} to $2.5\ 10^{-5}$, the inverted matrix has element precision all near 0.2 except the worst one at 0.35, showing how dramatically the element precision gets worse in matrix inversion.

$$\begin{aligned}\begin{pmatrix} 1 \pm 10^{-1}, & 2 \pm 10^{-2} \\ 3 \pm 10^{-3}, & 4 \pm 10^{-4} \end{pmatrix}^{-1} &= \frac{\begin{pmatrix} 4 \pm 10^{-4}, & -2 \pm 10^{-2} \\ -3 \pm 10^{-3}, & 1 \pm 10^{-1} \end{pmatrix}}{-2 \pm 0.161} \\ &\simeq \begin{pmatrix} -2.000 \pm 0.401, & 1.000 \pm 0.201 \\ 1.500 \pm 0.301, & -0.510 \pm 0.180 \end{pmatrix};\end{aligned} \tag{5.20}$$

Formula (5.21) gives the Taylor expansion for the element $\boldsymbol{M}^{-1}_{j,i}$. From it, $\overline{\boldsymbol{M}^{-1}_{j,i}}$ and $\delta^2 \boldsymbol{M}^{-1}_{j,i}$ can be deduced, in the same way as how $\overline{|\boldsymbol{M}|}$ and $\delta^2|\boldsymbol{M}|$ are deducted from $|\widetilde{\boldsymbol{M}}|$. Although the analytic expression for $\overline{|\boldsymbol{M}|}$ and $\delta^2|\boldsymbol{M}|$ is very complex on

paper, the deduction is straight-forward in analytic programming such as in *SymPy*.

$$\widetilde{\boldsymbol{M}^{-1}}_{j,i} = (-1)^{i+j}\frac{|\widetilde{\boldsymbol{M}_{i,j}}|}{|\widetilde{\boldsymbol{M}}|} \tag{5.21}$$

$$= (-1)^{i+j}\left(\sum_{m=0}^{n}\sum_{<i_1\ldots i_m>_n}^{i\in\{i_1\ldots i_m\}}\sum_{[j_1\ldots j_m]_n}^{j_i=j} M_{<i_1\ldots i_m>_n,[j_1\ldots j_m]_n}\prod_{k=1\ldots m}^{k\in\{i_1\ldots i_m\}}\tilde{x}_{k,p_k}\right)$$

$$\sum_{h=0}^{\infty}\frac{(-1)^h}{|M|^{h+1}}\left(\sum_{m=1}^{n}\sum_{<i_1\ldots i_m>_n}\sum_{[j_1\ldots j_m]_n} M_{<i_1\ldots i_m>_n,[j_1\ldots j_m]_n}\prod_{k=1\ldots m}^{k\in\{i_1\ldots i_m\}}\tilde{x}_{k,p_k}\right)^h;$$

5.5 First Order Approximation

Formula (5.22) shows the first order approximation of $|\widetilde{\boldsymbol{M}}|$ leads to the first order approximation of $\delta^2|\boldsymbol{M}|$. It essentially states that when the input precision is much less than 1, the determinant $|\boldsymbol{M}|$ of an imprecise matrix $\boldsymbol{M}$ can be calculated in variance arithmetic using Formula (5.1) directly.

$$|\widetilde{\boldsymbol{M}}| \simeq \sum_{[p_1\ldots p_n]_n}\$[p_1\ldots p_n]_n\prod_{k=1\ldots n}(x_{k,p_k}+\tilde{x}_{k,p_k});\quad\Rightarrow\quad \delta^2|\boldsymbol{M}| \simeq \sum_{i}^{n}\sum_{j}^{n} M_{i,j}(\delta x_{i,j})^2; \tag{5.22}$$

Figure 23 contains the result of applying Formula (5.22). It is very similar to Figure 17, validating Formula (5.22).

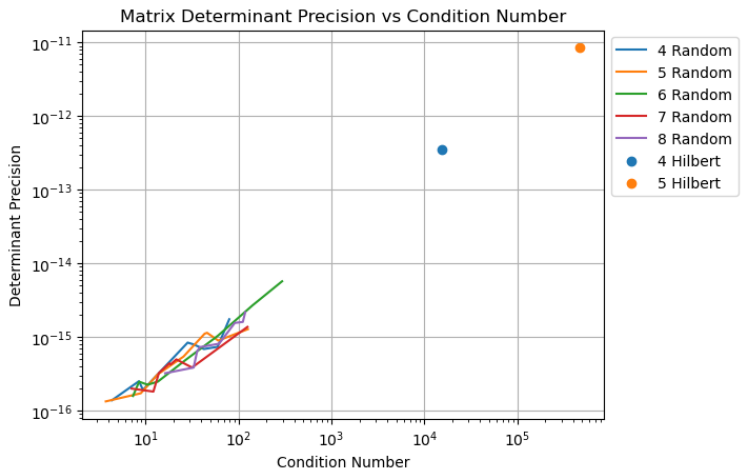

Figure 22: The linear correlation between the precision of a matrix determinant to its condition number. The x-axis shows the condition number, while the y axis shows the precision of the determinant. The legend shows the size of the matrix, as well as the type of the matrix as *Random* for randomly generated matrix, and *Hilbert* as the Hilbert matrix.

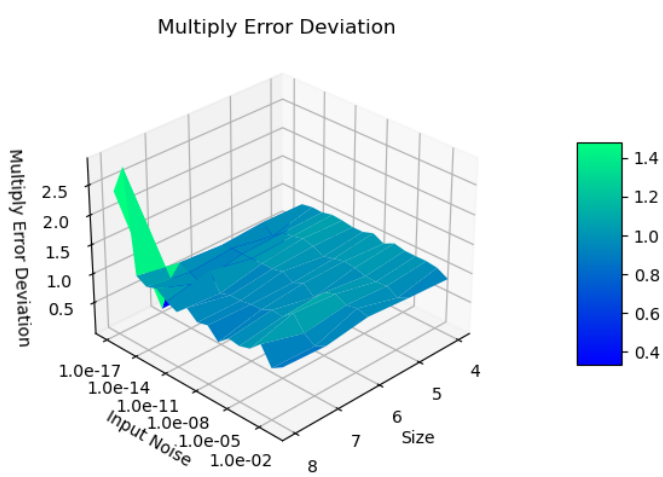

Figure 23: The error deviation of the first approximation calculation of $|\boldsymbol{M}|$ vs. input noise precision and matrix size.

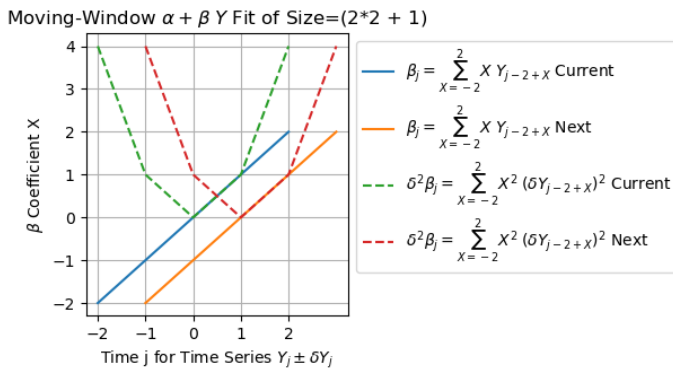

Figure 24: Coefficients of X in Formula (6.4) and (6.8) at a current and the next position in a time series of the $\alpha + \beta\ Y$ fit. It shows that the β_{j+1} in Formula (6.4) can be calculated progressively from β_j, as Formula (6.6) and (6.5) in order. It also shows that the $\delta^2\beta_{j+1}$ in Formula (6.8) cannot be calculated easily from $\delta^2\beta_j$.

6 Moving-Window Linear Regression

6.1 Moving-Window Linear Regression Algorithm [38]

Formula (6.1) and (6.2) give 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].

$$\alpha = \frac{\sum_j Y_j}{\sum_j 1}; \tag{6.1}$$

$$\beta = \frac{\sum_j X_j Y_j \ \sum_j 1 - \sum_j X_j \ \sum_j Y_j}{\sum_j X_j X_j \ \sum_j 1 - \sum_j X_j \ \sum_j X_j}; \tag{6.2}$$

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 Formula (6.1) and (6.2) into Formula (6.3) and (6.4), respectively:

$$\alpha_j = \alpha\ 2H = \sum_{X=-H+1}^{H} Y_{j-H+X}; \tag{6.3}$$

$$\beta_j = \beta\ \frac{H(H+1)(2H+1)}{3} = \sum_{X=-H}^{H} X Y_{j-H+X}; \tag{6.4}$$

According to Figure 24, in which H takes an example value of 2, the calculation of (α_j, β_j) can be obtained from the previous values of $(\alpha_{j-1}, \beta_{j-1})$, to reduce the

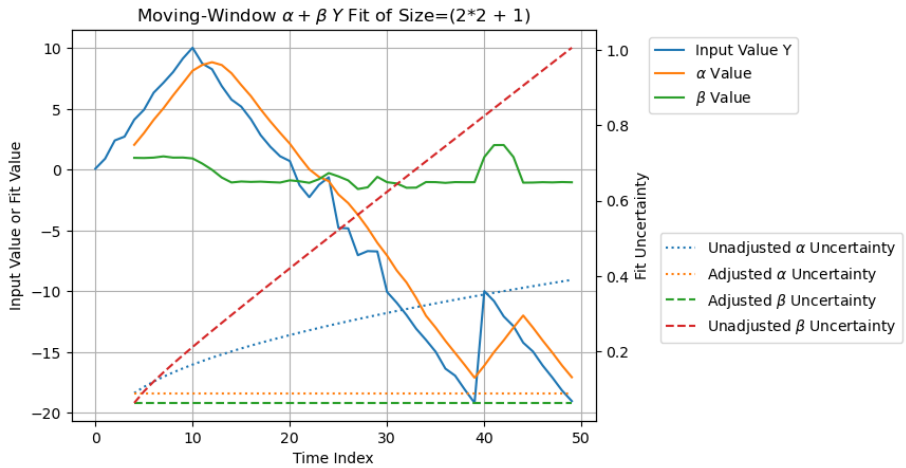

Figure 25: The result of fitting $\alpha + \beta\ Y$ to a time-series input Y within a moving window of size $2 * 2 + 1$. The x-axis marks the time index. The y-axis on the left is for the value of Y, α, and β, while the y-axis on the left is for the uncertainty of α and β. The uncertainty for Y is a constant of 0.2. In the legend, *Unadjusted* means the result of applying Formula (6.5) and (6.6) directly using variance arithmetic, while *Adjusted* means using Formula (6.5) and (6.6) for α and β values but using Formula (6.7) and (6.8) for α and β variances.

