

Precision Arithmetic: A New Floating-Point Arithmetic*

Chengpu Wang

40 Grossman Street, Melville, NY 11747, USA

Chengpu@gmail.com

Abstract

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

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

Keywords: computer arithmetic, error analysis, interval arithmetic, multi-precision arithmetic, numerical algorithms.

AMS subject classifications: 65-00

1 Introduction

1.1 Measurement Precision

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

In scientific and engineering measurements the uncertainty of a measurement x is usually characterized by the sample deviation δx [1][2][4]. In certain cases, such as raw reading from an ideal analog-to-digital converter (ADC), the uncertainty of a

*Submitted: May 20, 2010; Revised: November 10, 2010; July 18, 2011; September 1, 2012; Accepted:

measurement x is given as a bounding range Δx^1 [5]. If $[x - \Delta x, x + \Delta x]$ crosses 0, x is neither positive nor negative for certainty due to the following two possibilities:

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

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

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

1.2 Problem of Conventional Floating-Point Arithmetic

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

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

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

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

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

Equation (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

¹ x is normally an integer as the output of an ADC. Ideally, Δx equals a half bit of ADC. Δx can be larger if the settle time is not long enough, or if the ADC is not ideal.

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

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 Equation (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 algorithms³ [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 [5]. The result obtained from an ADC is an integer with fixed uncertainty; thus, a smaller signal value has a coarser precision. When a waveform containing raw digitalized signals from ADC is converted into conventional floating-point representation, the information content of the digitalized waveform is distorted to favour small signals since all converted data now have the same and best possible precision. However, the effects of such distortion in signal processing are generally not clear.

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

1.3 Interval Arithmetic

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

$$[x_1] + [x_2] = [\underline{x}_1 + \underline{x}_2, \bar{x}_1 + \bar{x}_2]; \quad (1.3)$$

$$[x_1] - [x_2] = [\underline{x}_1 - \bar{x}_2, \bar{x}_1 - \underline{x}_2]; \quad (1.4)$$

$$[x_1] \times [x_2] = [\min(\underline{x}_1 \underline{x}_2, \underline{x}_1 \bar{x}_2, \bar{x}_1 \underline{x}_2, \bar{x}_1 \bar{x}_2), \max(\underline{x}_1 \underline{x}_2, \underline{x}_1 \bar{x}_2, \bar{x}_1 \underline{x}_2, \bar{x}_1 \bar{x}_2)]; \quad (1.5)$$

$$[x_1] / [x_2] = [\min(\underline{x}_1 / \underline{x}_2, \underline{x}_1 / \bar{x}_2, \bar{x}_1 / \underline{x}_2, \bar{x}_1 / \bar{x}_2), \max(\underline{x}_1 / \underline{x}_2, \underline{x}_1 / \bar{x}_2, \bar{x}_1 / \underline{x}_2, \bar{x}_1 / \bar{x}_2)]; \quad (1.6)$$

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

³A classic example is the contrast of the uncertainty propagation in the solutions for the 2nd-order linear differential equation v.s. 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).

statistical mean and deviations to characterize uncertainty. Most measured values are well approximated by Gaussian distribution [1][2][4], which has no limited bounding range. Let *bounding leakage* be defined as the possibility of the true value to be outside a bounding range. If a bounding range is defined using a statistical rule on bounding leakage, such as the $6\sigma - 10^{-9}$ rule for Gaussian distribution [4] (which says that the bounding leakage is about 10^{-9} for a bounding range of mean ± 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 foundation⁴.

Another problem is that interval arithmetic only provides the worst case of uncertainty propagation, so that it tends to over-estimate uncertainty in reality. For instance, in addition and subtraction, it gives the result when the two operands are +1 and -1 correlated respectively [24]. However, if the two operands are -1 and +1 correlated respectively instead, the actual bounding range after addition and subtraction reduces, which is called the best case in random interval arithmetic [25]. The vast overestimation of bounding ranges in these two worst cases prompts the development of affine arithmetic [24], 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, Equation (1.7), Equation (1.8) and Equation (1.9) are different expressions of the same $f(x)$; however, the correct result is obtained only through Equation (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 Equation (1.8) and Equation (1.9), respectively. This is called the dependence problem of interval arithmetic [21].

$$f(x) = (x - 1/2)^2 - 1/4; \quad (1.7)$$

$$f(x) = x^2 - x; \quad (1.8)$$

$$f(x) = (x - 1)x; \quad (1.9)$$

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

1.4 Statistical Propagation of Uncertainty

If each operand is regarded as a random variable, and the statistical correlation between the two operands is known, the result ing uncertainty is given by the *statistical*

⁴There 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.

propagation of uncertainty [26][27], with the following arithmetic equations, in which σ is the deviation of a measured value x , P is its precision, and γ is the correlation between the two operands x_1 and x_2 :

$$(x_1 \pm \sigma_1) + (x_2 \pm \sigma_2) = (x_1 + x_2) \quad \pm \sqrt{\sigma_1^2 + \sigma_2^2 + 2\sigma_1\sigma_2\gamma}; \quad (1.10)$$

$$(x_1 \pm \sigma_1) - (x_2 \pm \sigma_2) = (x_1 - x_2) \quad \pm \sqrt{\sigma_1^2 + \sigma_2^2 - 2\sigma_1\sigma_2\gamma}; \quad (1.11)$$

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

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

Tracking uncertainty propagation statistically seems an ideal solution. However, in practice, the correlation between two operands is generally not precisely known, so the direct use of statistical propagation of uncertainty is very limited. In this paper, as a proxy for statistical propagation of uncertainty, an *independence arithmetic* always assumes that no correlation exists between any two operands, whose arithmetic equations are Equation (1.10), Equation (1.11), Equation (1.12) and Equation (1.13), where $\gamma = 0$. Independence arithmetic is actually de facto arithmetic in engineering data processing, such as in the common belief that uncertainty after averaging reduces by the square root of number of measurements [1][2], or the ubiquitous Monte Carlo method⁵ [29][28], or calculating the mean and variance of a Taylor expansion [30].

1.5 Significance Arithmetic

Significance arithmetic [31] tries to track reliable bits in an imprecise value during the calculation. Except for two early attempts [32][33], significance arithmetic has not yet been implemented digitally. In these two attempts, 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, so they both can cause artificial quantum reduction of significance. Significance arithmetic is not widely practiced in scientific and engineering calculations [31].

Stochastic arithmetic [15][34], which can also be categorized as significance arithmetic, randomizes the least significant bits (LSB) of each of input floating-point values, repeats the same calculation multiple times, and then uses statistics to seek invariant digits among the calculation results as significant digits. This approach may require too much calculation since the number of necessary repeats for each input is specific to each algorithm, especially when the algorithm contains branches. Its sampling approach may be more time-consuming and less accurate than direct statistical characterization [4], such as directly calculating the mean and deviation of the underlying distribution. It is based on modeling rounding errors in conventional floating-point arithmetic, which is quite complicated. A better approach may be to define arithmetic rules that make error tracking by probability easier.

As the mathematical foundation to significance arithmetic, when a uncertainty-bearing value is multiplied by a constant, the significance or relative precision still

⁵Most but not all applications of Monte Carlo methods assume independence between any two random variables. In a minority of applications, a Monte Carlo method can be used to construct specified correlation between two random variables [28].

holds, while the absolute precision [1][2] scales with the constant. In this respect, fix-point arithmetic [6], which assumes a fixed absolute precision, does not have a sounding mathematical foundation.

1.6 An Overview of This Paper

In this paper, a new floating-point arithmetic called *precision arithmetic* [35] is developed to track uncertainty during floating-point calculations, as described in Section 2. Generic standards and systematic methods for validating uncertainty-bearing arithmetics are discussed in Section 3. Precision arithmetic is compared with other uncertainty-bearing arithmetics in Section 4 to Section 7. A brief discussion is provided in Section 8.

2 Precision Arithmetic

2.1 Assumptions for Precision Arithmetic

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

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

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

As shown later in this paper, the objectives of precision arithmetic are extended from the central limit theorem [4].

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

2.2 The Independent Uncertainty Assumption

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

Let X , Y , and Z denote three mutually independent random variables [4] with variance $\sigma^2(X)$, $\sigma^2(Y)$ and $\sigma^2(Z)$, respectively. Let α denote a constant. Let $Cov()$

denote the covariance function. Let γ denote the correlation between $(X + Y)$ and $(\alpha X + Z)$. And let:

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

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

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

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

1. The 3rd digit of the measurement is 94% correlated to the signal with $\eta = 1/4$;
2. The 2nd digit of the measurement is 80% correlated to the signal with $\eta = 1/2$;
3. The 1st digit of the measurement is 50% correlated to the signal with $\eta = 1$;
4. The 0th digit of the measurement is 20% correlated to the signal with $\eta = 2$.

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

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

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

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

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

It is tempting to add noise to otherwise unqualified values to make their uncertainties independent of each other. As an extreme case of this approach, if two values are constructed by adding noise to the same signal, they are 50% correlated at the uncertainty level so that they will not satisfy the independent uncertainty assumption⁶.

2.3 Precision Representation and Precision Round Up Rule

Let the content of a floating-point number be denoted as $S@E$, in which S is the significand⁷ and E is the exponent of 2 of the floating-point number. In addition, the precision representation $S\sim@E$ contains a carry \sim to indicate its rounding error, which can be:

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

A *round up* proceeds according to the following round up rule:

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

⁶The 50% curve in Figure 2 thus defines the maximal possible correlations between any two measured signals. This other conclusion of Equation (2.4) makes sense because the measurable correlation between two measurements should be limited by the precisions of their measurements.

⁷While “significand” is the official word [10] to describe “The component of a binary floating-point number that consists of an explicit or implicit leading bit to the left of its implied binary point and a fraction field to the right.”, “mantissa” is often unofficially used instead.

Let the value before any rounding up be the original value, the round-up rule ensures that $S@E$ is always the closest value with exponent E to the original value. After each round up, the original rounding error is reduced by half for the new significand. If the original significand is odd, the round up generates a new rounding error of $1/2$, which is added to the existing rounding error. Since the newly generated rounding error always cancels the existing rounding error, the rounding error range is limited to half bit of the significand, or the bounding range for the rounding error is $[-1/2, +1/2]$. The precision arithmetic also tracks the rounding error bounding range R so that the precision representation becomes $S\sim R@E$.

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

2.4 Precise, Imprecise and Insignificant Values

When R is zero, the precision value $S\sim R@E$ has no uncertainty so it is defined as *precise*; otherwise, it is defined as *imprecise*. After a precise value with odd significand is rounded up once, it becomes an imprecise value of $S?1/2@E$.

An imprecise value can be decomposed as a precise value plus an imprecise zero:

$$S\sim R@E = S@E + 0\sim R@E; \quad (2.5)$$

In Equation (2.5), $S@E$ carries mathematically expected value, while $0\sim R@E$ carries uncertainty. If S is less than R , the imprecise value becomes *insignificant*.

2.5 Probability Distribution of Rounding Errors

An ideal floating-point calculation is carried out conceptually to infinitesimal precision before it is rounded up to representation precision [10][13][15]. Thus, rounding up should be a process independent of any calculation, and it should be evaluated separately. To estimate the rounding error distribution within its bounding range $[-1/2, +1/2]$, a large number⁸ of positive random integers are converted into precision values and then rounded up once at a step time until each of them has a significand smaller than a predefined minimal significand threshold. The precision value at each step is compared with the original value for the rounding error. Figure 3 shows the result histogram of rounding errors for the minimal significand thresholds 0, 1, 4 and 16, respectively. When each significand bit has an equal chance to be either 0 or 1, the result distribution of the rounding errors is expected to be uniformly distributed within the range $[-1/2, +1/2]$ [36]. However, the precision round up rule changes this equal chance for a few lowest digits of a significand. So when the minimal significand threshold is smaller, the bias in rounding error distribution is larger, as shown in Figure 3, and the result distribution is close to uniform only when the minimal significand threshold is 4 and above.

2.6 Result Uncertainty For Addition and Subtraction

In floating-point arithmetic, rounding errors are uncertainties [10][13][15]. The precision round-up rule incorporates all randomness of an imprecise value into its carry and

⁸For each minimal significand threshold, 64K random integers are used. The actual number of random integers is not important as far as 1) it gives a stable empirical histogram, and 2) the random integers are uniformly distributed without repeat in values.

bounding range so that it preserves the independent uncertainty assumption between any two values. The independent uncertainty assumption suggests that the result rounding error distribution of addition is the convolution of the two operand rounding error distributions, while the result rounding error distribution of subtraction is the convolution of the first operand rounding error distribution and the mirror image of the second operand rounding error distribution [4]. Thus, when the exponents of two operands are equal, the results of addition and subtraction are:

$$S_1 \sim_1 R_1 @ E \pm S_2 \sim_2 R_2 @ E = (S_1 \pm S_2) \sim (R_1 + R_2) @ E; \quad (2.6)$$

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

Let $P_{\frac{1}{2}}(x)$ be the rounding error distribution after rounding up, which is uniformly distributed between $[-1/2, +1/2]$ according to Equation (2.7). Let $P_{\frac{n}{2}}(x)$ be the convolution of $P_{\frac{1}{2}}(x)$ according to Equation (2.8):

$$P_{\frac{1}{2}}(x) \equiv 1, \quad -1/2 \leq x \leq +1/2; \quad (2.7)$$

$$P_{\frac{n}{2}}(x) \equiv \int_{-\infty}^{+\infty} P_{\frac{1}{2}}(y) P_{\frac{n-1}{2}}(x-y) dy = \int_{-1/2}^{+1/2} P_{\frac{n-1}{2}}(x-y) dy, \quad n = 2, 3, 4 \dots; \quad (2.8)$$

Equation (2.8) shows that $P_{\frac{n}{2}}(x)$ has a bounding range of $R \equiv \frac{n}{2}$, in which case it is easy to prove that the deviation σ of $P_{\frac{n}{2}}(x)$ is determined by its bounding range R :

$$\sigma^2 = R/6; \quad (2.9)$$

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

$$P_{\frac{m+n}{2}}(x) = \int_{-\infty}^{+\infty} P_{\frac{m}{2}}(y) P_{\frac{n}{2}}(x-y) dy; \quad (2.10)$$

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

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

Figure 4 shows that the rounding error distributions for the same bounding range largely repeat each other, confirming Equation (2.10). Addition and subtraction have a slightly different result distribution due to uneven $P_{\frac{1}{2}}(x)$. In all cases, the distributions quickly approach Gaussian with the increase of bounding ranges.

