TEMA Tend. Mat. Apl. Comput., **4**, No. 3 (2003), 297-312.

An Introduction to Affine Arithmetic

J. STOLFI[1], Instituto de Computação, UNICAMP, Cx.P. 6176, 13084-971 Campinas, SP, Brazil

L.H. de FIGUEIREDO[2], IMPA - Instituto de Matemática Pura e Aplicada, Estrada Dona Castorina 110, 22460-320 Rio de Janeiro, RJ, Brazil.

Abstract. Affine arithmetic (AA) is a model for self-validated computation which, like standard interval arithmetic (IA), produces guaranteed enclosures for computed quantities, taking into account any uncertainties in the input data as well as all internal truncation and roundoff errors. Unlike standard IA, the quantity representations used by AA are first-order approximations, whose error is generally quadratic in the width of input intervals. In many practical applications, the higher asymptotic accuracy of AA more than compensates for the increased cost of its operations.

1. Introduction

A *self-validated* (SV) numerical algorithm keeps track of the accuracy of all quantities that it computes, as part of the process of computing them. So, even if one cannot pre-determine a general upper bound for the numerical error of the result, at least one can obtain such a bound *a posteriori* for each computation.

Self-validated computation was originally developed as a means of certifying numerical algorithms used in real-world applications, which are often too complex to allow theoretical error analysis. Of course, an SV algorithm is not suitable for critical applications where one needs *a priori* accuracy guarantees. Typically, an SV algorithm is embedded in a system that can take appropriate remedial action whenever the computed error bounds are too large. (If the error is mainly due to internal approximations and rounding, one option is to repeat the SV computation with increased precision, until the output error estimates are small enough. This idea is the foundation of *lazy real arithmetic* [35].)

The techniques that SV algorithms use for keeping track of internal approximation errors can also encode the uncertainty in the computed quantities that comes from external causes, such as measurement errors, manufacturing tolerances, adjustable parameters, or other unknown deviations in the input data. The final error bounds computed by the algorithm will then automatically account for those external factors. In fact, SV algorithms are often used not so much to bound computation

[1]stolfi@ic.unicamp.br. Supported in part by CNPq and FAPESP.
[2]lhf@visgraf.impa.br. Supported in part by CNPq.

errors, but for *range analysis* — finding *guaranteed* upper and lower bounds for the value of a mathematical function over a specified region of its domain.

A *self-validated computation model* is a general method for constructing SV algorithms for arbitrary mathematical formulas. The canonical self-validated computation model is *interval arithmetic* [38], which we describe in section 2. Several other models have been designed to produce more accurate error estimates, usually by storing more information about the computed quantities and how they relate to the input data. In section 3 we describe one particular such model, *affine arithmetic* [5], which has been used with advantage in a number of practical applications.

2. Interval arithmetic

In *interval arithmetic* (IA), also known as *interval analysis*, a real quantity x is represented by an interval $\bar{x} = [x_{\min} \_\_ x_{\max}]$ of floating-point numbers. Those intervals are added, subtracted, multiplied, etc., in such a way that each computed interval $\bar{x}$ is guaranteed to contain the corresponding *ideal quantity* x — the (unknown) real value of the corresponding variable in the exact (error-free, uncertainty-free) computation which is being approximated. For example, if the ideal quantities x and y are known to lie in the intervals $\bar{x} = [2 \_\_ 4]$ and $\bar{y} = [-3 \_\_ 2]$, then the sum $x + y$ lies in the interval $\bar{x} + \bar{y} = [2 - 3 \_\_ 4 + 2] = [-1 \_\_ 6]$, and the product xy in $[4 \cdot -3 \_\_ 4 \cdot 2] = [-12 \_\_ 8]$. (Note that any roudoff errors in the computed bounds must be directed so as to displace the endpoints outwards.)

Interval analysis was formalized by Ramon E. Moore in the 1960s [38, 39]. After some three decades of neglet — if not outright prejudice — by numerical analysts, IA was "rediscovered" by researchers from many applied fields, who found it to be a flexible and effective tool for rangeanalysis. This revival of IA was greatly helped by the widespread acceptance of the IEEE Floating-Point Standard [24], whose directed-rounding capabilities allowed efficient and machine-independent bounding of roundoff errors.

Successful applications of IA now include, for example, robust root finders for ray tracing [36, 2], domain enumeration for solid modeling, [11, 40, 45, 47, 48], surface intersection [14], global optimization [17, 18, 19, 23, 26, 37, 43, 42, 49]. Interval computations recently settled the *double bubble conjecture* [21], a longstanding open problem in the theory of minimal surfaces. Several good-quality portable IA libraries are available [28, 29, 27, 1]. Interval arithmetic and related techniques now have a dedicated journal [44], a central web site containing a wealth of information and links [30], and several established conferences.