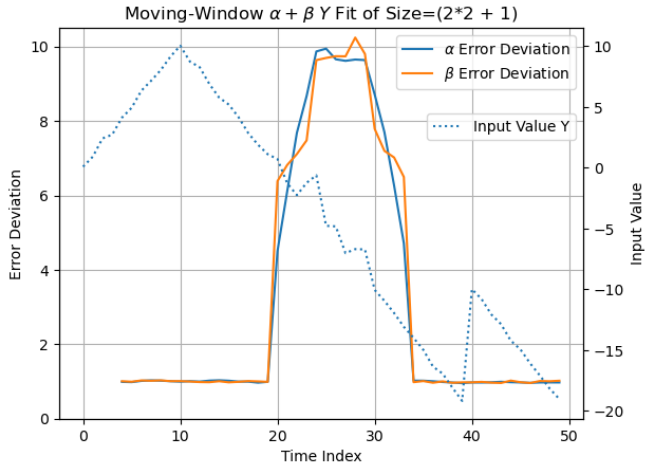

Figure 26: The error deviations of the $\alpha + \beta\ Y$ fit vs time index. The x-axis marks the time index. The y-axis on the left is for error deviation. An input time-series signal Y is also provided for reference, whose value is marked by the y-axis on the right.

calculation of Formula (6.3) and (6.4) into a progressive moving-window calculation of Formula (6.5) and (6.6), respectively:

$$\beta_j = \beta_{j-1} - \alpha_{j-1} + H\left(Y_{j-2H-1} + Y_j\right); \tag{6.5}$$

$$\alpha_j = \alpha_{j-1} - Y_{j-2H-1} + Y_j; \tag{6.6}$$

6.2 Variance Adjustment

When the time series contains uncertainty, the direct application of Formula (6.5) and (6.6) will result in loss of precision, because both formulas use each input multiple times and accumulate the variance of the input each time. Thus, the values for α_j and β_j should be calculated progressively using Formula (6.6) and (6.5), while the variance should be calculated using Formula (6.7) and (6.8), respectively. Formula (6.8) is no longer progressive.

$$\delta^2\alpha_j = \sum_{X=-H+1}^{H} (\delta Y_{j-H+X})^2 = \delta^2\alpha_{j-1} + (\delta Y_j)^2 - (\delta Y_{j-2H})^2; \tag{6.7}$$

$$\delta^2\beta_j = \sum_{X=-H+1}^{H} X^2(\delta Y_{j-H+X})^2; \tag{6.8}$$

Figure 25 shows that the input signal Y_j is composed of:

1. An increasing slope for $j = 0 \ldots 9$.
2. A decreasing slope for $j = 1 \ldots 39$.
3. A sudden jump with an intensity of $+10$ at $j = 40$
4. A decreasing slope for $j = 41 \ldots 49$.

For each increase of j, the increasing rate and the decreasing rate are $+1$ and -1, respectively.

The specified input uncertainty is always 0.2. Normal noises of 0.2 intensity are added to the slopes, except Normal noises of 2 intensity are added to the slope for $j = 10 \ldots 19$, during which the actual noises is 10-fold of the specified value.

Figure 25 also shows the result of moving window fitting of $\alpha + \beta\, Y$ vs time index j. The result values of α and β behave correctly with the expected delay in j. When Formula (6.3) and (6.4) are used both for values and uncertainties for α and β, the result uncertainties of α and β both increase exponentially with the index time j. When Formula (6.3) and (6.4) are used only for values, while Formula (6.7) and (6.8) are used for variance for α and β, the result uncertainty of α is $\sqrt{\frac{1}{2H+1}}\delta Y$, and the result uncertainty of β is $\sqrt{\frac{3}{H(H+1)(2H+1)}}\delta Y$, both of which are less than the input uncertainty δY, due to the averaging effect of the moving window.

6.3 Unspecified Input Error

To obtain the error deviation of α and β, the fitting is done on many time-series data, each with independent generated noises. Figure 26 shows the corresponding error deviation vs index time j, which are 1 except near 10 for $j = 10 \ldots 19$ when the actual noise is 10-fold of the specified one. It shows that a larger than 1 error deviation may probably indicate unspecified additional input errors other than rounding errors, such as the numerical errors from math libraries.

7 Math Library Functions

Formula (2.36), (2.40), (2.44), and (2.48) 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 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 the same underlying distribution. The deviation of the value errors is defined as the *value deviation*, which the uncertainty should match.

7.1 Exponential

Figure 27 shows that the calculated uncertainties using Formula (2.36) 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 though both the uncertainties and the value error deviations increase exponentially with x and δx. Such strong tracking power holds for all x and δx of $e^{x+\delta x}$.

7.2 Logarithm

The probability density function for $\log(x)$ has a pole at $x = 0$.

Figure 28 shows that the calculated uncertainties using Formula (2.40) 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 variance arithmetic invalidates the calculation when the input precision is worse 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. In Figure 28 the 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.

Such divergences of the result uncertainties are expected when the input precision is worse than the estimated applicable precision threshold of $1/\sigma$, in which $\sigma = 5$ is the bonding factor of 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 (2.31). 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 implementation:

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.

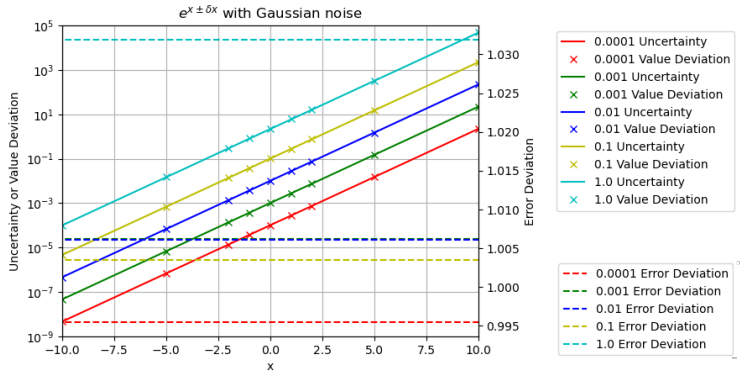

Figure 27: The calculated uncertainties vs the measured value deviations as well as the measured error deviations for $e^{x \pm \delta x}$, for different x as shown by the x-axis, and different δx as shown in the legend. The uncertainties and the value deviations are drawn using the logarithmic y scales on the left side, while the error deviations are drawn using the linear y scales on the right side. Each color represents a δx. Gaussian noises are used to produce the input noise δx.

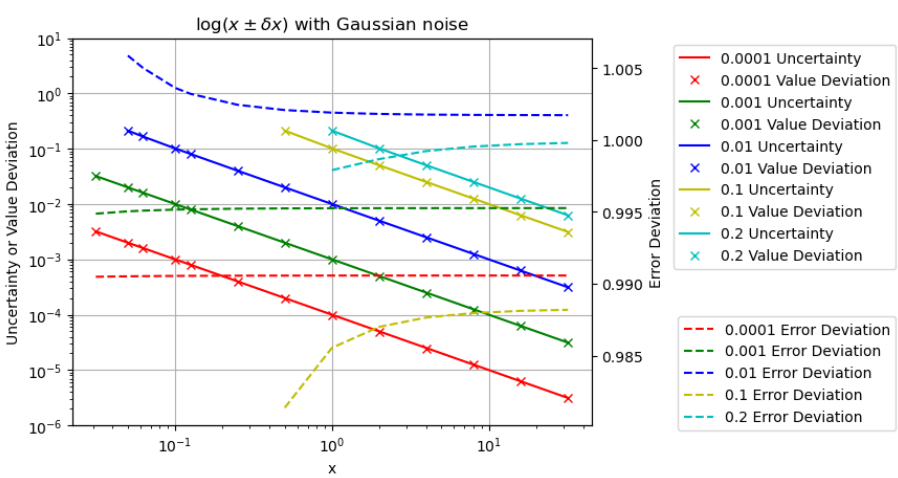

Figure 28: The calculated uncertainties vs the measured value deviations as well as the measured error deviations for $\log(x \pm \delta x)$, for different x as shown by the x-axis, and different δx as shown in the legend. The uncertainties and the value deviations are drawn using the logarithmic y scales on the left side, while the error deviations are drawn using the linear y scales on the right side. Each color represents a δx. Gaussian noises are used to produce the input noise δx.

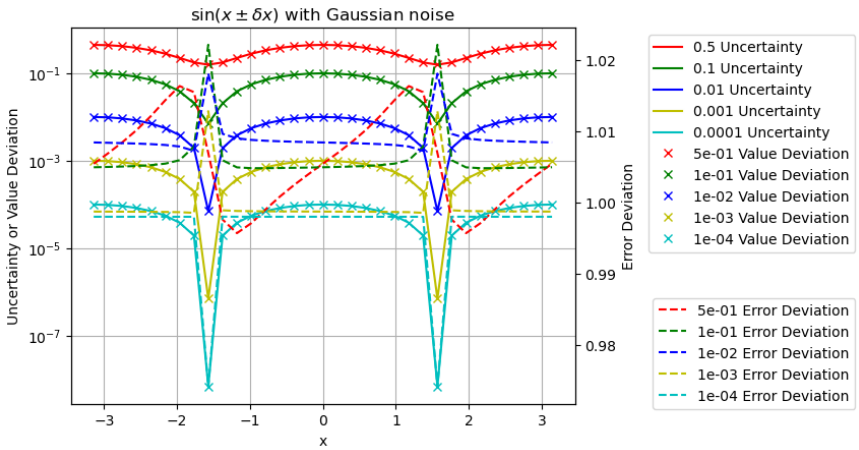

Figure 29: The calculated uncertainties vs the measured value deviations as well as the measured error deviations for $\sin(x \pm \delta x)$, for different x as shown by the x-axis, and different δx as shown in the legend. The uncertainties and the value deviations are drawn using the logarithmic y scales on the left side, while the error deviations are drawn using the linear y scales on the right side. Each color represents a δx. Gaussian noises are used to produce the input noise δx.

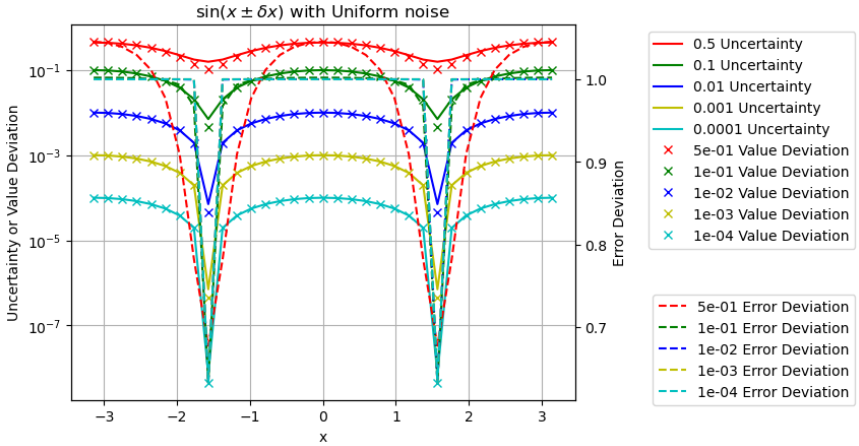

Figure 30: The calculated uncertainties vs the measured value deviations as well as the measured error deviations for $\sin(x \pm \delta x)$, for different x as shown by the x-axis, and different δx as shown in the legend. The uncertainties and the value deviations are drawn using the logarithmic y scales on the left side, while the error deviations are drawn using the linear y scales on the right side. Each color represents a δx. Uniform noises are used to produce the input noise δx.