Even for the worst-case $P_{\frac{1}{2}}(x)$, the deviation σ relates to the bounding range R empirically as $\sigma = 0.423005 R^{0.50000}$ with a reliable factor of 0.9999999, confirming Equation (2.9) empirically.

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

$$D_y(y) = D_x((y - \beta)/\alpha)/\alpha; \quad (2.11)$$

According to the central limit theorem [4], $P_{n/2}(x)$ converges in distribution to a Gaussian distribution of mean 0 and deviation σ with an increased n :

$$P_{n/2}(y) \xrightarrow{d} N(y/\sigma)/\sigma, \quad y \in [-R, +R]; \quad (2.12)$$

In Equation (2.12), $N(x)$ is the density function of a normal distribution. Figure 4 shows that such convergence is very fast.

2.7 Result Uncertainty for Rounding Up

According to Equation (2.11), when an imprecise value is rounded up once, both its original bounding range R and its original deviation σ are reduced to half for the new significand. According to Equation (2.9), there are two ways to carry out rounding up:

- By range: When R is reduced to 1/2-fold, σ is reduced to $1/\sqrt{2}$ -fold.
- By deviation: When σ is reduced to 1/2-fold, R is reduced to 1/4-fold.

Figure 5 compares these two ways of rounding up when the original rounding error range is $R=8$, in which $R=4$ is rounded up by range, while $R=2$ is rounded up by deviation. It clearly shows that rounding up by deviation results in a much similar rounding error distribution. After rounded up, the stable distribution in Equation (2.12) becomes $N(y/\frac{\sigma}{2})/\frac{\sigma}{2}, y \in [-\frac{R}{2}, +\frac{R}{2}]$, which is better approximated as $N(y/\frac{\sigma}{2})/\frac{\sigma}{2}, y \in [-\frac{R}{4}, +\frac{R}{4}]$ than as $N(y/\frac{\sigma}{\sqrt{2}})/\frac{\sigma}{\sqrt{2}}, y \in [-\frac{R}{2}, +\frac{R}{2}]$. Rounding up by deviation is also required by the scaling principle so it is used universally in precision arithmetic. During round up by deviation, the precision of the value in precision representation is preserved.

Rounding up by deviation also introduces bounding leakage called *round-up leakage*. In Figure 5, the 8/2 distribution of the rounding error outside the range $[-2, +2]$ contributes to a round-up leakage of 0.05%. Empirically, Equation (2.13) shows that the round-up leakage φ decreases exponentially with the increased bounding range R . Smaller round-up φ leakage means that the actual rounding error distribution becomes more similar to the rounding error distribution with increased bounding range R .

$$\varphi = 0.31942 \times 0.45775^R; \quad (2.13)$$

2.8 Normalization

When R is above a threshold R_{max} , round-up leakage is small enough so that rounding up by deviation can be applied repeatedly. This is the *normalization* process in precision arithmetic, with the maximal round-up leakage defined as the *normalization leakage* for R_{max} . An imprecise value $S \sim R @ E$ is in normalized format if its R is in the range of $[R_{max}/4, R_{max}]$. Otherwise, it is called a *nearly precise* value. According to Equation (2.13), when $R_{max} = 16$, the maximal normalization leakage is 10^{-6} , which is small enough for most applications. One function of normalization is to enforce the correctness of carry sign \sim in precision representation $S \sim R @ E$. Another reason is to keep the magnitudes of both S and R manageable, e.g., within a CPU word [8].

Because of normalization, $P_{\frac{1}{2}}(x)$ is extended to $P_R(x)$ for 2's fractional $R \in (1/6, R_{max})$ so that the deviation δx and bounding range Δx of $S \sim R @ E$ is:

$$\delta x = \sigma \times 2^E, \quad \sigma < \sqrt{R_{max}/6}; \quad (2.14)$$

$$\Delta x = R \times 2^E, \quad R < R_{max}; \quad (2.15)$$

$$\Delta x / \delta x = R / \sigma = \sqrt{6R}; \quad (2.16)$$

The stable probability density function in Equation (2.12) becomes the probability density function $\rho(\tilde{y})$ in Equation (2.17):

$$\rho(\tilde{y}) = N(\tilde{y}/\delta x)/\delta x, \quad \tilde{y} \in [-\Delta x, +\Delta x]; \quad (2.17)$$

All characteristics of a distribution are decided by its moments [4]. A Gaussian distribution decreases very fast at its two tails so that its tails do not contribute significantly to moment calculation [4][37]. Thus, according to Equation (2.16), when R is sufficiently large, $\rho(\tilde{y})$ has moments identical to those of the corresponding Gaussian distribution, and it is truly the Gaussian distribution but with truncated range at $\pm R$. For example, in Equation (2.13), the bounding leakage actually indicates the difference of 0th moment from 1 with increased R . It is assumed that when $R \geq R_{max}/4$, R is sufficiently large for obtaining at least the 0th, 1st and 2nd momentums of the corresponding unbounded Gaussian distribution. Equation (2.5) can be reinterpreted as Equation (2.18), in which x is the mathematically expected value and y is a ρ -distributed random variable measuring uncertainty of x .

$$x \pm \delta x = x + \tilde{y}, \quad \tilde{y} \in \rho(\tilde{y}); \quad (2.18)$$

2.9 Precision Round Down Rule

As an inverse operation to rounding up by deviation, a precision round-down rule is defined using the scaling principle. After rounding down once, $S \sim R @ E$ becomes $(2S) \sim (4R) @ (E - 1)$. Round down reduces bounding leakage.

To add or subtract two operands with different exponents, the operand with a larger exponent is first rounded down to the other exponent, and the result of addition or subtraction using Equation (2.6) is normalized afterwards.

2.10 Uncertainty Distribution

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

- Because multiplication is implemented as a series of addition and rounding, while division is implemented as a series of subtraction and rounding [8], the stable rounding error distribution after arithmetic operations is Gaussian.
- Because the stable rounding error distribution is *independent* of any initial rounding error distribution of X_i , the rounding error distribution characterized by Equation (2.17) can be extended effectively to describe the uncertainty distribution in general.

- For addition, the mean value for the sum is the sum of the operand mean values, while the variance of the sum is the sum of the operand variances. This relation is extended to all arithmetic operations, to conclude the two objectives of precision arithmetic.

2.11 Uncertainty Initiation

An integer S is initialized as a precise value $S@0$.

A conventional 64-bit floating-point value $S@E$ is usually initialized as a nearly precise value $S?1/2@E$ because the IEEE floating-point standard [10] guarantees accuracy to half bit of a significand.

A mean-deviation pair $(x \pm \delta x)$ of 64-bit conventional floating-point values is initialized as an imprecise value $S\sim R@E$ by:

1. rounding up δx until Equation (2.14) is satisfied;
2. obtaining R and E from final δx ;
3. rounding up x to E ; and
4. obtaining S and \sim from final x .

If the precision of the measured value is worse than 10^{-16} , the result value is normalized. Practically all measured values are normalized.

2.12 Calculation Inside Uncertainty

Table 3 shows the characteristic of different R_{max} . Although a larger R_{max} has smaller normalization leakage, it achieves this by having a larger bounding range, thus it greatly increase the chance for a value to be insignificant. According to Table 3, only when $R_{max} = 16$, the precision arithmetic has a comparable bounding range as the de facto $6\sigma - 10^{-9}$ rule for negligible bounding leakages in statistics. Therefore, $R_{max} = 16$ is used in this paper.

The examples in Table 3 suggest that precision arithmetic does not calculate inside uncertainty, e.g., precision arithmetic represents the expected value of 1.000 ± 0.001 as $2^{10}/2^{10}$; in contrast, all other arithmetic represents the value as $2^{53}/2^{53}$. While calculating many bits inside uncertainty does not seem meaningful according to significance arithmetic [31], not calculating at all inside uncertainty may not be an optimal approach either. Thus, Equation (2.14) is modified as Equation (2.19), in which χ is a small constant positive integer, to introduce the χ -bit calculation inside uncertainty by providing an altered interpretation of the precision for $S\sim R@E$.

$$\delta x = \sqrt{R/6 \cdot 2^{-\chi}} 2^{E+\chi}; \quad (2.19)$$

Table 4 shows examples of precision arithmetic with different χ for $R_{max} = 16$, e.g., with $\chi = 2$, precision arithmetic represents the expected value of 1.000 ± 0.001 as $2^{12}/2^{12}$. χ will be set to 4 empirically later in this paper.

The limited calculation inside uncertainty does not necessarily mean that precision arithmetic has a larger calculation error. The following example shows that the symmetry of precision representation cancels out rounding errors:

$$\begin{aligned} (1/3 \pm 0.001) + (2/3 \pm 0.001) &= (341 + 6.29@ - 10) + (683 - 6.29@ - 10) \\ &= (1024?12.6@ - 10) = 1 \pm 0.001\sqrt{2}; \end{aligned} \quad (2.20)$$

In the above equation, the mathematically expected value for the result is precisely 1, even though the mathematically expected values of the two operands for addition are not precisely 1/3 and 2/3 after the uncertainty initiation, respectively.

2.13 Function Evaluation

The uncertainty of the function $f(x)$ at $(x \pm \delta x)$ is evaluated by the set $\{f(x + \tilde{y}) - f(x)\}$, in which \tilde{y} is a random variable defined by Equation (2.17). Because the bounding goal of precision arithmetic centers on the mathematically expected value, the mean of the set is assumed to be zero, and the variance $(\delta f)^2$ of the set is calculated as:

$$(\delta f)^2 \equiv \int (f(x + \tilde{y}) - f(x))^2 \rho(\tilde{y}) d\tilde{y}; \quad (2.21)$$

Let $M(n)$ be the n th moment of \tilde{y} . If \tilde{y} is symmetrically distributed around $\tilde{y} = 0$, then all the odd moments of \tilde{y} are zero, as described by Equation (2.22), in which j is a natural number. If y is normal-distributed, then its non-zero moments are given by Equation (2.23).

$$M(2j + 1) = 0; \quad (2.22)$$

$$M(2j + 2) = \prod_{k=0}^j (2k + 1); \quad (2.23)$$

If the function $f(x)$ is Taylor expandable at x , $f(x + \tilde{y}) - f(x)$ is calculated according to Equation (2.25), in which $f^{(n)}$ is the n th derivatives of $f(x)$ at x . $(\delta f)^2$ is given by Equation (2.25).

$$f(x + \tilde{y}) - f(x) = \sum_{n=1}^{\infty} \frac{f^{(n)}(x)}{n!} \tilde{y}^n; \quad (2.24)$$

$$(\delta f)^2 = \sum_{n=0}^{\infty} \sum_{j=0}^{\infty} \frac{f^{(n)}(x)}{n!} \frac{f^{(j)}(x)}{j!} (\delta x)^{n+j} M(n+j) - f(x)^2; \quad (2.25)$$

Let $P(f(x)) \equiv \delta f(x)/|f(x)|$ be defined as the precision for $f(x)$; and let α be a constant. According to Equation (2.25):

$$P(\alpha x) = P(x); \quad (2.26)$$

$$|x| \gg \delta x : P(1/x) \approx P(x); \quad (2.27)$$

$$P(x^2) = \sqrt{4P(x)^2 + 3P(x)^4}; \quad (2.28)$$

$$|x| \gg \delta x : P(\sqrt{x}) \approx P(x)/2; \quad (2.29)$$

Equation (2.26) and Equation (2.27) confirm the scaling principle.

The Taylor expansion can also be used to find the result $(\delta f)^2$ of the function $f(x_1, x_2)$, in which $f^{(m,n)}$ is the m th and n th partial derivatives of x_1 and x_2 , respectively; and the independent uncertainty assumption between x_1 and x_2 leads to independence between the random variables \tilde{y}_1 and \tilde{y}_2 in Equation (2.31):

$$f(x_1 + \tilde{y}_1, x_2 + \tilde{y}_2) - f(x_1, x_2) = \sum_{m=0}^{\infty} \sum_{n=0}^{\infty} \frac{f^{(m,n)}}{m!n!} \tilde{y}_1^m \tilde{y}_2^n - f(x_1, x_2); \quad (2.30)$$

$$(\delta f)^2 = \sum_{m=0}^{\infty} \sum_{n=0}^{\infty} \sum_{i=0}^{\infty} \sum_{j=0}^{\infty} \frac{f^{(m,n)}}{m!n!} \frac{f^{(i,j)}}{i!j!} (\delta x_1)^{m+i} (\delta x_2)^{n+j} M(m+i) M(n+j) - f(x_1, x_2)^2; \quad (2.31)$$

Such an approach can be extended to a function of an arbitrary number of input variables. Equation (2.31) shows that an input contributes to the result uncertainty in more than one way, in the same way as Δx appears in more than one term in the Taylor expansion of $\Delta f(x, y, z)$.

According to Equation (2.31):

$$\delta(x_1 \pm x_2)^2 = (\delta x_1)^2 + (\delta x_2)^2; \quad (2.32)$$

$$P(x_1 x_2) = \sqrt{P(x_1)^2 + P(x_2)^2 + P(x_1)^2 P(x_2)^2 / 6}; \quad (2.33)$$

$$|x| \gg \delta x : P(x_1/x_2) \approx \sqrt{P(x_1)^2 + P(x_2)^2 + P(x_1)^2 P(x_2)^2 / 6}; \quad (2.34)$$

Equation (2.32) confirms Equation (2.6). Due to Equation (2.27), Equation (2.33) is identical to Equation (2.34), both of which can be concluded independently from the scaling principle.

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

2.14 Dependence Problem

Equation (2.25) and its multi-dimension extensions such as Equation (2.31) accurately account for all contribution to the result uncertainty within $f(x)$, providing a clean and deterministic solution for $(\delta f)^2$, e.g., it gives the same result for Equation (1.7), Equation (1.8) and Equation (1.9). Therefore, precision arithmetic has no expression-based dependence problem.

It is tempting to define basic arithmetic operations as Equation (2.32), Equation (2.33) and Equation (2.34), and apply them progressively to calculate $f(x)$, similar to how basic arithmetic operations are used in conventional floating arithmetic. However, such an approach may apply the independent uncertainty assumption wrongly between a value and its mathematical expression, such as between x and x^2 , so that it may result in the dependence problem similar to that of interval arithmetic [21]. For example, for such a use of precision arithmetic, only Equation (1.7) gives the correct result $4(x - \frac{1}{2})^2(\delta x)^2 + 3(\delta x)^4$, while Equation (1.8) over-estimates the result uncertainty by $4x(\delta x)^2$, which has the largest fold of over-estimation at $x = \frac{1}{2}$ when $\delta x < 1$. Let x , y and z be three values satisfying the independent uncertainty assumption. Function $f(x, y)$ and $g(x, z)$ are correlated through x and they need to be tested for the independent uncertainty assumption before they can be used to calculate $h(f, g)$ using precision arithmetic. For example, using precision arithmetic, the correlation γ between $(x \pm \delta x)$ and $(x \pm \delta x)^2$ is calculated by Equation (2.35), which shows that γ increases with decreased precision $P \equiv \delta x/|x|$ of x , in contrast with how the independent uncertainty assumption favors finer precision P in Equation (2.4). After applying Equation (2.4), the correlation on the uncertainty level γ_P is no less than $\frac{16}{19}$,

so that precision arithmetic rejects calculating Equation (1.8) progressively using the basic arithmetic operations.

$$\gamma = \frac{\int ((x + \tilde{y})^2 - x^2)((x + \tilde{y}) - x)\rho(\tilde{y})d\tilde{y}}{\sqrt{\int ((x + \tilde{y})^2 - x^2)^2\rho(\tilde{y})d\tilde{y}}\sqrt{\int ((x + \tilde{y}) - x)^2\rho(\tilde{y})d\tilde{y}}} = \frac{1}{\sqrt{1 + \frac{3}{4}P^2}} \quad (2.35)$$

In other words, converting a numerical algorithm from using conventional floating-point arithmetic to using precision arithmetic may be more complicated than directly replacing the variable types and the arithmetic being used. To avoid the dependence problem, the safest approach is to obtain an analytic form of the final expression of an algorithm before applying Equation (2.25) and its multi-dimension extensions, similar to how symbolic calculations are currently used in affine arithmetic [38].

2.15 Conditional Execution

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

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

2.16 Implementation

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

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

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

- 11 bits for storing exponent E ;
- 53 bits for storing significand S (without using the hidden MSB);
- 1 bit for storing sign;

- 2 bits for storing carry \sim ;
- 13 bits to store the bounding range R as a fixed-point value.

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

2.17 Alternative Form of Precision Arithmetic

Because the independent uncertainty assumption can lead directly to 1) the Gaussian distribution as the underlying distribution for rounding errors, and 2) Equation (2.25) and its multi-dimension extensions such as Equation (2.31) for generic Taylor expansion, an *alternative form of precision arithmetic* is to represent each uncertainty-bearing value as $x \pm \delta x$ in Equation (2.18). The bounding range is then calculated from δx as the confidence interval [4] for any required upper limit on bounding leakage, e.g., if the required bounding leakage is 10^{-9} or less, the bounding interval is $[x - 6\delta x, x + 6\delta x]$. This alternative form of precision arithmetic is *not* adopted in this paper for the following reasons:

- For the actual numerical calculation, if conventional floating-point arithmetic is used separately for x and δx , then x and δx will be contaminated by unspecified amount of rounding errors. Thus, the current form of precision arithmetic defines its own floating-point representation for $x \pm \delta x$ as $S \sim R @ E$.
- Another effect of using conventional floating-point arithmetic for x and δx is to calculate many bits inside uncertainty, whose validity is not clear at this moment. In contrast, as demonstrated by Table 3 and Table 4, the current form of precision arithmetic controls the number of bits calculated inside uncertainty.

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

2.18 Types of Uncertainties Included in Precision Arithmetic

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

- input uncertainties
- rounding errors
- truncation errors
- modeling errors

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

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

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

3 Standards and Methods for Validating Uncertainty-Bearing Arithmetic

3.1 Validating Standards and Methods

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

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

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

3.2 Comparing Uncertainty-Bearing Arithmetics

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

As stated previous:

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

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

- According to Equation (2.6) and Equation (2.9), the result deviation of addition and subtraction by precision arithmetic propagates in the same way as that of independence arithmetic, while the result bounding range propagates in the same way as that of interval arithmetic. Hence addition and subtraction can not differentiate the three arithmetics.
- According to Equation (2.33) and Equation (2.34), the result precision of multiplication and division by precision arithmetic is always larger than that by independence arithmetic. However, if both operands have precisions much less than 1, the result precision of multiplication and division is very close to that of independence arithmetic. Thus, the result of precision arithmetic should be much closer to that of independence arithmetic.
- The uncertainty distribution of precision arithmetic is a truncated Gaussian distribution according to Equation (2.17). When an imprecise value is multiplied by a constant, because its uncertainty bounding range and its uncertainty distribution deviation can not be scaled linearly simultaneously according to Equation (2.14) and Equation (2.15), precision arithmetic chooses to preserve the the distribution deviation rather than the bounding range, thus introducing bounding leakages. Figure 5 suggests that the bounding range of precision arithmetic should be much narrower than that of interval arithmetic, while Equation (2.13) shows that such introduced bounding leakage should be small, e.g., less than 10^{-6} for the chosen normalization method.
- Equation (2.25) and its multi-dimensional expansions such as Equation (2.31) are mathematically strict so that precision arithmetic has no dependence problem on expression differences. In contrast, there seems no similar solution for generic Taylor expansion using interval arithmetic, because there seems no general analytic solution to find maxima and minima for generic polynomial at any range [12]. In this respect, precision arithmetic is mathematically simpler than interval arithmetic.

3.3 Comparing Algorithms for Tests

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

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

- *Transforming*: A transforming algorithm has about equal amounts of input and output data. The information contained in the data remains about the same after transforming. The Discrete Fourier Transformation is a typical transforming algorithm, which contains exactly the same amount of input and output data, and its output data can be transformed back to the input data using essentially the same algorithm. Matrix inversion is another such reversible algorithm. For reversible transformations, a unique requirement for uncertainty-bearing arithmetic is to introduce the least amount of additional uncertainty after forward and reverse transformation, which provides an objective testing standard for a uncertainty-bearing arithmetic. A test of uncertainty-bearing arithmetic using FFT algorithms is provided in Section 4, and a test of matrix inversion is provided in Section 5.
- *Generating*: A generating algorithm has much more output data than input data. Solving differential equations numerically and generating a numerical table of a specific function are two typical generating algorithms. The generating algorithm codes mathematical knowledge into data, so there is an increase of information in the output data. In generating algorithms, input uncertainty should also be considered when deciding if the result is good enough so that the calculation can stop. Some generating algorithms are theoretical calculations which involve no imprecise input so that all result uncertainty is due to rounding errors. Section 6 demonstrates such an algorithm, which calculates a table of sine function using trigonometric relations and two precise input data, $\sin(0) = 0$ and $\sin(\pi/2) = 1$.
- *Reducing*: A reducing algorithm has much less output data than input data such as numerical integration and statistical characterization of a data set. Some information of the data is lost while other information is extracted during reducing. Conventional wisdom is that a reducing algorithm generally benefits from a larger input data set [4]. Such notion needs to be re-evaluated when uncertainty accumulates during calculation. A test of uncertainty-bearing arithmetic using numerical integration is provided in Section 7.

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

- In an *expressive* algorithm, each output is implemented as an analytic mathematical expression of inputs. Equation (2.25) and Equation (2.31) of precision arithmetic are powerful tools to solve expressive algorithms using precision arithmetic.
- In a *progressive* algorithm, each output is based on partial inputs and previously generated outputs. If an output depends on the state which is defined by previous inputs and outputs, the algorithm is also progressive. Most practical algorithms are progressive. Even if there may be an expected analytic mathematical expression between its input and output, an algorithm may not be expressive due to its progressive implementation. The dependence problem usually exists in a progressive algorithm.

3.4 Input Data to Use

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

- A *clean* signal is obtained by directly casting a perfect signal to a specific precision. Such casting may contain systematic rounding errors. For instance, if a perfect sine signal repeats 2^n times in 2^{n+2} samples, the signal contains only values 0, ± 1 and $\pm 1/\sqrt{2}$, with each value repeated multiple times in the signal. The symmetry of the arithmetic may be tested by the output symmetry of clean input signals, e.g., in Discrete Fourier Transformation, the frequency space should be conjugately symmetrical [12] for a clean signal in signal space.
- A *noisy* signal is obtained by adding Gaussian noise of the same deviation as the input deviation to a perfect signal before casting. It represents a realistic signal, and it should be used in validating arithmetic on uncertainty propagation.

4 Validation Using FFT

4.1 Frequency Response of DFT (Discrete Fourier Transformation)

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

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

$$H[n] = \sum_{k=0}^{N-1} h[k] e^{i2\pi \frac{k}{N}n}; \quad h[k] = \frac{1}{N} \sum_{n=0}^{N-1} H[n] e^{-i2\pi \frac{n}{N}k}; \quad (4.1)$$

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

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

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

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

$$H[n] = N/\pi; \quad (4.5)$$

The DFT $H[n]$ of the signal $h[k]$ is the digital implementation of the continuous Fourier transformation $H(s)$ of the signal $h(t)$ [12], in which $H(s) = i\delta(s - f)$ for

$h[k] = \sin(2\pi f)$. From Equation (4.4), when the signal frequency of the original signal falls between two index frequencies of the transformation, the peak is lower and wider with a wrong phase, depending on the fractional frequency $|f - F|$. Thus, the DFT is only faithful for signal components with exactly one of the index frequencies of the transform, and it suppresses and widens unfaithful signal components, each of which has a phase different from its closest faithful representation, with the phase of a sine wave distorted toward that of a cosine wave, and vice versa. Examples of unfaithful representation of fractional frequency by the DFT are shown in Figure 6.

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

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

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

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

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

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

$$h[N] = h[0]; \quad (4.8)$$

Thus, the DFT has an implied assumption that the signal $h[k]$ repeats itself outside the region of $[0, N - 1]$ [39]. For an unfaithful frequency, $h[N - 1]$ and $h[N]$ are discontinuous in regard to signal periodicity, resulting in larger peak width, lower peak height, and the wrong phase.

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

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

4.2 FFT (Fast Fourier Transformation)

When $N = 2^L$, in which L is a positive integer, the generalized Danielson-Lanczos lemma [12] can be applied to the DFT as FFT [12], in which $m = L, L - 1, \dots, 1, 0$

indicates progress of the transformation, and j is the bit-reverse of n :

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

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

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

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

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

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

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

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

FFT is one of the most widely used algorithms [12]. By providing a balanced usage of addition, subtraction and multiplication involving trigonometric functions, it services as one of the most important benchmarks in testing processors for overall mathematical performance [40]. Since all three uncertainty-bearing arithmetics are generic in nature without special optimization for FFT algorithms, the testing result using FFT algorithms should be generic for expressive algorithms. FFT algorithms provide a good linear platform to test any uncertainty-bearing arithmetic, with 1) a clearly defined value measuring the amount of calculation, 2) a known error propagation mechanism, 3) no conditional execution in the algorithm, and 4) using only basic arithmetic operations without the dependence problem.

4.3 Evaluating Calculation Inside Uncertainty

Figure 7 shows the output deviations and value errors for a noisy sine signal after forward FFT. It shows that the output deviations using precision arithmetic are slightly larger than the output deviations using independence arithmetic, but much less than those using interval arithmetic. For a fixed input deviation, the output deviation using independence arithmetic is a constant for each FFT. Because the value and uncertainty interact with each other through normalization in precision arithmetic, output deviations of Equation (4.11) are no longer a constant. One interesting consequence is that only in precision arithmetic the output deviations for a noisy input signal are larger than those for a corresponding clean input signal.

Figure 7 shows that the value errors calculated using precision arithmetic are comparable to those using conventional floating-point arithmetic, and they are both comparable to the output deviations using either precision arithmetic or independence arithmetic. In other words, the result of calculating 2-bit or 53-bit into uncertainty are quite comparable so that the limited calculation inside uncertainty is reasonable.

Figure 8 compares the output value errors of precision arithmetic calculating different bits inside uncertainty. With no calculation inside uncertainty, the output value errors exist only on four levels. Such quantum distribution is reduced noticeably by the 2-bit calculation inside uncertainty, and is further reduced by the 4-bit calculation inside uncertainty. Compared with Figure 7, Figure 8 shows that the result using precision arithmetic with the 4-bit calculation inside uncertainty approaches that using independence arithmetic so that the 4-bit calculation inside uncertainty seems sufficient. Precision arithmetic with the 4-bit calculation inside uncertainty is used for further tests.

4.4 Evaluating Uncertainty Distribution

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

Because value errors are only observable as errors of the result significand, while precision arithmetic has limited bits calculated inside uncertainty, the theoretical rounding error distributions obtained from Equation 2-16 are integrated around LSB of the significand, to result in theoretical probability densities of the value error significand for different bounding ranges. The occurrence frequencies of the output value error significand are counted for different bounding ranges and normalized as measured probability densities. Figure 9 and Figure 10 show the theoretical and empirical distribution of significand errors for different bounding ranges on noisy sine signals for the forward and reverse FFT, respectively. They show that the uncertainty distribution of the forward FFT is very close to the bounding distribution, while the uncertainty distribution of the reverse FFT is indeed narrower. In contrast, for the linear input signal, the forward and reverse FFT have almost identical value error significands distribution. Thus, the previous hypothesis on the distribution width of value error significands is validated empirically in this case.

Another way to measure the empirical probability distribution is to normalize the output value error with the corresponding output uncertainty deviation according to Equation 2-11. If output value errors are Gaussian-distributed with the deviation given precisely by the corresponding output uncertainty deviation, then the normalized histogram should be normal-distributed. Figure 11 shows that the normalized histograms using independence arithmetic are best fit by a Gaussian distribution with the deviation of 0.98 and the mean of 0.06. If there were no rounding errors, all the results should have zero value errors. Indeed, in Figure 11, the measured histograms have a much larger population at where the value errors are zero so that the measured deviation is expected to be less than 1. The reason for the small but non-zero mean for the measured probability distribution is not clear at this moment. Figure 12 shows that the measured normalized histograms using precision arithmetic are very similar

to those using independence arithmetic, and they are all well fitted by a Gaussian distribution which is very close to a normal distribution.