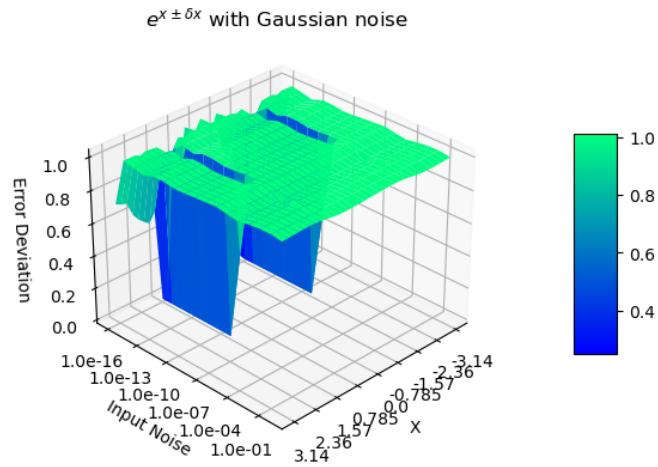

Figure 31: The error deviation for $\sin(x \pm \delta x)$ vs. x and δx. The x-axis is x between $-\pi$ and $+\pi$. The y-axis is δx between -10^{-16} and 1. The z-axis is the error deviation. Gaussian noises are used to produce the input noise δx.

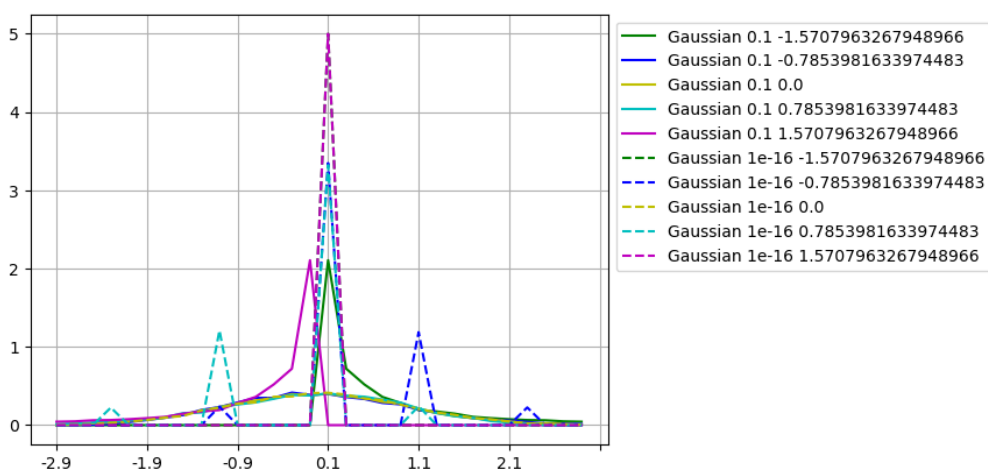

Figure 32: The histogram of the error deviation for $\sin(x \pm \delta x)$, for $x = -\pi/2, -\pi/4, 0, +\pi/4, +\pi/2$ and $\delta x = 10^{-1}, 10^{-16}$ as shown in the legend. The y-axis is for normalized count. Each color represents a different x, while each line pattern represents a different δx. Gaussian noises are used to produce the input noise δx.

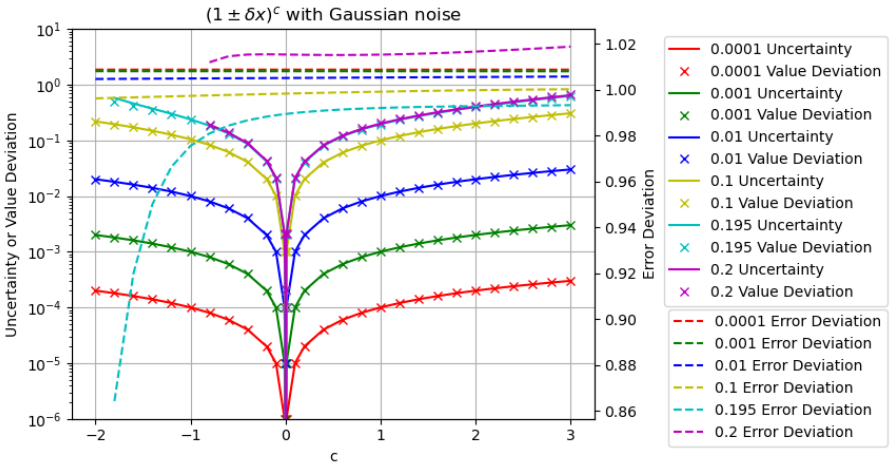

Figure 33: The calculated uncertainties vs the measured value deviations as well as the measured error deviations for $(1 \pm \delta x)^c$, for different c as shown by the x-axis, and for different δx as shown in the legend. The uncertainties and the value deviations are drawn using the logarithmic y scales on the left side, while the error deviations are drawn using the linear y scales on the right side. Each color represents a δx.

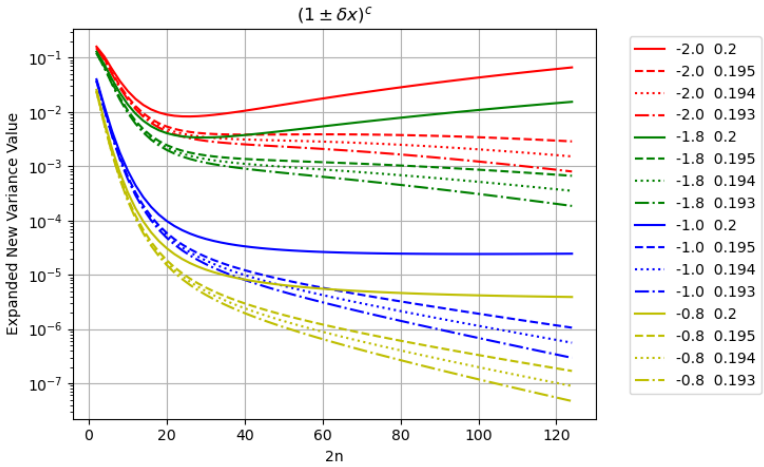

Figure 34: The individual variance term of $(1 \pm \delta x)^c$ vs the different Taylor expansion order $2n$, for different c and δx as shown in the legend. The x-axis is Taylor expansion order $2n$ from 1 to 400 in Formula (2.23). The y-axis is the contribution to $(1 \pm \delta x)^c$ at each Taylor expansion order $2n$ according to Formula (2.48). It shows that $(1 \pm 0.2)^{-2}$ and $(1 \pm 0.2)^{-1.5}$ clearly diverge, $(1 \pm 0.2)^{-1}$ likely diverges, and other cases clearly converge.

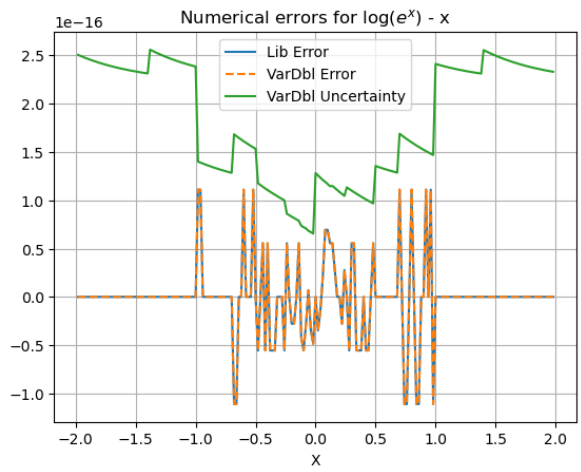

Figure 35: The values and uncertainties of $\log(e^x) - x$ vs x, as *VarDbl Error* and *VarDbl Uncertainty* in the legend. The result of the same calculation using conventional floating-point library functions is shown as *Lib Error* in the legend.

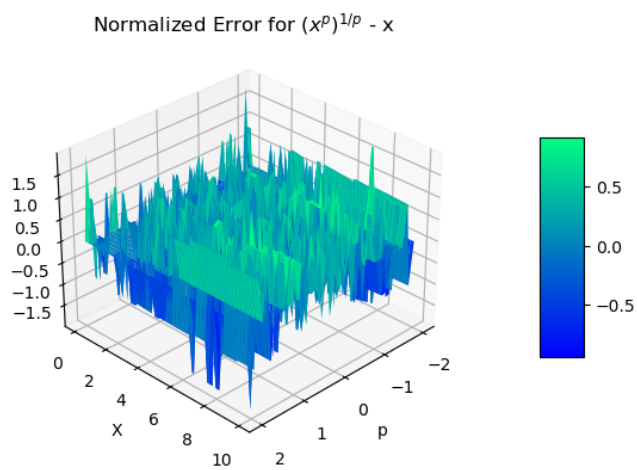

Figure 36: The normalized errors of $(x^p)^{\frac{1}{p}} - x$ vs x and p.

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 $\log(x + \delta x)$.

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

7.3 Sine

Figure 29 shows that the calculated uncertainties using Formula (2.44) agree very well with the measured value deviations for $\sin(x+\delta x)$. Figure 29 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 29, Gaussian noises are used, while in Figure 30, the input noises are uniformly distributed. The difference of Figure 29 and 30 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 31 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 32 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 larger 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. Both the result uncertainty and the error deviation at the zero become 0. The corresponding histogram is also distorted from Gaussian, with the peak at $x = \pm\pi/2$, and all other values on one side of the peak only.

When the function is flat, both the result uncertainty and the error deviation become 0.

7.4 Power

Figure 33 shows that the calculated uncertainties of $(1 \pm \delta x)^c$ using Formula (2.48) agree very well with the measured value deviations for $(1 + \delta x)^c$.

Figure 34 shows the individual variance term of $(1 \pm \delta x)^c$ for the different Taylor expansion order $2n$, c, and δx. It shows that $(1 \pm 0.2)^{-2}$ clearly diverges, while $(1 \pm 0.2)^{-2}$ marginally diverges, so that variance arithmetic invalidate both using the monotonic requirements. When $\delta x = 0.2$, variance arithmetic limits c above -0.8. For the same reason, when $\delta x = 0.195$, variance arithmetic limits c above -1.8.

7.5 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 35 shows that using either 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 variance arithmetic bound 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 36 shows that the normalized errors are not specific to either x or p, resulting in an error deviation 0.548 ± 0.8.

The numerical errors of the library functions $\sin(x)$, $\cos(x)$, and $\tan(x)$ will be studied in detail later in Section 8.

7.6 Summary

Formula (2.36), (2.40), (2.44), and (2.48) give 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.

The convergence of the result variance calculation can be judged numerically.