4.5 Evaluating Uncertainty-Tracking

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

$$\delta y = \alpha \beta^L \delta x; \quad (4.14)$$

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

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

$$z = \alpha \beta^L; \quad (4.15)$$

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

The estimated average output error significant can then be compared with the measured ones to evaluate the predictability of the uncertainty-tracking mechanism. One example of measured average output error significands is shown in Figure 16, which shows that the average output error significands using precision arithmetic are a constant despite that both average output uncertainty deviations and value errors increase linearly with the input deviation and exponentially with the FFT order. Equation (4.14) and Equation (4.15) are found empirically to be a good fit for any FFT algorithm with any input signal using any arithmetic.

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

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

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

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

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

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

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

- Figure 17 confirms that independence arithmetic is ideal for uncertainty-tracking for FFT algorithms: 1) β for error significands is a constant 1; and 2) β for both the average output deviations and value errors is both 1 for the round-trip FFT because the result signal after the round-trip FFT should be restored as the original signal. Thus, the theoretical β for the forward FFT and the reverse FFT are $\sqrt{2}$ and $1/\sqrt{2}$, respectively.
- Precision arithmetic has β for average output deviations slightly larger than those of value errors, resulting in β for average output error significands to be a constant slightly less than 1. Its β for average output deviations is slightly larger than the corresponding β of independence arithmetic, so its average output deviations propagate slightly faster with an increased FFT order than those of independent arithmetic. Such slightly faster increase with the amount calculation is anticipated by the difference between Equation (2.33) and Equation (1.12) with $\gamma = 0$.
- The β for average output deviations using interval arithmetic is always much larger than β for average output value errors, resulting in β for average output error significands of about 0.62 for the forward and reverse FFT, and about $0.39 \cong 0.62^2$ for the roundtrip FFT. Consequently, using interval arithmetic, the average output deviations propagate much faster with the amount of calculations than the value error does. Such fast propagation of uncertainty ranges is intrinsic to interval arithmetic due to its worst-case assumption.

Figure 20 shows that for the forward FFT, the measured average output error significands using either precision arithmetic or independence arithmetic are approximately constant of 0.8 in both cases, regardless of the FFT order. In contrast, Figure 20 shows that using interval arithmetic the measured average output error significands decrease exponentially with the FFT order L . Such trends of average error significands

hold for all three FFT algorithms and all input signals. Thus, in this case, the direct uncertainty tracking provided by precision arithmetic is better than the indirect uncertainty tracking provided by interval arithmetic.

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

4.6 Evaluating Uncertainty-Bounding

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

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

5 Validation using Matrix Inversion

5.1 Uncertainty Propagation in Matrix Determinant

Let vector $[p_1, p_2 \dots p_n]_n$ denote a permutation of the vector $(1, 2 \dots n)$ [41]. Let $\$[p_1, p_2 \dots p_n]_n$ denote the permutation sign of $[p_1, p_2 \dots p_n]_n$ [41]. For a n -by- n square matrix M with the element $x_{i,j}$, $i, j = 1, 2 \dots n$, let its determinant be defined as

Equation (5.1) [12] and let the sub-determinant at index (i, j) be defined as Equation (5.2) [41]:

$$|M| \equiv \sum_{[p_1 \dots p_n]_n} \$[p_1 \dots p_n]_n \prod_k x_{k, p_k}; \quad (5.1)$$

$$|M|_{i,j} \equiv \sum_{\substack{p_i=j \\ [p_1 \dots p_n]_n}} \$[p_1 \dots p_n]_n \prod_{\substack{k \neq i \\ k}} x_{k, p_k}; \quad (5.2)$$

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

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

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

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

$$|M| = \sum_{j_1} x_{i_1, j_1} |M_{i_1, j_1}| = \sum_{j_1} \sum_{\substack{i_2 \neq i_1, j_2 \neq j_1 \\ j_2}} x_{i_1, j_1} x_{i_2, j_2} |M_{< i_1, i_2 >_n [j_1, j_2]_n}|; \quad (5.5)$$

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

$$||M_{< i_1, i_2 >_n [j_1, j_2]_n}|| = ||M_{< i_1, i_2 >_n [j_2, j_1]_n}||; \quad (5.6)$$

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

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

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

According to the basic assumption of precision arithmetic, the uncertainty of each element $x_{i,j}$ is independently and symmetrically distributed. Let $\tilde{y}_{i,j}$ denote a random variable at the index (i, j) symmetrically distributed with the deviation $\delta x_{i,j}$. Let $|\widetilde{M}|$ denote the determinant of the matrix \widetilde{M} whose element is $(x_{i,j} + \tilde{y}_{i,j})$. Applying Taylor

expansion to Equation (5.8) results in Equation (5.9), which results in Equation (5.10) after applying Equation (2.21) and Equation (2.22):

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

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

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

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

The element $z_{i,j}$ at the index (i, j) of the inverted matrix M^{-1} is calculated as [41]:

$$z_{i,j} = \frac{|M_{j,i}|}{|M|}; \quad (5.12)$$

Equation (5.12) shows that the uncertainty of the matrix determinant $|M|$ propagates to every element of the inverted matrix M^{-1} . Instead, the matrix which consists of the element $|M_{j,i}|$ at the index (i, j) is defined as the adjugate matrix M^A [41], whose elements are not directly affected by M^{-1} . M^A is recommended to replace M^{-1} whenever the application allows [12].

5.2 Matrix Testing Algorithm

A matrix \widehat{M} is constructed using random integers between $[-16384, +16384]$. Its adjugate matrix \widehat{M}^A and its determinant $|\widehat{M}|$ are calculated precisely using integer arithmetic. \widehat{M} , $|\widehat{M}|$ and \widehat{M}^A are all scaled proportionally as M , $|M|$ and M^A so that the elements of M are 2's fractional numbers randomly distributed between $[-1, +1]$. The scaled matrix M is called a clean testing matrix. M^{-1} is calculated from $|M|$ and M^A and it is deemed precise⁹. Floating-point arithmetic is used to calculate M^A and M^{-1} from M , and the results are compared with the corresponding precise results for value errors. Gaussian noises corresponding to different deviations between 10^{-17} and 10^{-1} may be added to each clean testing matrix, to result in noisy testing matrix. Each combination of matrix size and input deviation is tested by 32 different noisy matrices.

5.3 Testing Matrix Stability

Each matrix has a different stability [42], which means how stable the inverted matrix is in regard to small value changes of the original matrix elements. It is well known that

⁹Because $|M_{j,i}|$ and $|M|$ are not independent of each other, M^{-1} calculated by Equation (5.12) contains the dependency problem. Only introduced in the last step, this dependency problem does not affect M^{-1} severely, but it affects $(M^{-1})^{-1}$ noticeably. For example, the result error distribution of $(M^{-1})^{-1}$ is no longer Gaussian according to Figure 30, and the result average error significands of $(M^{-1})^{-1}$ increase slowly with the matrix size.

more mutual cancellations in Equation (5.1) mean less stability of the matrix [11][12], with the Hilbert matrix [43] being the most famous unstable matrix. The condition number has been defined to quantify the stability of a matrix [42]. Even though the definition of the condition number excludes the effects of rounding errors, in reality most calculations are done numerically using conventional floating-point arithmetic so that the combination effect of rounding errors and matrix instability cannot be avoided in practice. When a matrix is unstable, the result is more error prone due to rounding errors of conventional floating-point arithmetic [11]. Unfortunately, conventional floating-point arithmetic has neither control nor characterization on rounding errors. Consequently, there are no general means to avoid the mysterious and nasty “numerical instability” in numerical applications due to rounding errors [11]. For example, the numerical value of the calculated condition number of a matrix may have already been a victim of “numerical instability” and there is no sure way to judge this suspicion, so this value may not be very useful in judging the stability of the matrix in practice. On the other hand, the rounding errors of conventional floating-point arithmetic can be used to test the stability of a matrix. Rounding errors effectively change the item values of a matrix, so they produce a larger effect on a less stable matrix. If the inverted matrix and the adjugate matrix are calculated using conventional floating-point arithmetic, larger value errors indicate that the matrix is less stable.

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

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

While each condition number [42] only gives the result sensitivity to one matrix item, Equation (5.10) contains the the result sensitivity to any matrix item, any combination of matrix items, as well as the aggregated result uncertainty deviation. Therefore, Equation (5.10) and Equation (5.11) may be better than condition number for describing matrix stability.

5.4 Testing Uncertainty Propagation in Adjugate Matrix

When the adjugate matrix is calculated using precision arithmetic, Figure 28 shows that the average output deviations for the adjugate matrix increase linearly with the input deviation, which is in good agreement with Equation (4.14). Such relation is also true for maximal and average output values errors. Equation (4.14) is expected to describe the general value error propagation for linear algorithms in which L is the amount of calculations [14]. The question is what value L should be when calculating the adjugate matrix of a square matrix of size N . Figure 28 suggests that L increases with N^2 for the average output precision and average output error, while similar to Figure 15, L increases with N for the maximal output deviation and maximal output error. The discrepancy between Figure 28 and Figure 15 may be due to the fact that the average calculation involves N^2 array items, while the maximal calculation involves only N array items¹⁰.

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

6 Validation Using Regressive Calculation of Sine Values

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

$$\sin(0) = \cos\left(\frac{\pi}{2}\right) = 0; \quad \sin\left(\frac{\pi}{2}\right) = \cos(0) = 1; \quad (6.1)$$

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

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

This algorithm is very different from both FFT and matrix inversion in nature because Equation (6.2) and Equation (6.3) are no longer linear, and the test presents a pure theoretical calculation without input uncertainty. The regression iteration count L is a good measurement for the amount of calculations. Each repeated use of

¹⁰The amount of calculation L does not mean the calculation complexity using the Big O notation [44]. It is just a measurement of how output uncertainty increases with a dimension of calculation according to (4.14) [14]. For example, any sorting algorithm will not change the uncertainty distribution so that L is always 0 regardless the calculation complexity for the sorting algorithm. The actual calculation time suggests calculation complexity of $O(2^N)$ for using Equation (5.11) to calculate matrix determinant.

Equation (6.2) and Equation (6.3) accumulates calculation errors to the next usage so that both value errors and uncertainty are expected to increase with L . Each regression iteration L corresponds to 2^{L-2} outputs, which enables statistical analysis for large L .

Figure 31 shows that both average output value errors and the corresponding average output deviation increase exponentially with the regression count for all three arithmetics, and Figure 32 shows that in response to the increased amount of calculations:

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

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

7 Validation Using Numerical Integration

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

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

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

$$x_{mid} \equiv (x_{start} + x_{end})/2; \quad (7.1)$$

$$f_{err} \equiv (f(x_{start}) + f(x_{end}))/2 - f(x_{mid}); \quad (7.2)$$

$$f_{\Delta} \equiv f(x_{mid})(x_{end} - x_{start}); \quad (7.3)$$

If f_{err} becomes insignificant, the interval $[x_{start}, x_{end}]$ is considered to be fine enough, and f_{Δ} is added to the total integration. Otherwise, the search continues on the

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

Equation (7.4) provides an example test for the above simple algorithm, in which n is a positive integer.

$$\frac{4^{n+1} - 10^{-6(n+1)}}{n+1} = \int_{10^{-6}}^4 x^n dx; \quad (7.4)$$

Table 5 shows that the result of numerical integration is very comparable to the expected value. It shows that the above integration algorithm introduces no broadening of result uncertainty, so the above algorithm always selects optimal integration intervals when calculating the best possible result for a numerical integration. Tests of integration using different polynomials with different integration ranges all confirm the above result.

One thing worth noticing in Table 5 is that even though Equation (7.3) consistently underestimates integration for each integration interval $[x_{start}, x_{end}]$, the final underestimation is quite small and comparable to the uncertainty deviation. This example shows that the bias inside the uncertainty range has insignificant contribution to the final result using precision arithmetic.

8 Conclusion and Discussion

8.1 Summary

The starting point of precision arithmetic is the independent uncertainty assumption, which requires that its low-significance input data not be overwhelmed by systematic errors, and all of its input data not have messed-up identities. Figure 2 quantifies the statistical requirements for input data to precision arithmetic.

Due to the independent uncertainty assumption and central limit theorem, the rounding errors of precision arithmetic are shown to be bounded by Gaussian distribution with a truncated range. The rounding error distribution is extended to describe the uncertainty distribution in general, with the uncertainty deviation of a single precision value given by Equation (2.17), and the result uncertainty deviation of a function given by Equation (2.25) and its multi-dimension extensions such as Equation (2.31).

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

Precision arithmetic is quite different from conventional arithmetic. Due to the dependence problem when precision arithmetic is applied incorrectly, precision arithmetic has much less operational freedom than conventional arithmetic and may require extensive symbolic calculations. Also, the comparison relation in conventional arithmetic needs to be re-evaluated in precision arithmetic.

8.2 Improving Precision Arithmetic

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

There are actually three different ways to round up $(2S+1)R@E$:

1. always round up $(2S+1)R@E$ to $(S+1) - R@(E+1)$;
2. always round up $(2S+1)R@E$ to $S + R@(E+1)$;
3. randomly round up $(2S+1)R@E$ to either $(S+1) - R@(E+1)$ or $S + R@(E+1)$.

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

The objectives of precision arithmetic needs to be studied further. For example, Equation (2.21) has rejected the effect of uncertainty on the expected value by incorporating the value shift due to uncertainty as increase of variance, such as in the case of calculating $f(x) = x^2$. The effect of such asymmetrical broadening is unclear at this moment.

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

Some empirical evidences suggest that when an algorithm contains dependence problem, its result uncertainty distribution deviates from Gaussian, such as the uncertainty distribution of $(M^{-1})^{-1}$ in Figure 30. The generality of this connection needs to be studied further.

The convergence property of Equation (2.25) and Equation (2.31) needs to be studied further theoretically, such as their convergence range in terms of the input variable x when calculating $f(x) = 1/x$ or $f(x) = \sqrt{x}$.

The measured nearly constant values of average error significands of precision arithmetic for each particular algorithm need to be explained theoretically. Without such an explanation, precision arithmetic is mostly a pure numerical approach.

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

8.3 Acknowledgements

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

References

- [1] Sylvain Ehrenfeld and Sebastian B. Littauer. *Introduction to Statistical Methods*. McGraw-Hill, 1965.
- [2] John R. Taylor. *Introduction to Error Analysis: The Study of Output precisions in Physical Measurements*. University Science Books, 1997.
- [3] Jurgen Bortfeldt, editor. *Fundamental Constants in Physics and Chemistry*. Springer, 1992.
- [4] Michael J. Evans and Jeffrey S. Rosenthal. *Probability and Statistics: The Science of Uncertainty*. W. H. Freeman, 2003.
- [5] Paul Horowitz and Hill Winfield. *Art of Electronics*. Cambridge Univ Press, 1995.
- [6] Fixed-point arithmetic. http://en.wikipedia.org/wiki/Fixed-point_arithmetic, 2011. wikipedia, the free encyclopedia.
- [7] Arbitrary-precision arithmetic. http://en.wikipedia.org/wiki/Arbitrary-precision_arithmetic, 2011. wikipedia, the free encyclopedia.
- [8] John P Hayes. *Computer Architecture*. McGraw-Hill, 1988.
- [9] David Goldberg. What every computer scientist should know about floating-point arithmetic. *Computing Surveys*, March 1991.
- [10] Institute of Electrical and Electronics Engineers. *ANSI/IEEE 754-1985 Standard for Binary Floating-Point Arithmetic*, 1985.
- [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, 1969.
- [17] B. D. Rao. Floating-point arithmetic and digital filters. *IEEE, Trans. 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. Mayerb. 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] S. Neumaier A. Rump S.P. Shary B. Kearfott, M. Nakao and P. Van Hentenryck. Standardized notation in interval analysis. *Reliable Computing*, 15:713, 2010.
- [23] W. T. Tucker and S. Ferson. *Probability bounds analysis in environmental risk assessments*. Applied Biomathematics, 100 North Country Road, Setauket, New York 11733.
- [24] L. H. de Figueiredo J. Stolfi. 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] Propagation of uncertainty. http://en.wikipedia.org/wiki/Propagation_of_uncertainty, 2011. wikipedia, the free encyclopedia.
- [27] Helen M. Regan, Scott Ferson, and Daniel Berleant. Equivalence of methods for uncertainty propagation of real-valued random variables. *International Journal of Approximate Reasoning*, 36:1–30, 2004.
- [28] C. P. Robert. *Monte Carlo Statistical Methods*. Springer, 2001.
- [29] Monte carlo method. http://en.wikipedia.org/wiki/Monte_Carlo_method, 2011. wikipedia, the free encyclopedia.
- [30] C. L. Smith. Uncertainty propagation using taylor series expansion and a spreadsheet. *Journal of the Idaho Academy of Science*, 30-2:93–105, 1994.
- [31] Significance arithmetic. http://en.wikipedia.org/wiki/Significance_arithmetic, 2011. wikipedia, the free encyclopedia.
- [32] Max Goldstein. Significance arithmetic on a digital computer. *Communications of the ACM*, 6:111–117, 1963.
- [33] R. L. Ashenurst and N. Metropolis. Unnormalized floating-point arithmetic. *Journal of the ACM*, 6:415–428, 1959.
- [34] 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.
- [35] Chengpu Wang. Error estimation of floating-point calculations by a new floating-point type that tracks the errors. In Hamid R. Arabnia and Iyad A. Ajwa, editors, *Proceedings of the 2005 International Conference on Algorithmic Mathematics and Computer Science, AMCS 2005*, pages 84–92, 2005.
- [36] A. Feldstein and R. Goodman. Convergence estimates for the distribution of trailing digits. *Journal of the ACM*, 23:287–297, 1976.
- [37] Integrating the bell curve. <http://www.mathpages.com/home/kmath045/kmath045.htm>, 2011. Math Pages.

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

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

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

- [41] Jim Hefferon. Linear algebra. <http://joshua.smcvt.edu/linearalgebra/>, 2011.

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

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

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

9 Tables

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

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

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

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

R_{max}	16	64	256
Bounding range Δx in term of deviation δx	$[4.9 \delta x, 9.8 \delta x]$	$[9.8 \delta x, 19.6 \delta x]$	$[19.6 \delta x, 39.2 \delta x]$
Maximal normalization leakage	10^{-6}	10^{-22}	10^{-85}
Bit calculated inside uncertainty	0	1	2
$0.5 \pm 0.001 =$	512?6.29@-10	1024?25.16@-11	2048?100.64@-12
$1 \pm 0.001 =$	1024?6.29@-10	2048?25.16@-11	4096?100.64@-12
$1 \pm 0.002 =$	512?6.29@-9	1024?25.16@-10	2048?100.64@-11

Table 3: Characterization of precision arithmetic with different R_{max} for precision representation $S \sim R @E$.

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

Table 4: Examples of precision arithmetic with different χ for $R_{max} = 16$, in which χ stands for bits calculated inside uncertainty.

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

Table 5: Uncertainty deviation and value error of numerical integration vs. expected results using precision arithmetic for different power function. The search range is deepest near 10^{-6} .

10 Figures

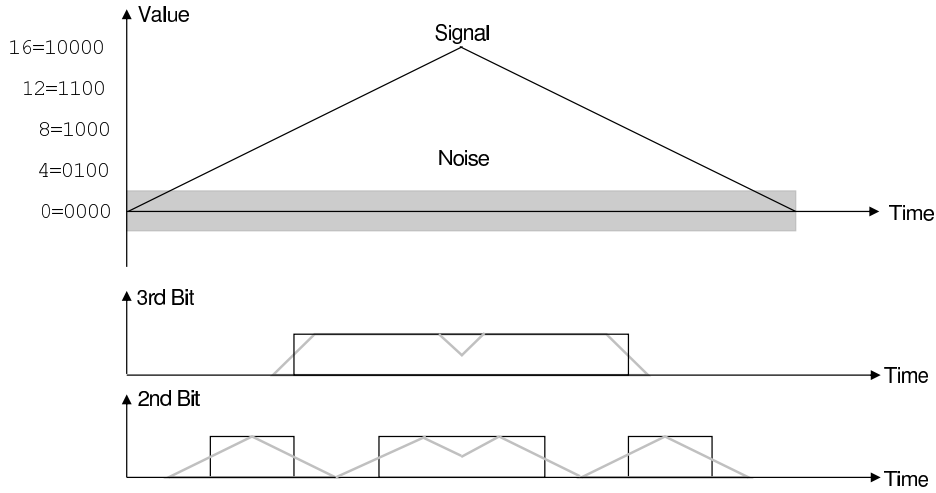


Figure 1: Effect of noise on bit values of a measured value. The triangular wave signal and the added white noise are shown at top using the thin black line and the grey area, respectively. The values are measured by a theoretical 4-bit DAC in ideal condition, assuming LSB is the 0th bit. The measured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

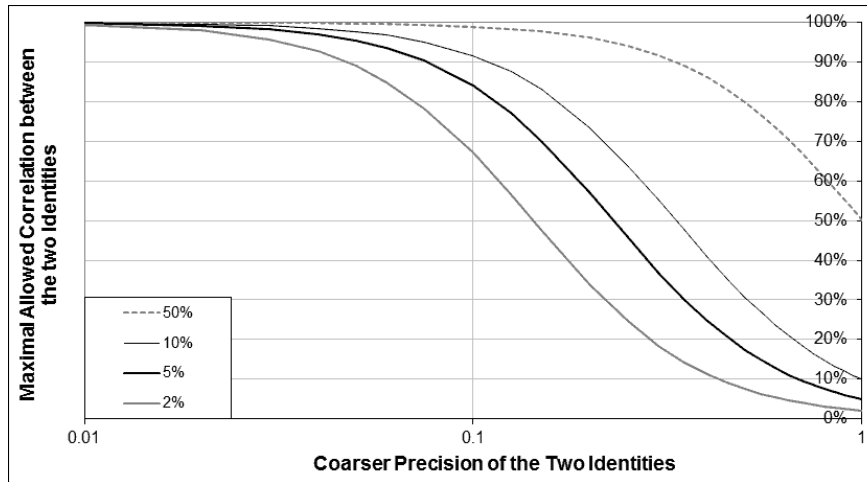


Figure 2: Allowed maximal correlation between two values vs. input precisions and independence standard (as shown in legend) for the independence uncertainty assumption of precision arithmetic to be true.

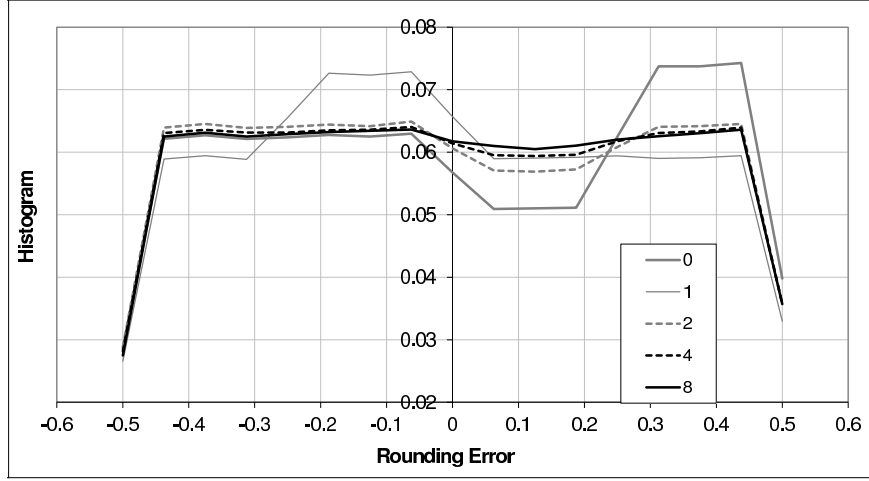


Figure 3: Measured probability distribution of rounding errors of precision round-up rule for the minimal significant thresholds 0, 1, 2, 4, and 8 respectively. Mathematically the probability is usually defined either in range $(-1/2, +1/2]$ or in range $[-1/2, +1/2)$, but not in range $[-1/2, +1/2]$.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

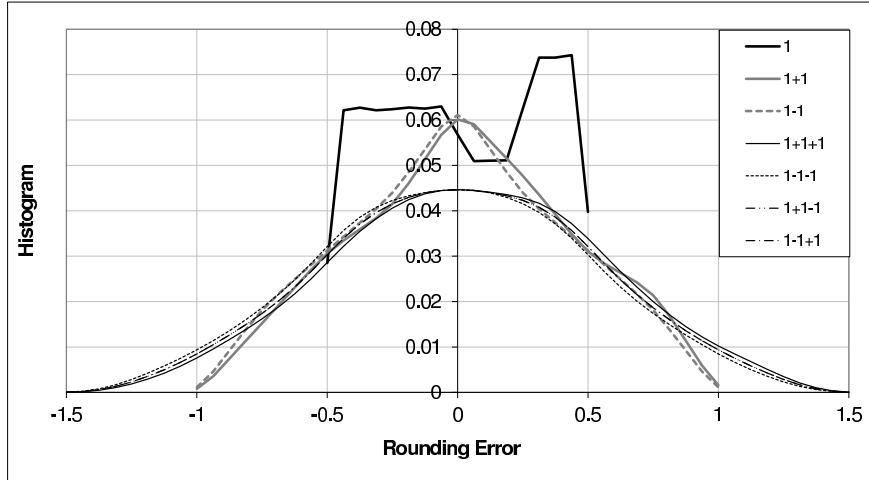


Figure 4: Measured probability distribution of the rounding error after addition and subtraction. In the legend, “1” for measured rounding error distribution for the minimal significant thresholds 0, “1+1” for addition once and “1-1” for subtraction once using the rounding error distribution of “1”,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

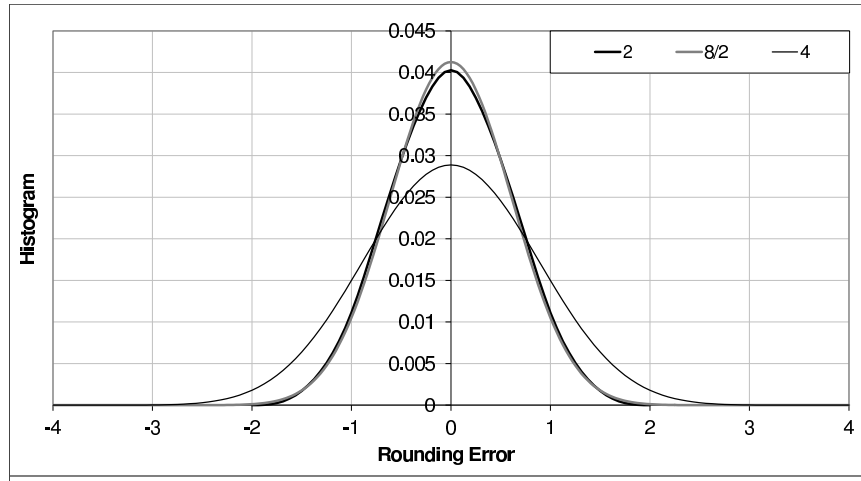


Figure 5: The result rounding error distribution $R = 8/2$ after the original error distribution $R = 8$ is rounded up once. The $R = 8/2$ distribution is compared with the $R = 4$ distribution and the $R = 2$ distribution, which have the same bounding range and deviation, respectively.