8 Fast Fourier Transformation

8.1 Unfaithful Frequency Response of Discrete Fourier Transformation [38]

Each testing algorithm needs to come under careful scrutiny. One important issue is whether the digital implementation of the algorithm is faithful to 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 (8.1) and (8.2), respectively [12], in which j is the *index time* and n is the *index frequency* for the discrete Fourier transformation [10]

$$H[n] = \sum_{k=0}^{N-1} h[k]\, e^{i2\pi kn/N}; \tag{8.1}$$

$$h[k] = \frac{1}{N} \sum_{n=0}^{N-1} H[n]\, e^{-i2\pi nk/N}; \tag{8.2}$$

Formula (8.3) 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 (8.3). Examples of unfaithful representations of fractional frequency by the discrete Fourier transformation are shown in Figure 37. The data for Figure 37 is generated using *SciPy*, which are very reproducible using any other math libraries, including *MathLab* and *Mathematica*.

$$\begin{aligned} H[n] &= \sum_{k=0}^{N-1} \sin(2\pi nk/N)\, e^{i2\pi nk/N} = \frac{1}{2i}\left(\sum_{k=0}^{N-1} e^{i2\pi(n+f)\frac{k}{N}} - \sum_{k=0}^{N-1} e^{i2\pi(n-f)\frac{k}{N}}\right) \\ &= \begin{cases} iN/2, & f \text{ is integer} \\ N/\pi, & f \text{ is integer} + 1/2 \\ \frac{1}{2}\frac{\sin(2\pi f - 2\pi\frac{f}{N}) + \sin(2\pi\frac{f}{N}) - \sin(2\pi f)e^{-i2\pi\frac{n}{N}}}{\cos(2\pi\frac{n}{N}) - \cos(2\pi\frac{f}{N})} & \text{otherwise} \end{cases} \end{aligned} \tag{8.3}$$

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. 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

[10] 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.

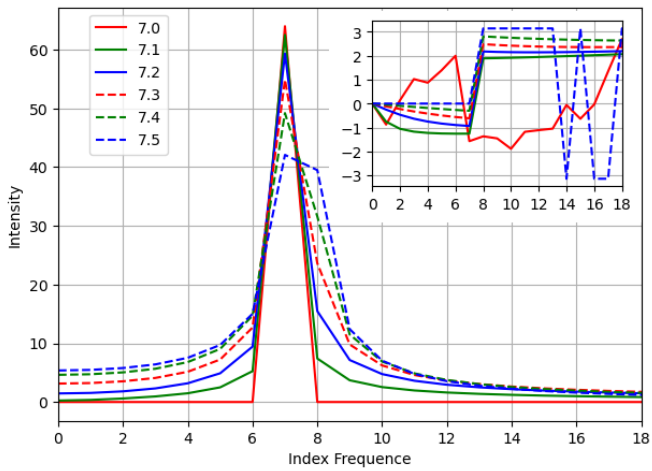

Figure 37: An unfaithful Discrete Fourier transformation is demonstrated by the spectra of a few sine signals having amplitude of 1 and slightly different frequencies as shown in legends. The x-axis is for index frequency. The y-axis is for intensity, while the y-axis of the embedded picture is for phase. This figure is a practical reproduction of a previous theoretical figure [38].

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:

$$h(t) \Leftrightarrow H(s): \quad h(t-\tau) \Leftrightarrow H(s)e^{i2\pi s\tau}; \tag{8.4}$$

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

$$G[n] = (H[n] + h[N] - h[0])e^{i2\pi n/N}; \tag{8.5}$$

Applying Formula (8.4) to Formula (8.5) results in Formula (8.6).

$$h[N] = h[0]; \tag{8.6}$$

Thus, the discrete Fourier transformation has an implied assumption that the signal $h[k]$ repeats itself outside the region of $[0, N-1]$ [43]. 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 been addressed seriously enough previously. Because the discrete Fourier transformation has widest applications in science and engineering, this problem needs some serious attention.

8.2 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 (2.23), 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 increases the variance by 2-fold, so that the result uncertainty mean increases with the FFT order L as $\sqrt{2}^L$. Because the reverse transformation divides the result by 2^L, the result uncertainty mean decreases with the FFT order L as $\sqrt{1/2}^L$. The result uncertainty means for the roundtrip transformations is thus: $\sqrt{2}^L \times \sqrt{1/2}^L = 1$.
- Forward and reverse transformations are identical except for a sign, so they are essentially the same algorithm, and their difference is purely due to input data.

Forward and reverse FFT transforms have data difference:

- 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, reverse FFT 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.

8.3 Testing Signals

Only Formula (8.1) and (8.2) 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 (8.7).

$$y \equiv i2\pi\frac{n}{N}: \quad G(y) = \sum_{k=0}^{N-1} e^{yk} = \frac{e^{Ny}-1}{e^y-1} = \begin{cases} y = 0: & N \\ y \neq 0: & 0 \end{cases} ;$$

$$H[n] = \frac{dG}{dy} = \begin{cases} n = 0: & \frac{N(N-1)}{2} \\ n \neq 0: & -\frac{N}{2}(1 + i/\tan(n\frac{\pi}{N})) \end{cases} ; \tag{8.7}$$

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.

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

8.4 Library Errors

Formula (8.1) and (8.2) 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 [11]

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 38 shows that the value difference between the library $\sin(x)$ and the indexed $\sin(x)$ increases with increasing x. However, checked by $\sin(x)^2 + \cos(x)^2 - 1$, Figure 39 shows that the value errors are quite comparable between the indexed and library $\sin(x)$ and $\cos(x)$ functions. In Figure 39, the deviations for the indexed sine function seem less stable statistically, because the deviations of the index sine function are calculated from much less samples: For the FFT order N, the deviations for the indexed sine function contain 2^{N-3} data covering $x \in [0, \pi/4]$, while the deviations for the library sine function contain 2^{2N-1} data covering $x \in [0, \pi 2^N]$. Figure 39 also shows that both $\sin(x)$ functions have proper coverage in variance arithmetic. Thus, the difference is just caused by the rounding of x in $\sin(x)$ and $\cos(x)$ library functions.

Figure 40 shows the value difference between the library $\cos(x)/\sin(x)$ and the indexed $\cos(x)/\sin(x)$ 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. It is likely that the rounding error of x causes the numerical error when $\cos(x)/\sin(x)$ becomes very large near $x = \pi$. According to Figure 41, the library $\cos(x)/\sin(x)$ function does not have proper coverage in variance arithmetic.

Using indexed $\sin(x)$ as standard, Figure 41 compares the value errors of the library $\sin(x)$, $\cos(x)/\sin(x)$, and $1/tan(x)$ for increasing FFT orders. It shows that the numerical errors of all these library functions grow exponentially with increasing FFT order. For all the three library functions, $1/tan(x)$ has the largest errors, and it will not be used for FFT transformations. $\cos(x)/\sin(x)$ has larger errors than those of $\sin(x)$.

The difference between the indexed and library sine functions shows the inconsistency in the library sine functions, because the indexed sine functions is just the library sine functions in the range of $[0, \pi/4]$, and should be related to other ranges according to precise mathematical relations. It is difficult to study the numerical errors of library functions using conventional floating-point arithmetic which has no concept of imprecise values. Having identified the nature and the strength of the numerical errors of the library $\sin(x)$ and $\cos(x)/\sin(x)$, variance arithmetic can show their effects using FFT transformations.

8.5 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:

[11] The sin and cos library functions in Python, C++, Java all behave in the same way.

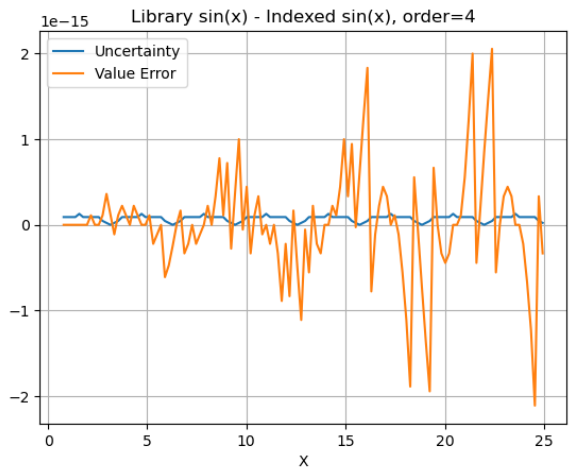

Figure 38: The difference between the library $\sin(x)$ and the indexed $\sin(x)$, for all integer input to the indexed sine functions used in the FFT transformations of FFT order 4. The uncertainties of the $\sin(x)$ values are also displayed, to mark the periodicity of π.

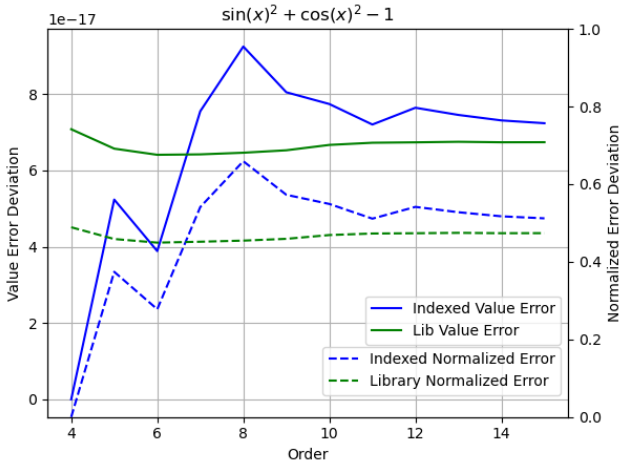

Figure 39: The value error deviation of $\sin(x)$ and $\cos(x)$ checked by $\sin(x)^2 + \cos(x)^2 - 1$ for different FFT order as shown in the x-axis, and for the indexed and library versions as shown in the legend. Also shown is the corresponding normalized error deviation for the indexed and library $\sin(x)$. The y-axis on the left is for value error deviation, while the y-axis on the right is for normalized error deviation.

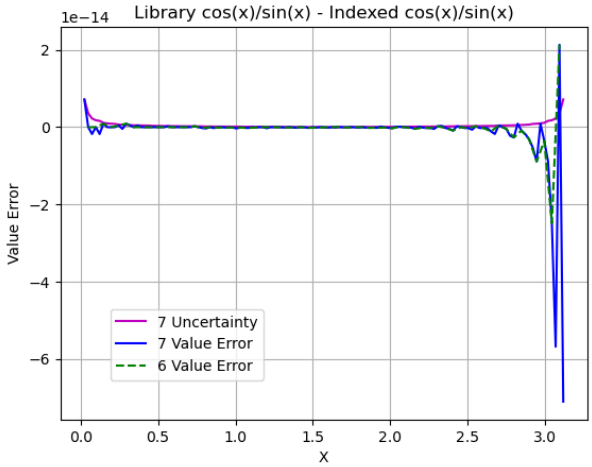

Figure 40: The difference between the library $\cos(x)/\sin(x)$ and the indexed $\cos(x)/\sin(x)$, for $x \in (0, \pi)$, for different FFT order as shown in the legend. In the legend, *Uncertainty* is the calculated uncertainty assuming that both $\cos(x)$ and $\sin(x)$ are imprecise in their least significant values, and *Value Error* is the difference between the library $\cos(x)/\sin(x)$ and the indexed $\cos(x)/\sin(x)$.