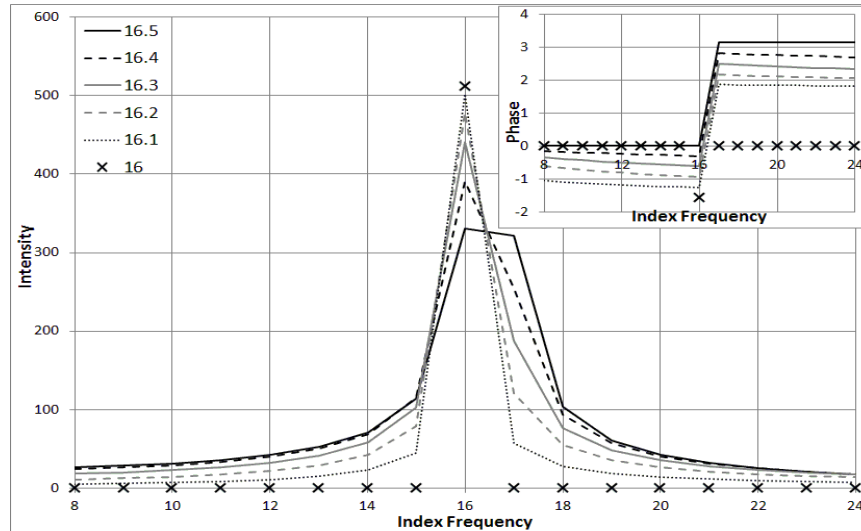


Figure 6: Unfaithful representations of perfect sine signals in the Discrete Fourier Transformation. The calculation is done on 1024 samples using FFT on a series of perfect sine signals having amplitude of 1 and slightly different frequencies as shown in legends. In the drawing, x axis shows frequency, y axis shows either intensity or phase (inlet). A faithful representation is also included for comparison, whose phase is $\pi/2$ at the index frequency, and undetermined at other frequencies.

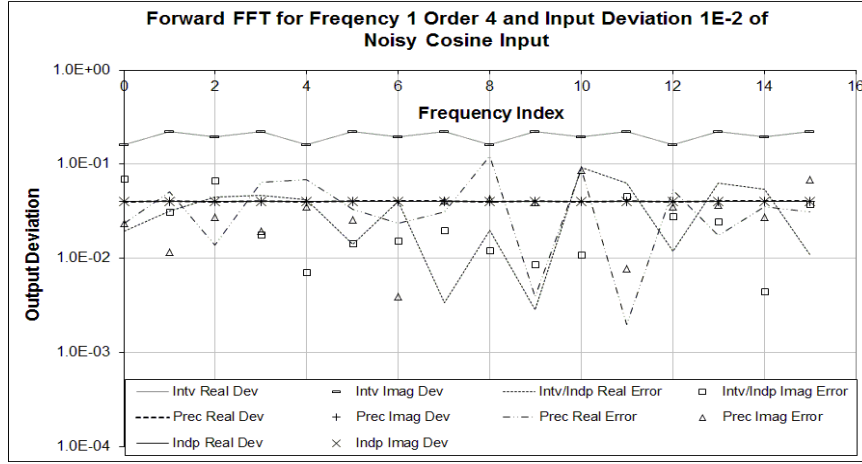


Figure 7: The output deviations and value errors of the forward FFT on a noisy sine signal of FFT order 4, index frequency 1 and input deviation 10^{-2} . In the legend, “Intv” means interval arithmetic, “Indp” means independence arithmetic, “Prec” means precision arithmetic, “Dev” means output uncertainty deviations, “Error” means output value errors, “Real” means real part, and “Imag” means imaginary part. Because both interval arithmetic and independence arithmetic using conventional floating arithmetic for underline calculations, they have the same value errors.

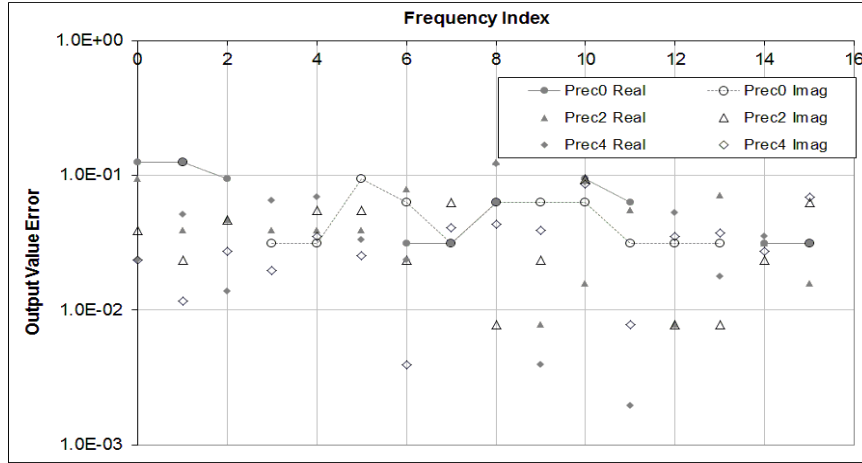


Figure 8: The output value errors of the forward FFT on a noisy sine signal of index frequency 1 and input deviation 10^{-2} using precision arithmetic with different bit inside uncertainty. In the legend, “Prec0” means precision arithmetic with 0-bit calculated inside uncertainty, “Prec2” means precision arithmetic with 2-bit calculated inside uncertainty, and “Prec4” means precision arithmetic with 4-bit calculated inside uncertainty.

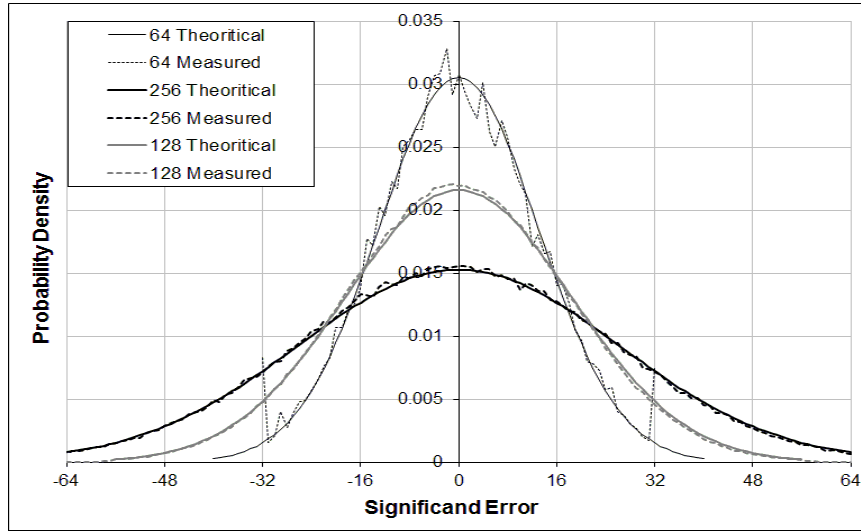


Figure 9: The theoretical and empirical distribution of significant errors for different bounding ranges (as shown in legend) using precision arithmetic for the forward FFT on noisy sine signals.

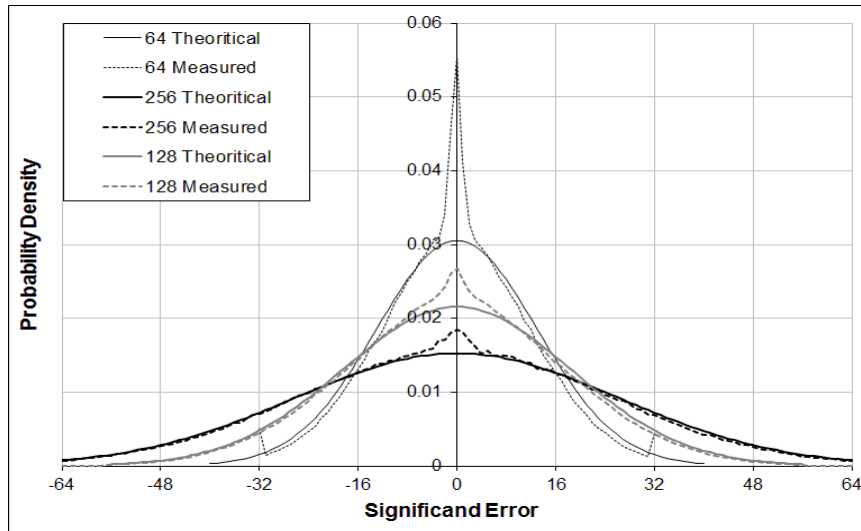


Figure 10: The theoretical and empirical distribution of significant errors for different bounding ranges (as shown in legend) using precision arithmetic for the reverse FFT on noisy sine signals.

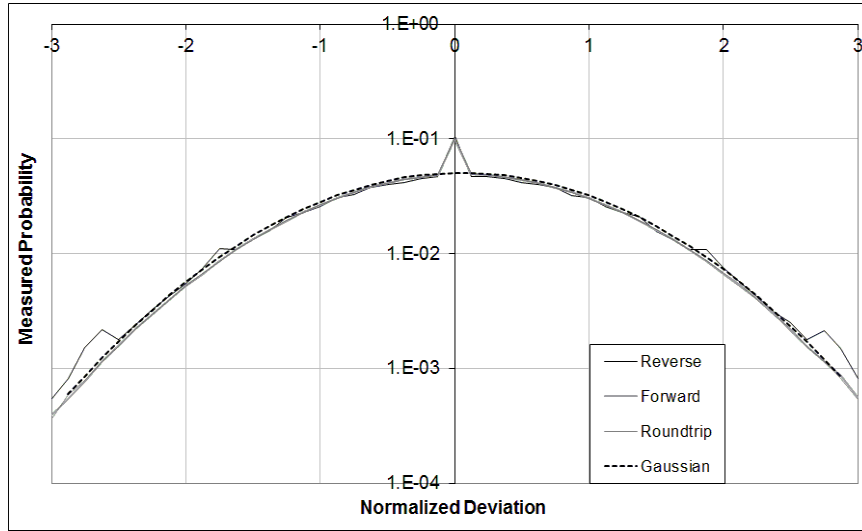


Figure 11: The measured normalized value error distributions using independence arithmetic for FFT algorithms (as shown in legend). They are best fitted by a Gaussian distribution with the mean of 0.06 and deviation of 0.98.

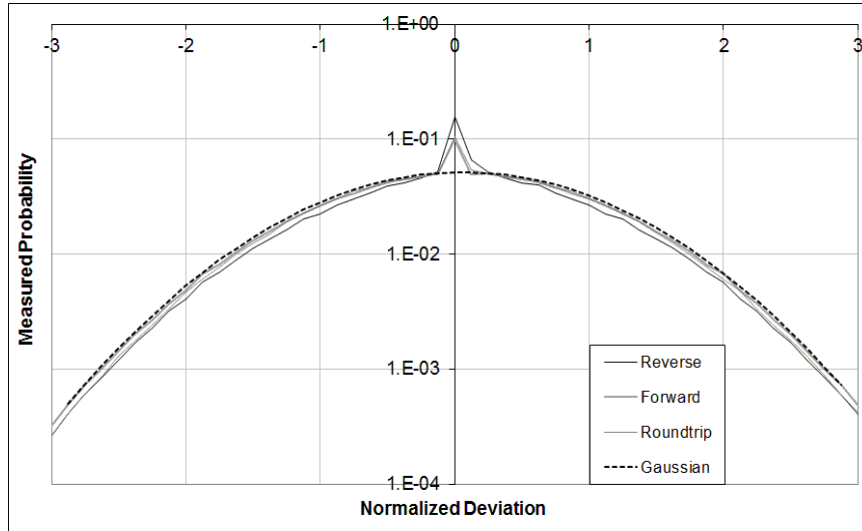


Figure 12: The measured normalized value error distributions using precision arithmetic for FFT algorithms (as shown in legend). They are best fitted by a Gaussian distribution with the mean of 0.06 and deviation of 0.97.

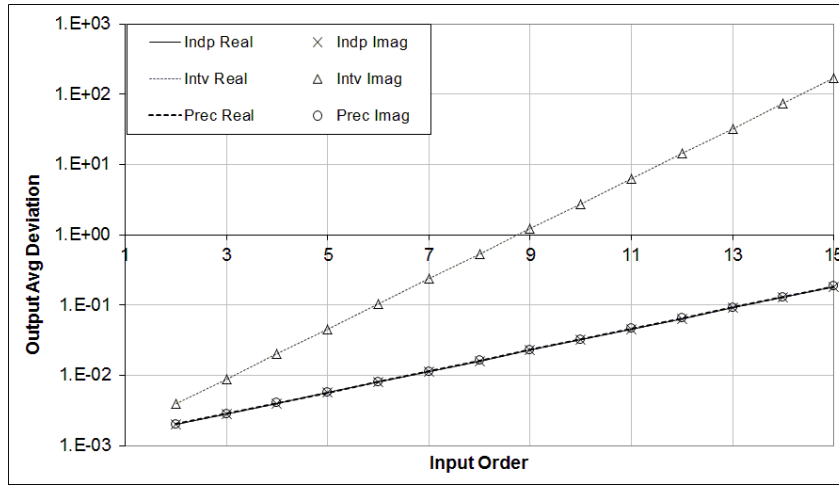


Figure 13: For the same input deviation of 10^{-3} , the empirical average output deviations of the forward FFT increase exponentially with the FFT order for all uncertainty-bearing arithmetics. In the legend, “Intv” means interval arithmetic, “Indp” means independence arithmetic, “Prec” means precision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

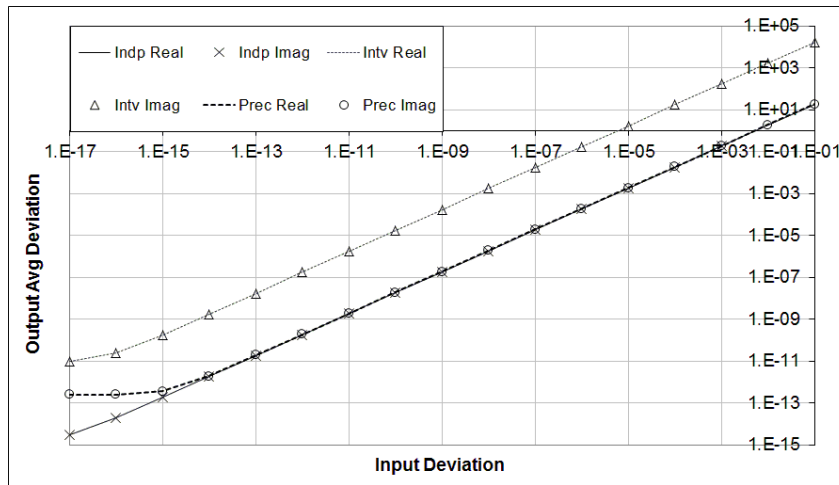


Figure 14: For the same order of the FFT calculation of 15, the empirical average output deviations of the forward FFT increases linearly with the input deviation for all uncertainty-bearing arithmetics. In the legend, “Intv” means interval arithmetic, “Indp” means independence arithmetic, “Prec” means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

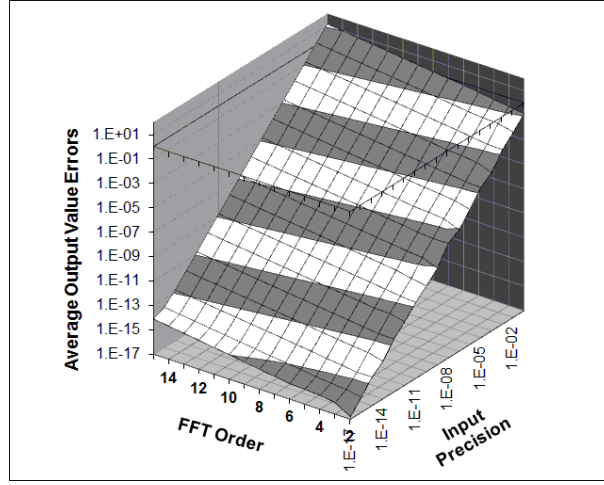


Figure 15: The empirical average output value errors using precision arithmetic increase exponentially with the FFT order and linearly with the input deviation, respectively.