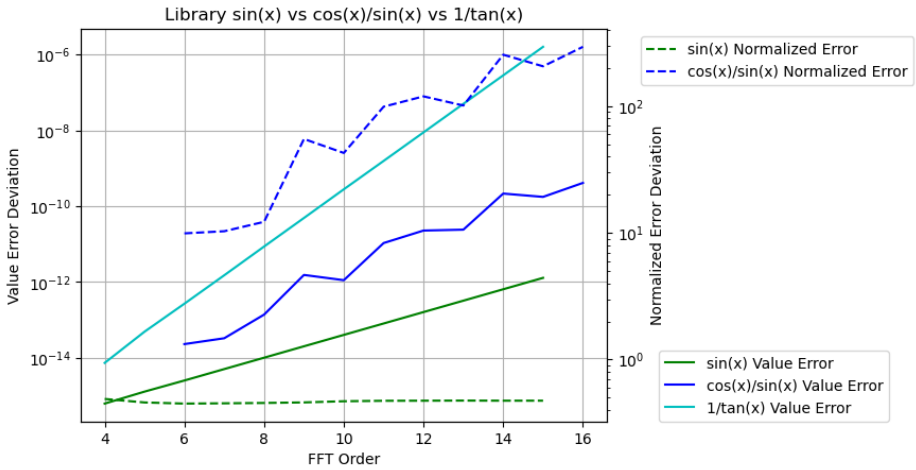

Figure 41: Comparing library $\sin(x)$ and $\cos(x)/\sin(x)$ for different FFT orders as shown by x-axis, and for either value error deviations or normalized error deviations, as shown in the legend. The y-axis on the left is for value error deviations, while the y-axis on the right is for normalized error deviations.

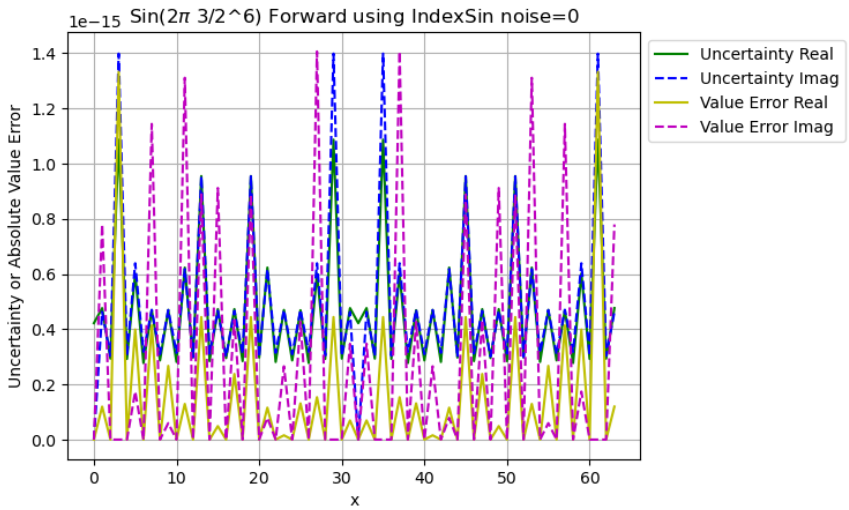

Figure 42: The FFT spectrum of $\sin(j3/2^6\pi)$ using the indexed sine functions after the forward transformation calculated by variance arithmetic, with the uncertainty and the value errors shown in the legend. The x-axis is for index frequency. The y-axis is for uncertainty and absolute value error.

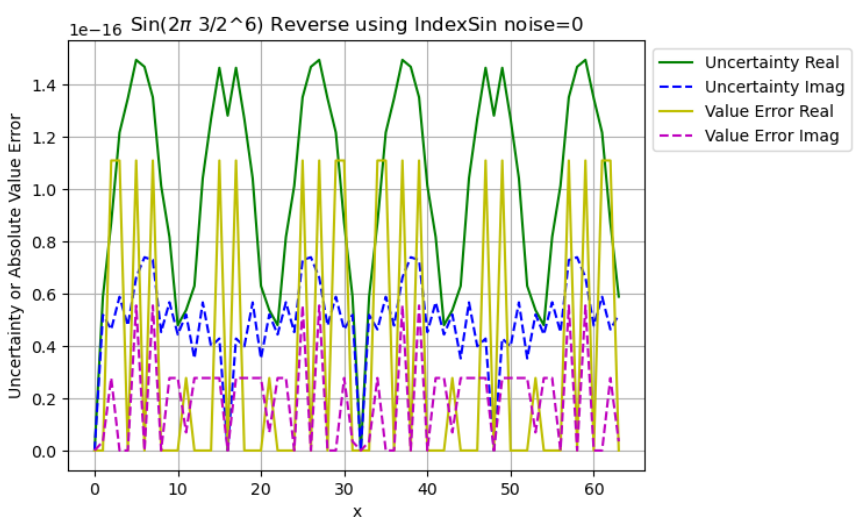

Figure 43: The FFT waveform of $\sin(j3/2^6\pi)$ using the indexed sine functions after the reverse transformation calculated by variance arithmetic, with the uncertainty and the value errors shown in the legend. The x-axis is for index time. The y-axis is for uncertainty and absolute value error.

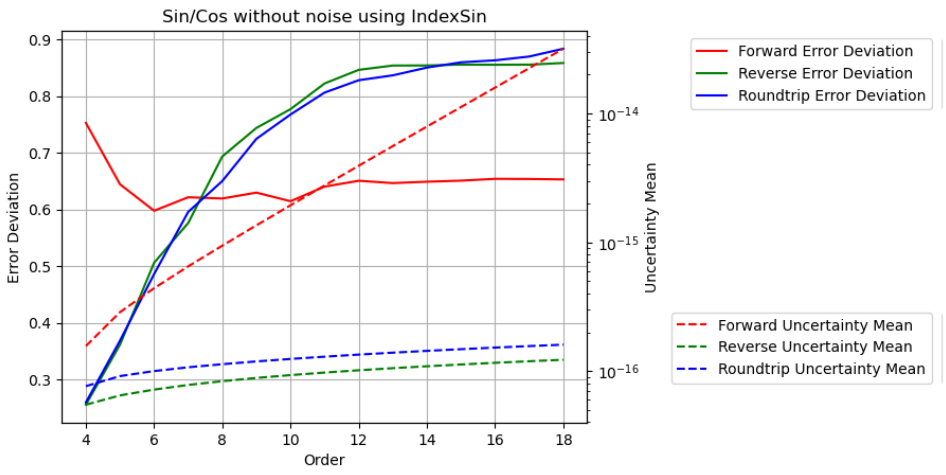

Figure 44: The result error deviation and uncertainty mean of Sin/Cos signals vs. FFT order using the indexed sine functions for forward, reverse and roundtrip FFT transformations, as shown in the legend. The y-axis on the left is for error deviation, while the y-axis on the right is for uncertainty mean.

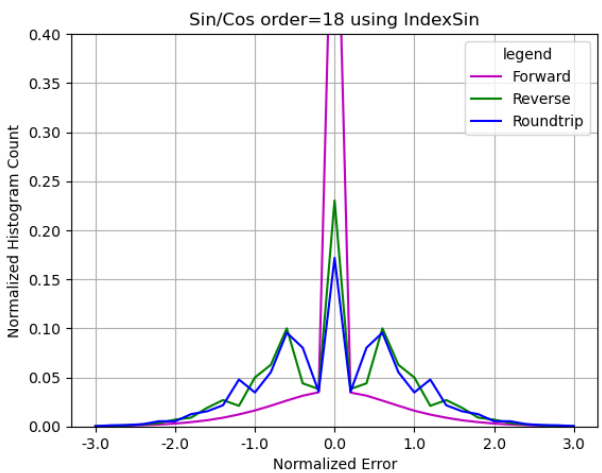

Figure 45: The histograms of the normalized errors of Sin/Cos signals using the indexed sine functions for forward, reverse and roundtrip FFT transformations, as shown in the legend. The FFT order is 18.

- Figure 42 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 43 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 44 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 45 shows the corresponding histograms at FFT order 18.

- As expected, the uncertainties grow much faster with 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 44 results in almost Gaussian distribution of the normalized errors in Figure 45. 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 44 results in structured distribution on top of a Gaussian distribution of the normalized errors in Figure 45. 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 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 increasing FFT orders until become stable when FFT orders is 12 or more. When the FFT order is about 12, statistical stability is reached.

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

8.6 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 have numerical calculation errors which is not specified by the input uncertainties of variance arithmetic. The question is whether variance arithmetic can track these 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 46 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 the real part, and 5.6 for the imaginary part. As expected, the error deviations are noticeably larger than their counterparts in Figure 42.
- Figure 47 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 of 8 where the value error is at a local maximum.

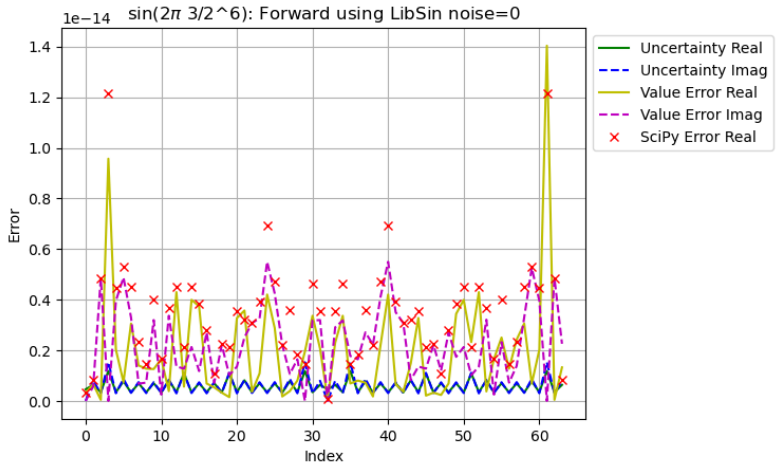

Figure 46: The FFT spectrum of $\sin(j3/2^6\pi)$ using the library sine functions after the forward transformation calculated by variance arithmetic, with the uncertainties and the value errors shown in the legend. Also included are the corresponding result value errors using *SciPy*. The x-axis is for index frequency. The y-axis is for uncertainties or absolute value errors.

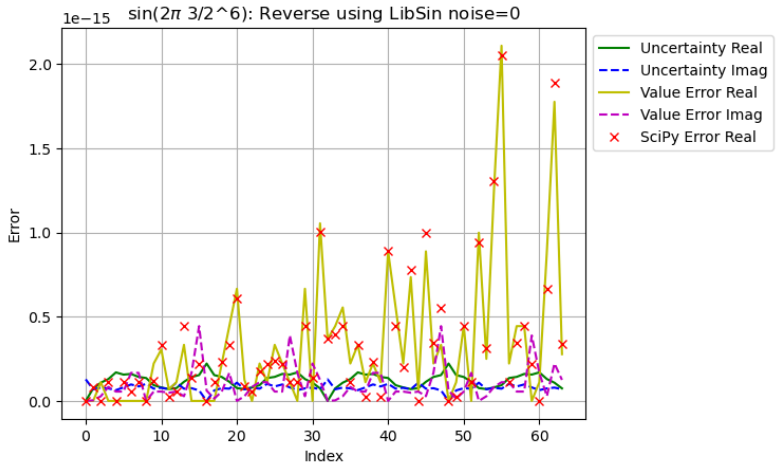

Figure 47: The FFT waveform of $\sin(j3/2^6\pi)$ using the indexed sine functions after the reverse transformation calculated by variance arithmetic, with the uncertainties and the value errors shown in the legend. Also included are the corresponding result value errors using *SciPy*. The x-axis is for index time. The y-axis is for uncertainties or absolute value errors.

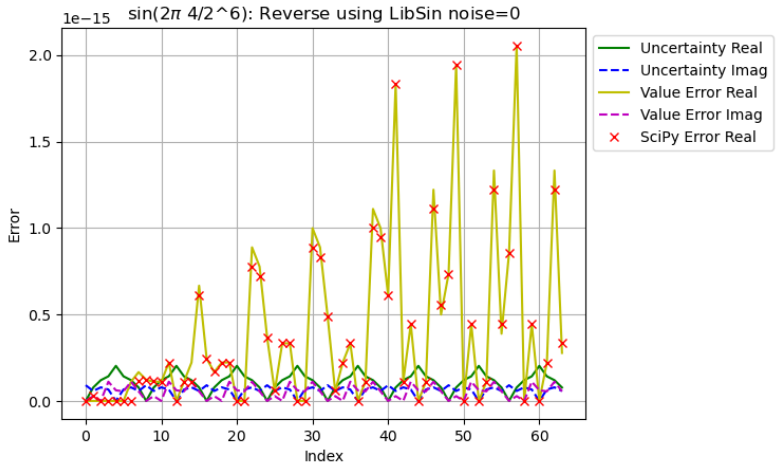

Figure 48: The waveform of $\sin(j4/2^6\pi)$ using the indexed sine functions after the reverse transformation calculated by variance arithmetic, with the uncertainties and the value errors shown in the legend. Also included are the corresponding result value errors using *SciPy*. The x-axis is for index time. The y-axis is for uncertainties or absolute value errors.

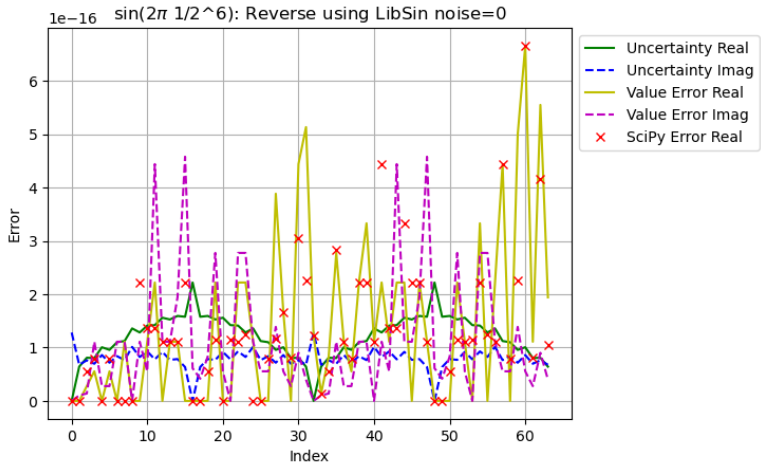

Figure 49: The waveform of $\sin(j1/2^6\pi)$ using the indexed sine functions after the reverse transformation calculated by variance arithmetic, with the uncertainties and the value errors shown in the legend. Also included are the corresponding result value errors using *SciPy*. The x-axis is for index time. The y-axis is for uncertainties or absolute value errors.

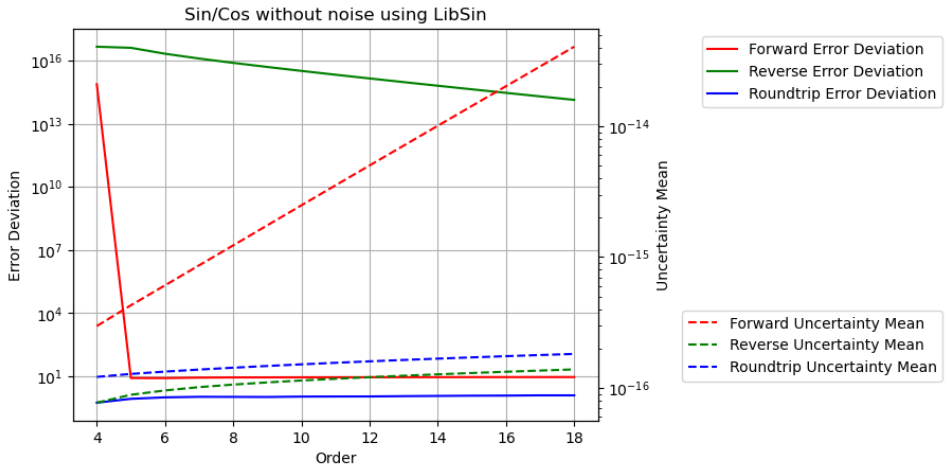

Figure 50: The result error deviation and uncertainty mean of Sin/Cos signals vs. FFT order using the library sine functions for forward, reverse and roundtrip FFT transformations, as shown in the legend. The y-axis on the left is for error deviation, while the y-axis on the right is for uncertainty mean.

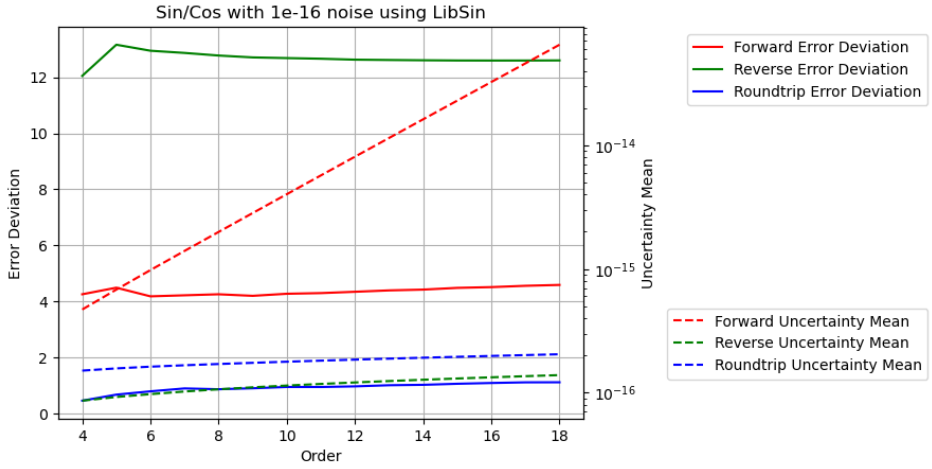

Figure 51: The result error deviation and uncertainty mean of Sin/Cos signals vs. FFT order using the library sine functions for forward, reverse and roundtrip FFT transformations, as shown in the legend. The y-axis on the left is for error deviation, while the y-axis on the right is for uncertainty mean. 10^{-16} input noises are added to the Sin/Cos signals.

- As the result of the huge error deviation, Figure 50 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 51.

To validate the FFT implementation in variance arithmetic, the results are compared with the corresponding calculations using the python numerical library *SciPy* in Figure 46 and 47. The results are quite comparable, except that in *SciPy*, the data waveform is always real rather than complex. In Figure 46, the *SciPy* results have been made identical for the frequency indexes f and $-f$, and the matching to the corresponding results of variance arithmetic is indicative. In Figure 47, the *SciPy* results match the corresponding results of variance arithmetic well.

In Figure 47, the value errors tend to increase with the index time, which is due to the periodic increase of the numerical errors of $\sin(x)$ with x as shown in Figure 38. 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 49, 47, and 48. In fact, Figure 48 suggests that the periodic numerical errors in Figure 38 resonate with the periodicity of the input *Sin* signal. In contrast, Figure 43 shows no such increase, because the indexed sine functions contain no library errors as described in Figure 38.

Variance arithmetic reveals the reason why to add small noises to the numerically generated input data, which is already a common practice [12]. Variance arithmetic further reveals that the goal to add small noises is to achieve proper coverage. When the proper tracking of variance arithmetic is not achieved, the result may contain large amount of errors.

8.7 Linear Signal

As shown in Figure 41, linear signals introduce more numerical errors to the result. The question is whether variance arithmetic can track these additional numerical errors or not.

Using the indexed sine functions:

- Figure 52 shows that proper coverage can be achieved for all FFT transformations for all FFT orders.
- Figure 53 shows that for each transformation, the histogram of the Linear signal is similar to the counterpart of the Sin/Cos signals in Figure 45.

Using the library sine functions:

- Figure 54 shows that proper coverage cannot be achieved any FFT transformation for all FFT orders, because the value errors outpace the uncertainties with increasing FFT order. Because of this, adding noise to the input cannot achieve proper coverage.
- Figure 55 shows that the reverse histogram contains additional peaks in addition to its counterpart in Figure 54. The additional peaks extend beyond the range of $[-3, +3]$

When the input uncertainty is too small, proper coverage cannot be achieved, which is the case when the $\sin(x)$ numerical errors in Figure 38 is combined with

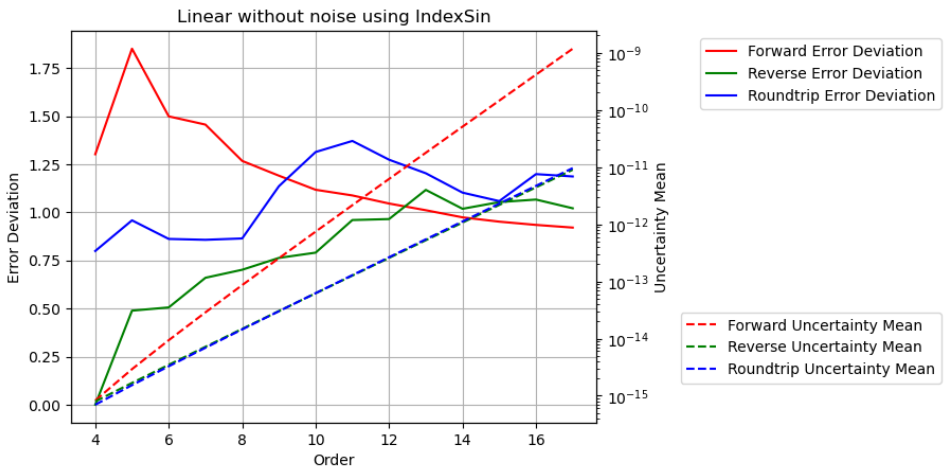

Figure 52: The result error deviation and uncertainty mean of Linear signal vs. FFT order using the library sine functions for forward, reverse and roundtrip FFT transformations, as shown in the legend. The y-axis on the left is for error deviation, while the y-axis on the right is for uncertainty mean.