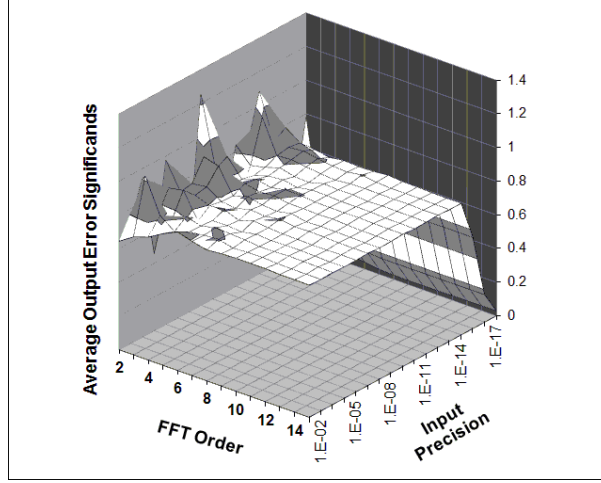


Figure 16: The empirical average output error significands using precision arithmetic is a constant when the input deviation is larger than 10^{-14} and the FFT order is more than 5 for forward FFT algorithms. Because the precision of conventional floating-point representation is at 10^{-16} , adding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

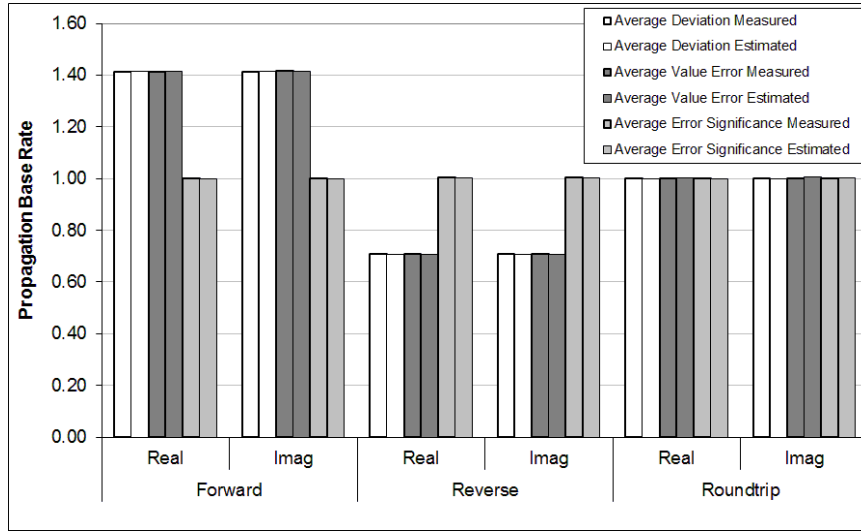


Figure 17: Empirical and theoretical β for fitting average output deviations, value errors and error significands for forward, reverse and roundtrip FFT using independence arithmetic on noisy sine signals. In the chart, “Real” means real part, and “Imag” means imaginary part.

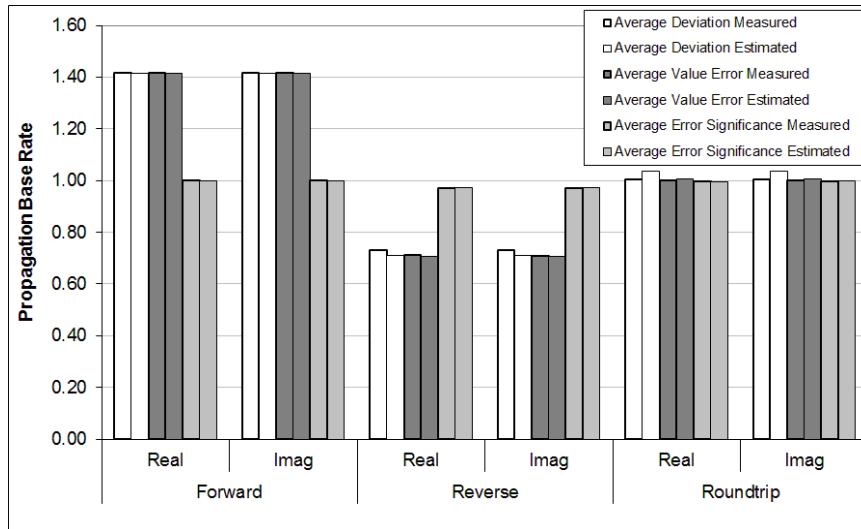


Figure 18: Empirical and theoretical β for fitting average output deviations, value errors and error significands for forward, reverse and roundtrip FFT using precision arithmetic on noisy sine signals. In the chart, “Real” means real part, and “Imag” means imaginary part.

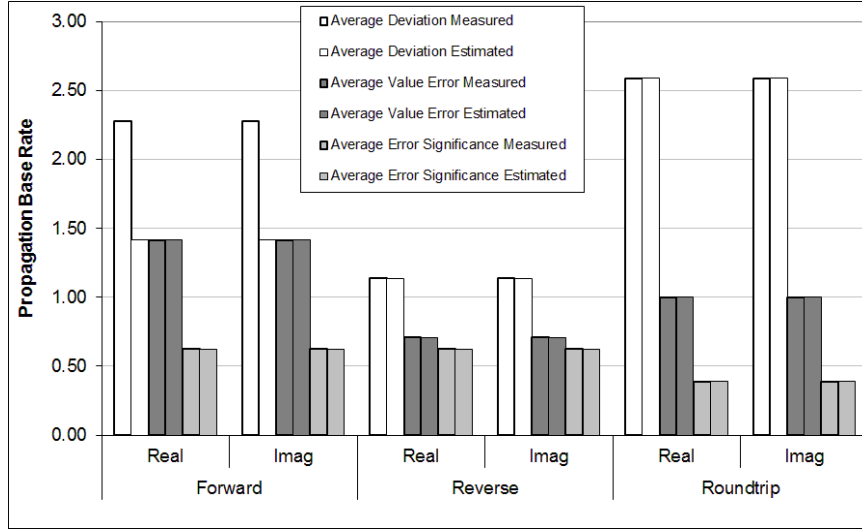


Figure 19: Empirical and theoretical β for fitting average output deviations, value errors and error significands for forward, reverse and roundtrip FFT using interval arithmetic on noisy sine signals. In the chart, “Real” means real part, and “Imag” means imaginary part.

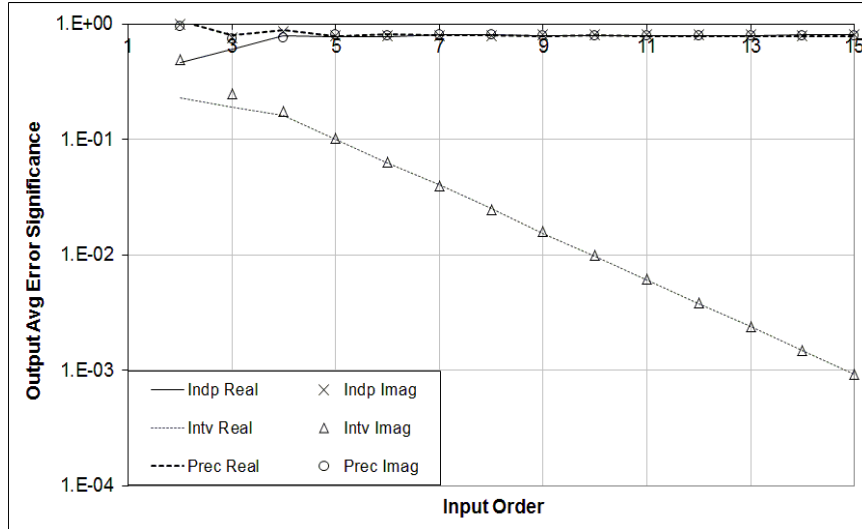


Figure 20: The empirical output average error significands vs. the FFT order of the forward FFT for all three arithmetics when the input uncertainty deviation is 10^{-3} . In the legend, “Intv” means interval arithmetic, “Indp” means independence arithmetic, “Prec” means precision arithmetic, “Real”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

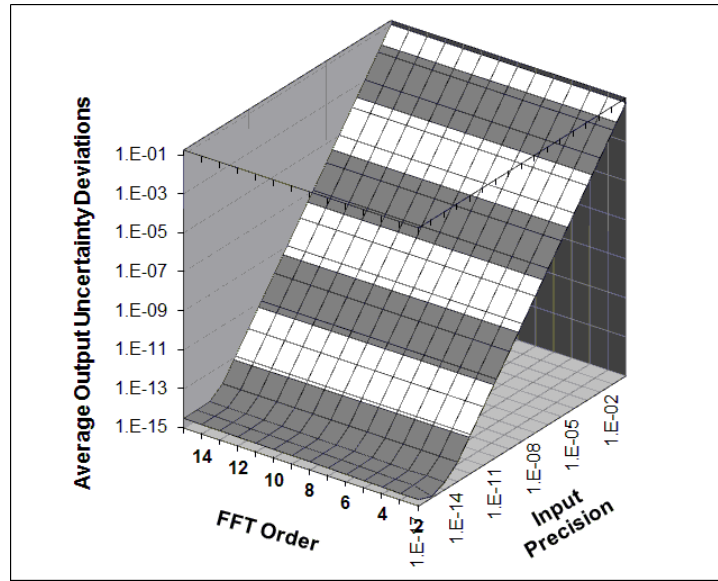


Figure 21: The empirical average output deviations vs. the FFT order and input deviations using precision arithmetic for the round-trip FFT algorithm.

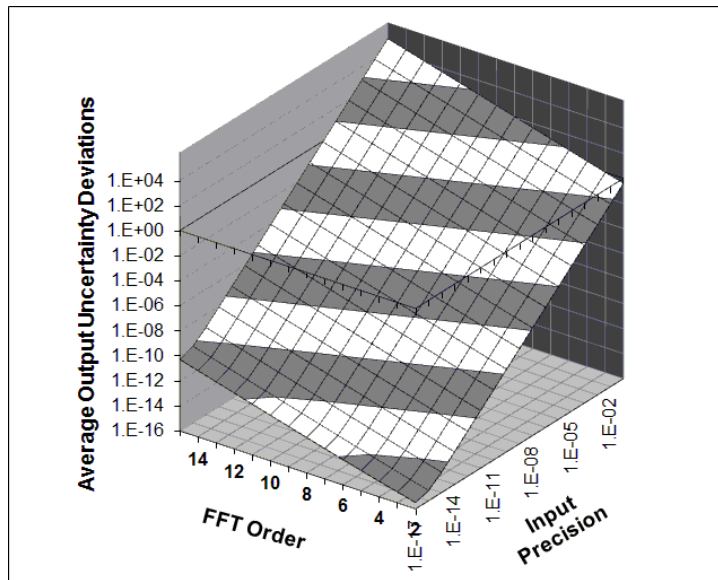


Figure 22: The empirical average output deviations vs. the FFT order and input deviations using interval arithmetic for the round-trip FFT algorithm.

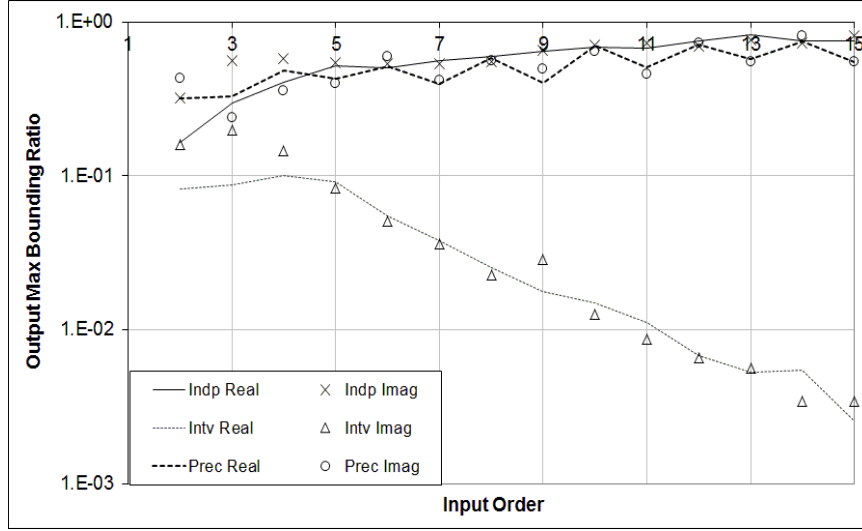


Figure 23: The empirical maximal output bounding ratios vs. the FFT order of the forward FFT for all three arithmetics. In the legend, “Intv” means interval arithmetic, “Indp” means independence arithmetic, “Prec” means precision arithmetic, “Real” means real part, and “Imag” means imaginary part.

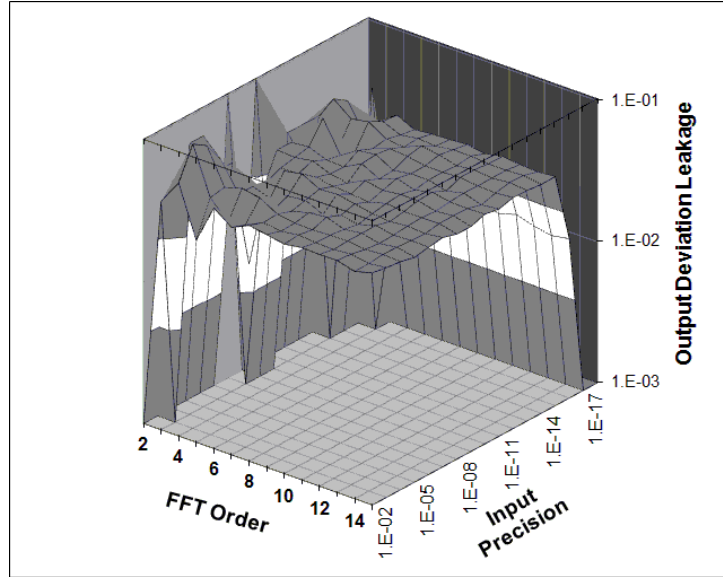


Figure 24: The empirical deviation leakages vs. the FFT order and input deviations using precision arithmetic for the forward FFT algorithm.

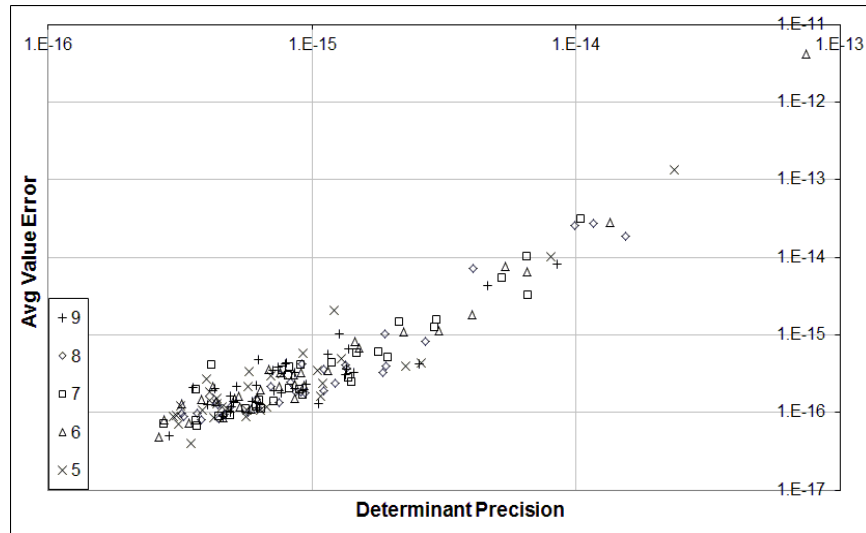


Figure 25: The empirical average value errors of the inverted matrix using conventional floating-point arithmetic vs. matrix determinant precision using precision arithmetic for clean matrices of different sizes (as shown in legend).

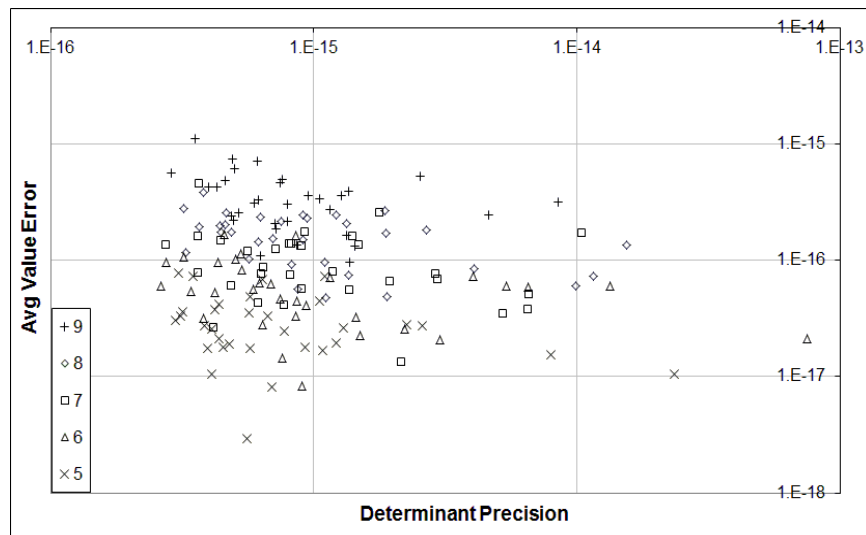


Figure 26: The empirical average value errors of the adjugate matrix using conventional floating-point arithmetic vs. matrix determinant precision using precision arithmetic for clean matrices of different sizes (as shown in legend).

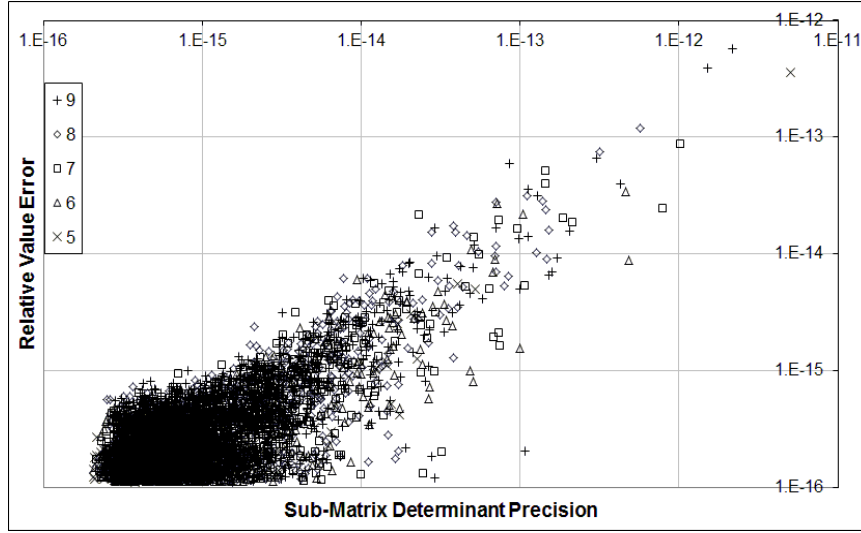


Figure 27: Empirical Relative value errors of the adjugate matrix using conventional floating-point arithmetic vs. corresponding sub-matrix determinant precision using precision arithmetic for clean matrices of different sizes (as shown in legend).

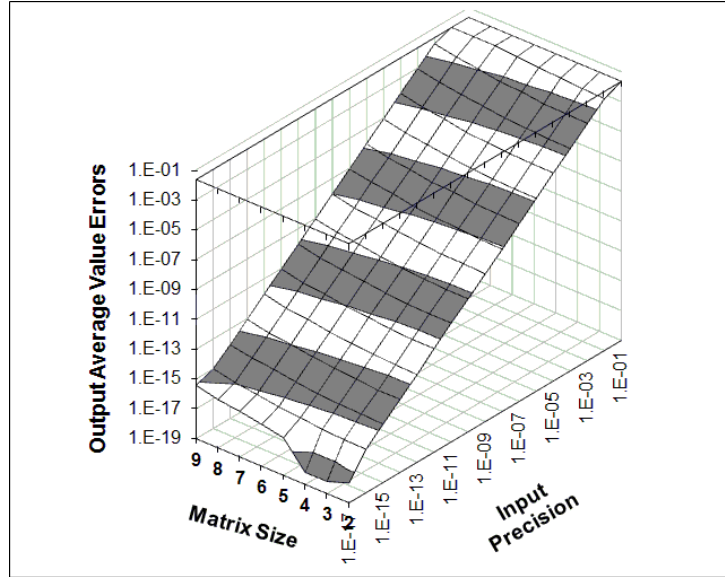


Figure 28: Using precision arithmetic, the average output deviations of the adjugate matrix vs. input precision and the matrix size.

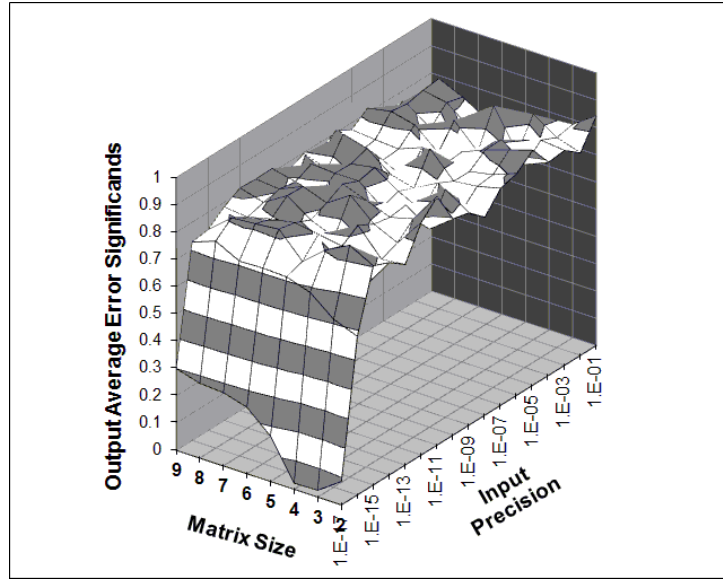


Figure 29: Using precision arithmetic, the average output error significands of the adjugate matrix vs. input precision and the matrix size.

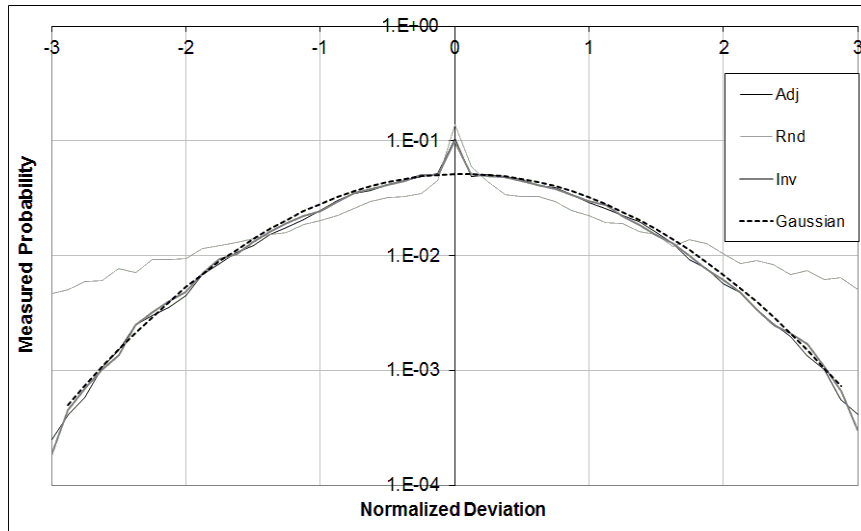


Figure 30: The measured normalized value error distributions using precision arithmetic for matrix calculations of matrix size 9. They are best fitted by a Gaussian distribution with the mean of 0.06 and deviation of 0.96. In the legend, “Adj” means calculating adjugate M^A , “Inv” means calculating inverted M^{-1} , and “Rnd” means calculating double inverted $(M^{-1})^{-1}$. Please also see Footnote 7 for additional explanations.

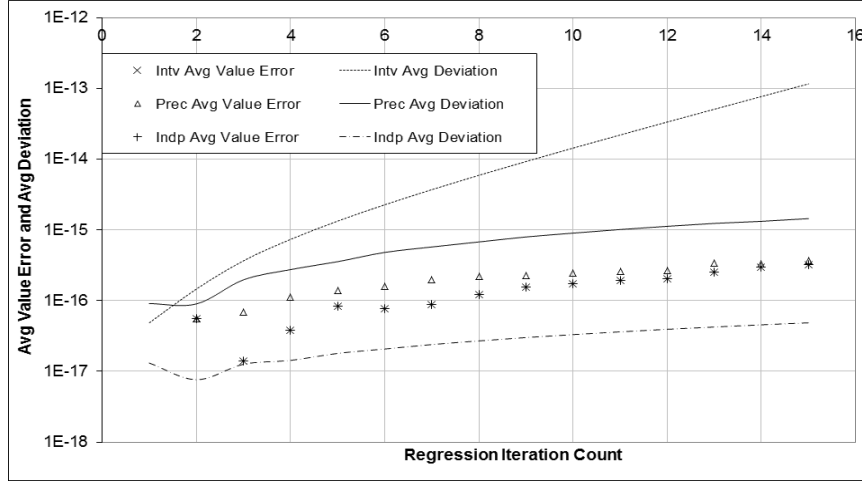


Figure 31: The empirical output average value errors and corresponding average output deviations vs. the regression iteration count of the regressive calculation of sine values using interval arithmetic, precision arithmetic and independent arithmetic. The x-axis indicates the regression iteration count L , while the y-axis indicates either the average value errors or average uncertainty deviations. In the legend, “Intv” means interval arithmetic, “Indp” means independence arithmetic, and “Prec” means precision arithmetic.

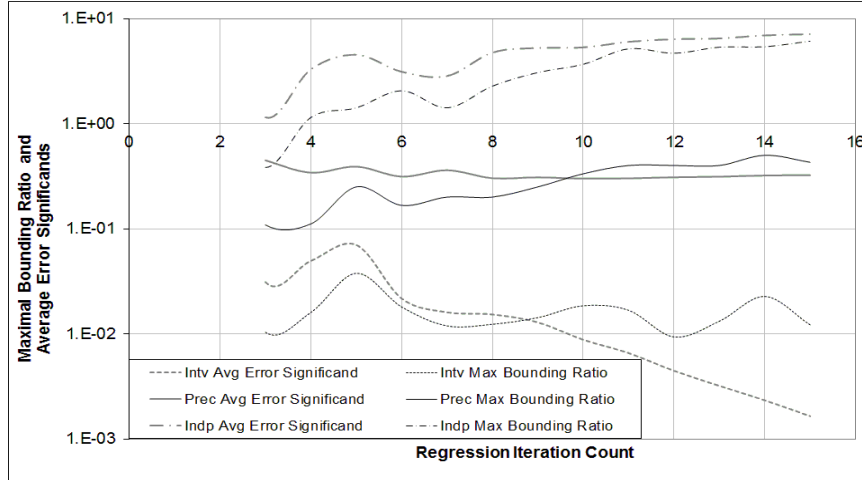


Figure 32: The empirical output maximal bounding ratios and average error significands vs. the regression iteration count of the regressive calculation of sine values using interval and precision arithmetics. In the legend, “Intv” means interval arithmetic, “Indp” means independence arithmetic, and “Prec” means precision arithmetic.