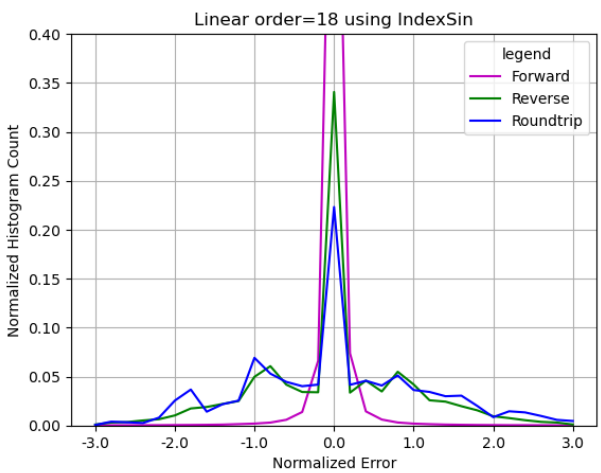

Figure 53: The histograms of the normalized errors of Linear signal using the indexed sine functions for forward, reverse and roundtrip FFT transformations, as shown in the legend.

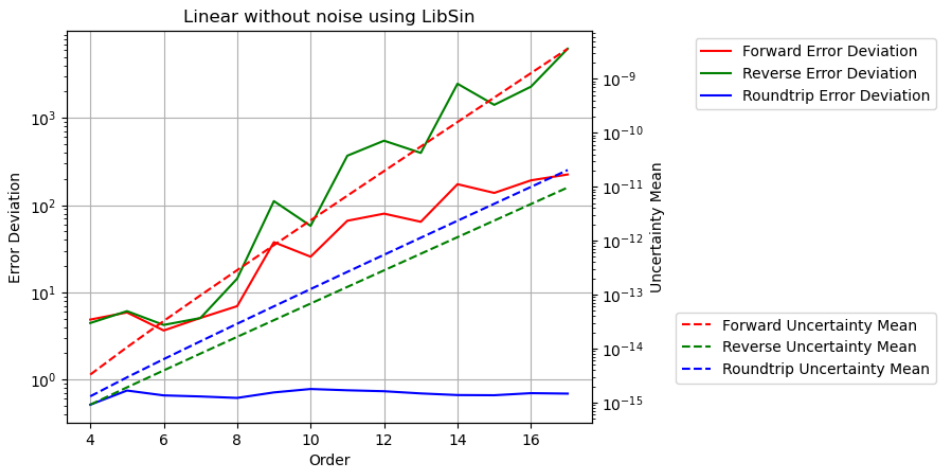

Figure 54: The result error deviation and uncertainty mean of Linear signal vs. FFT order using the library sine functions for forward, reverse and roundtrip FFT transformations, as shown in the legend. The y-axis on the left is for error deviation, while the y-axis on the right is for uncertainty mean.

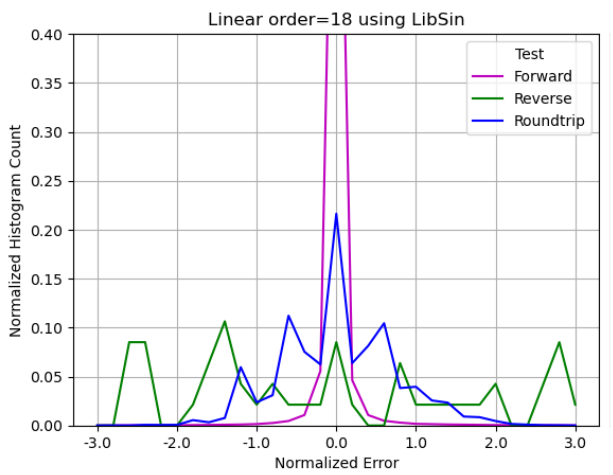

Figure 55: The histograms of the normalized errors of Linear signal using the library sine functions for forward, reverse and roundtrip FFT transformations, as shown in the legend.

the $\cos(x)/\sin(x)$ numerical errors in Figure 40. The only solution seems to create a new sin library using variance arithmetic so that all numerical calculation errors are accounted for.

8.8 Ideal Coverage

Even when proper coverage cannot be achieved, adding enough noise to the input can overpower the numerical calculation errors, to achieve ideal coverage. For Linear signal with 10^{-3} input noise, using the library sine functions, Figure 56 and Figure 57 show characteristics for 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 increasing FFT order.

However, the range of ideal coverage depends on how well the input uncertainty specifies the input noise.

For Linear signals using library sine functions, Figure 58 and 59 show the error deviation vs. the added noise vs. FFT order for the forward and reverse transformations, respectively. The ideal coverage is shown as the regions where the error deviations are 1. 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 numerical errors increase with the amount of calculation, the input noise range reduces with increasing FFT order. It is possible that ideal coverage is 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 algorithms that is very insensitive to input errors, FFT can breaks down 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 indexed sine functions, as shown by Figure 60 and 61 for the forward and reverse transformations, respectively, the ideal region is much larger, and proper coverage is achieved in other regions. Forward FFT transformation still shows larger region of ideal coverage than that of Reverse FFT transformation.

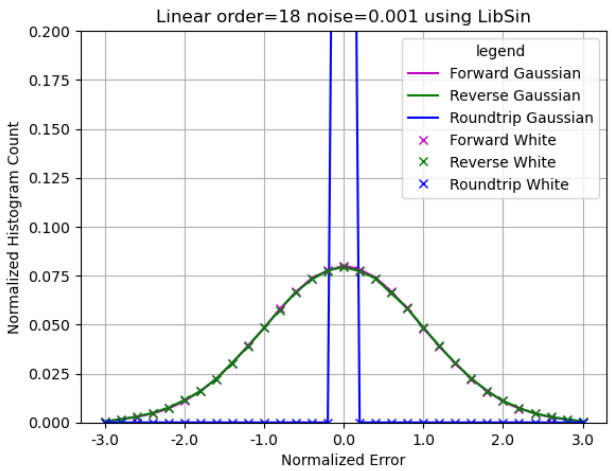

Figure 56: The histograms of the normalized errors of Linear signal with 10^{-3} input noise using the library sine functions for forward, reverse and roundtrip FFT transformations, as shown in the legend.

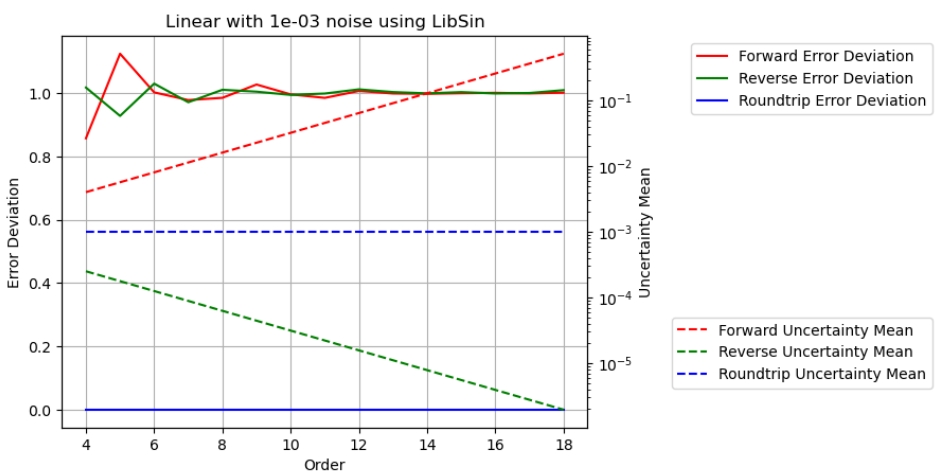

Figure 57: The result error deviation and uncertainty mean of Linear signal with 10^{-3} input noise using the library sine functions vs. FFT order for forward, reverse and roundtrip FFT transformations. The y-axis on the left is for error deviation, while the y-axis on the right is for uncertainty mean.

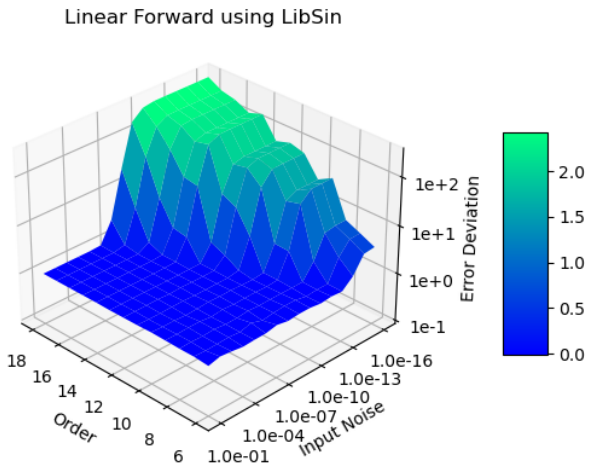

Figure 58: The result error deviations for Linear signals using library sine functions vs. input uncertainties and FFT orders for the forward transformations. The input uncertainties run from 10^{-16} to 10^{-1}, while the FFT Order runs from 6 to 18.

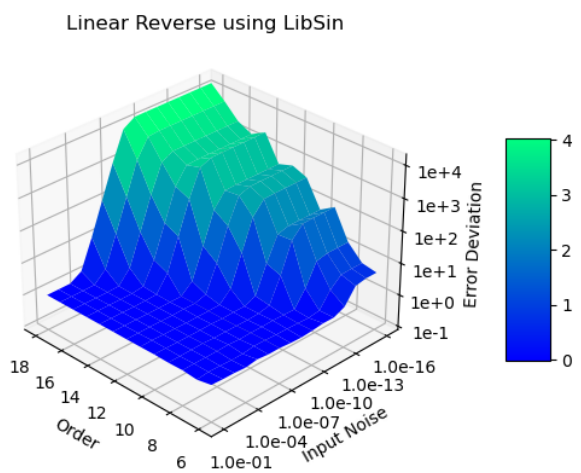

Figure 59: The result error deviations for Linear signals using library sine functions vs. input uncertainties and FFT orders for the reverse transformations. The input uncertainties run from 10^{-16} to 10^{-1}, while the FFT Order runs from 6 to 18.

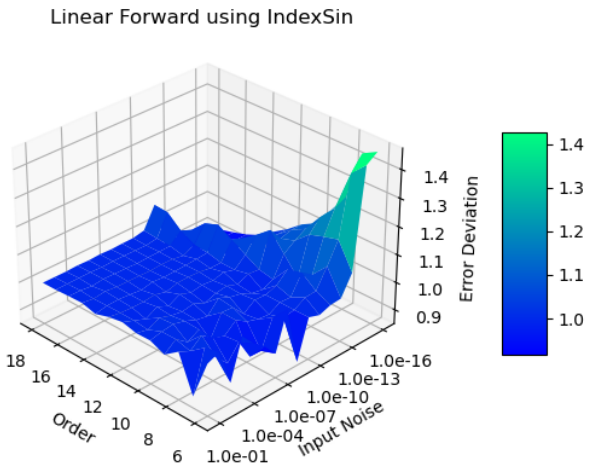

Figure 60: The result error deviations for Linear signals using indexed sine functions vs. input uncertainties and FFT orders for the forward transformations. The input uncertainties run from 10^{-16} to 10^{-1}, while the FFT Order runs from 6 to 18.

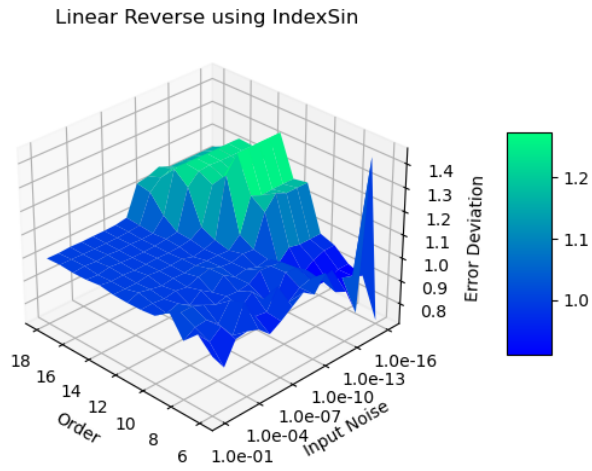

Figure 61: The result error deviations for Linear signals using indexed sine functions vs. input uncertainties and FFT orders for the reverse transformations. The input uncertainties run from 10^{-16} to 10^{-1}, while the FFT Order runs from 6 to 18.

For Sin/Cos using either indexed sine functions or library sine functions, the ideal region is achieved when the added noise is large enough to cover the effect of rounding errors, almost independent of FFT orders. In both cases, forward transformation needs 10^{-15} added noise, while reverse transformation needs 10^{-12} added noise. The difference is that using indexed sine functions, in the proper coverage region, error deviations deviate from 1 only slightly.

8.9 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 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.

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.

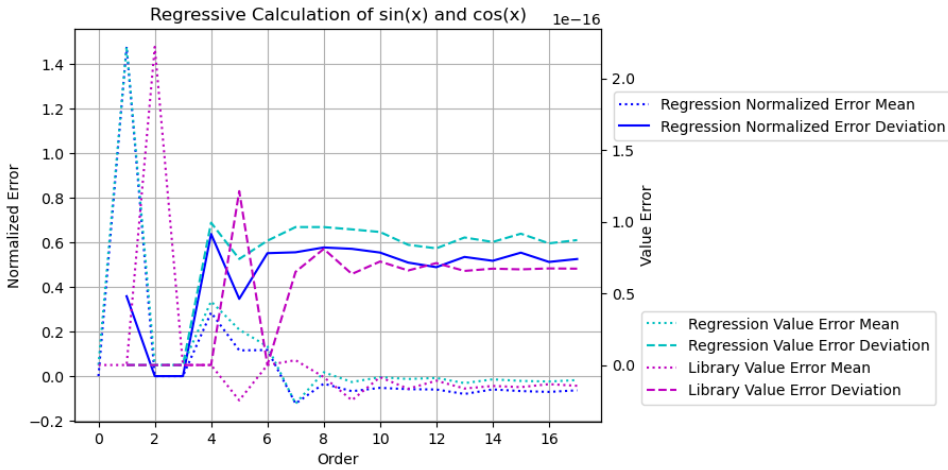

Figure 62: The error deviation and mean for $\sin(x)^2 + \cos(x)^2 - 1, x \in [0, \pi/4]$, for different regression order as shown by the x-axis. In the legend, *Value Error* means the values of $\sin(x)^2 + \cos(x)^2 - 1$, *Normalized Error* means the value errors normalized by the calculated uncertainty, *Regression* means the $\sin(x)$ and $\cos(x)$ generated by regression, and *Library* means library $\sin(x)$ and $\cos(x)$. The y-axis on the left is for normalized error, while the y-axis on the right is for value error.

9 Regressive Generation of Sin and Cos

Starting from Formula (9.1), Formula (9.2) and Formula (9.3) can be used recursively to calculate the sin library function.

$$\sin(0) = \cos(\frac{\pi}{2}) = 0; \qquad \sin(\frac{\pi}{2}) = \cos(0) = 1; \tag{9.1}$$

$$\sin\left(\frac{\alpha+\beta}{2}\right) = \sqrt{\frac{1-\cos(\alpha+\beta)}{2}} = \sqrt{\frac{1-\cos(\alpha)\cos(\beta)+\sin(\alpha)\sin(\beta)}{2}}; \tag{9.2}$$

$$\cos\left(\frac{\alpha+\beta}{2}\right) = \sqrt{\frac{1+\cos(\alpha+\beta)}{2}} = \sqrt{\frac{1+\cos(\alpha)\cos(\beta)-\sin(\alpha)\sin(\beta)}{2}}; \tag{9.3}$$

When $\alpha + \beta \to 0$, Formula (9.2) calculates the square root of a value very close to 0, to result in very coarse precision according to Formula (2.51). Also, both value errors and uncertainties are accumulated in the regression. This regression is not suitable for calculating library sin and cos functions. It can be used to check the trigonometric relation for library sin and cos functions.

The number of regressions 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 value errors of $\sin(x)$ and $\cos(x)$ are checked by $\sin(x)^2+\cos(x)^2-1$. Figure 62 shows that the value errors of the generated $\sin(x)$ and $\cos(x)$ is comparable to those of library $\sin(x)$ and $\cos(x)$. It shows that the result error deviations are close to 0.5.

Because floating-point rounding errors are the only source of errors in the regression, as expected, variance arithmetic provides proper coverage for this regression.

10 Conclusion and Discussion

10.1 Summary of Variance Arithmetic

The starting point of variance arithmetic is the uncorrelated uncertainty assumption, which requires input data to have small enough precision as shown in Figure 2. In addition, it requires that systematic errors are not the major source of uncertainty.

Once the uncorrelated uncertainty assumption is satisfied, variance arithmetic quantifies uncertainty as the deviation of the value errors. It can trace the variable dependency in the intermediate steps using standard statistics. The statistical nature of variance arithmetic is reflected by the bounding leakage $\epsilon = 5.73\ 10^{-7}$ and the required value precision $\tau = 7.18\ 10^{-7}$, which determine the convergence and truncation when Taylor expansion is used to obtain the result mean and uncertainty of an analytic function. In variance arithmetic, ill-formed problems (such as inversion of an input which has a high probability of containing zero) can be invalidated as result divergence, unreliable result variance, or being practically unstable.

The presence of ideal coverage is the necessary condition for a numerical algorithm in variance arithmetic to be correct. The ideal coverage also defines the ideal applicable range for the algorithm. In ideal coverage, the calculated uncertainty equal the error deviation, and the result normalized errors is either normal distributed or delta distributed.

Due to digital limitation of 64-bit conventional floating-point representation, variance arithmetic can only provide proper coverage for floating-point rounding errors, with the empirical error deviations in the range of $[1/5, 5]$

Variance arithmetic has been shown to be widely applicable, so as for analytic calculations, progressive calculation, regressive generalization, polynomial expansion, statistical sampling, and transformations.

10.2 New Numerical Approaches

The conventional numerical approaches focus primarily on the result values, and they need to be reexamined or even reinvented in variance arithmetic. Due to this dependency tracing, for analytic expressions, variance arithmetic allows no numerical execution freedom, and it has no dependency on such freedoms. On the other hand, by using stability truncation, variance arithmetic no longer needs reminder estimator so that it is simpler for truncation problems. Also, variance arithmetic can reject ill-formed problems.

How to treat the result difference using variance arithmetic vs. using conventional floating-point arithmetic is also a general question. The uncertainty bias can be potentially significant near the approximate applicable precision threshold $1/\sigma$, such as $1/(1 \pm (0.2 - 2\ 10^{-5})) = 1.046 \pm 0.250$. Variance arithmetic generally reveals worse results than those of conventional floating-point arithmetic, such as very quick loss of precision in matrix inversion, especially for matrix of large size.

In variance arithmetic, it is generally an order-of-magnitude more complicated to calculate the result uncertainties than the result values. However, modern software programs for analytic calculation can be great help. Also, the calculation of the result uncertainty bias and uncertainty contains large sums of independent terms, so that it is a excellent candidate for parallel processing.

10.3 Recalculating Library Math Functions

The library math functions need to be calculated using variance arithmetic, so that each output value has its corresponding uncertainty. Otherwise, the effect of the existing value errors in the library function can have unpredictable and large effects. For example, this paper shows that the periodic numerical errors in the sine library functions cause resonant-like large result value errors in a FFT reverse transformation.

10.4 First Order Approximation

Most practical calculations are neither pure analytic nor pure numerical, in which the analytic knowledge guides the numerical approaches, such as solving differential equations. In these cases, when input precision is fine enough, first order approximation of variance arithmetic may be a viable solution. This paper provides an example of first order approximation of variance arithmetic, as Formula (5.22) for $\delta^2|\boldsymbol{M}|$. The difference between Formula (2.59) and (2.60) shows that first order approximation of $f(g(x))$ does not have dependency problem. However, Section 4 shows that the result uncertainty needs higher order than first order to stabilize. How to effectively apply variance arithmetic to practical calculations remains an open question.

10.5 Different Floating-Point Representation for Variance

In variance representation $x \pm \delta x$, δx is comparable in value with x, but $(\delta x)^2$ is calculated and stored. This limits the effective range for $x \pm \delta x$ to be much smaller than the full range of the standard 64-bit floating-point representation $\pm\, 2^{52}\, 2^{\pm 1024-52}$ [12], such as when applying Formula (2.22) and (2.23) to floating-point rounding errors in Section 4. Ideally, $(\delta x)^2$ should be calculated and stored in a different 64-bit floating-point representation with no sign bit, more exponent bits, and less significant bits.

10.6 Hardware Implementation

Because variance arithmetic has neither execution freedoms nor the associated dependency problems, it can be implemented in hardware inside CPU or GPU.

10.7 Acknowledgments

As an independent researcher, the author of this paper feels indebted to encouragements and valuable discussions with Dr. Zhong Zhong from Brookhaven National Laboratory, 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. Prof Dongfeng Wu from Louisville University provides valuable guidance and discussions on statistical related topic. The author of this paper is very grateful for the editors and reviewers of *Reliable Computing* for their tremendous help in shaping and accepting the previous version of this paper from unusual source, with managing editor, Prof. Rolph Baker Kearfott in particular.

[12] Including the hidden most significant bit for significand in the IEEE 64-bit floating-point standard.

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] C. P. Wang. A New Uncertainty-Bearing Floating-Point Arithmetic. *Reliable Computing*, 16:308-361, 2012

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

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

[41] Jagdish K. Patel; Campbell B Read Handbook of the normal distribution (2nd ed.). CRC Press. ISBN 0-8247-9342-0.

[42] 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.

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

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

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

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

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

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