2.1. The interval overestimation problem

Unfortunately, IA often yields an interval that is much wider than the exact range of the computed function. As an extreme example, the IA evaluation of $x - x$ given $x \in \bar{x} = [2 \_\_ 5]$ produces $[2 - 3 \_\_ 5 - 2] = [-3 \_\_ +3]$ — instead of $[0 \_\_ 0]$, which is the actual range of that formula. Note that the IA subtraction routine cannot

assume that the two given intervals denote the same ideal quantity, since they could also denote two independent quantities that happen to have the same range.

More generally, if the arguments of an operation $z \leftarrow f(x, y)$ are somewhat correlated, the interval $\bar{z} \leftarrow \bar{f}(\bar{x}, \bar{y})$ computed with IA may be significantly wider than the actual range of the ideal quantity z. We may define the *IA overestimation factor* σ of that operation as the width of the computed interval $\bar{z}$ divided by the width of the exact range of z, assuming that $\bar{x}$ and $\bar{y}$ are the true ranges of x and y. It turns out that, for intervals centered at any fixed values x and y, the factor σ depends on the amount and sign of correlation, and on the derivatives of f, but is roughly independent of the width of the operands. For example, when computing $x(10 - x)$ for x in the range [3 __ 5], IA yields the interval [15 __ 35], which is 5 times wider the exact range [21 __ 25]. If the input interval is reduced to [3.9 __ 4.1], the IA result will be [23.01 __ 25.01], still 5 times wider than the exact range [23.79 __ 24.19].

This overestimation problem has hampered the use of IA in many potential applications. In chained computations, where the results of one step are inputs for the next step, the overestimation factors of the individual steps tend to get multiplied. As a consequence, the final intervals may easily become too wide to be useful — by several orders of magnitude. See figure 1. To obtain a useful range estimate with IA in such cases, one would have to partition the given interval into thousands of sub-intervals, and repeat the evaluation on each part.

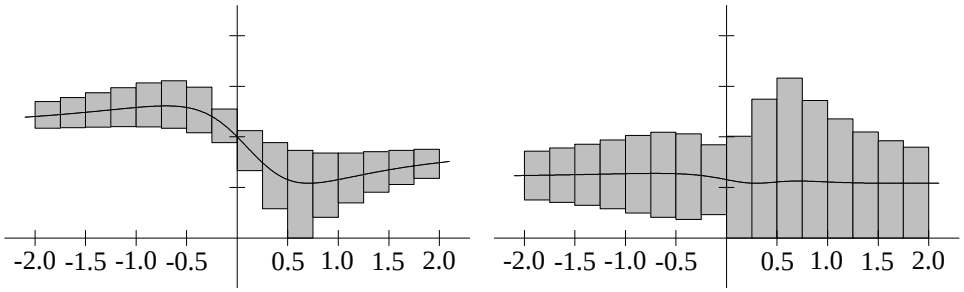

Figure 1: Error explosion in IA estimates for iterated functions. **Left:** The function $g(x) = \sqrt{x^2 - x + 1/2}/\sqrt{x^2 + 1/2}$ (black curve) and the result of evaluating $g(\bar{x})$ with standard IA, for 16 consecutive intervals $\bar{x}$ of width 0.25 in $[-2 \_\_ +2]$. **Right:** The same data for the second iterate $h(x) = g(g(x))$ of $g(x)$. Note that the iterated interval evaluation $\bar{g}^k(\bar{x})$ diverges, even though the iterates $g^k(x)$ of the original function quickly converge to a constant (≈ 0.559).

3. Affine arithmetic

Affine arithmetic (AA) is one of several SVC models that were proposed to overcome the error explosion problem of standard IA [5, 10, 8]. Besides recording a range for each ideal quantity, affine arithmetic also keeps track of *correlations* between those quantities. Thanks to this extra information, the approximation error incurred in each AA operation normally has a *quadratic* dependency on the size of

the input intervals — even when the operands are correlated, as in the $x(10 - x)$ example above. Therefore, if the input intervals are small enough, each operation will provide a fairly tight estimate of the exact range of the corresponding quantity, with overestimation factor near 1 (except near stationary points). This benefit will hold also for many chained computations where IA undergoes error explosion.

In affine arithmetic, an ideal quantity x (given or computed) is represented by an *affine form* $\hat{x}$, which is a first-degree polynomial:

$$\hat{x} = x_0 + x_1\varepsilon_1 + x_2\varepsilon_2 + \cdots + x_n\varepsilon_n. \tag{3.1}$$

The coefficients x_i are finite floating-point numbers, and the ε_i are symbolic real variables whose values are unknown but assumed to lie in the interval $\mathbb{U} = [-1 \_\_ +1]$. We call x_0 the *central value* of the affine form $\hat{x}$; the coefficients x_i are its *partial deviations*, and the ε_i are the *noise symbols*.

Each noise symbol ε_i stands for an independent component of the total uncertainty of the ideal quantity x; the corresponding coefficient x_i gives the magnitude of that component. The sources of this uncertainty may be "external" (already present in the input data) or "internal" (due to arithmetic roundoff or approximations in the computation of $\hat{x}$).

The *fundamental invariant of affine arithmetic* states that, at any instant between AA operations, there is a single assignment of values from $\mathbb{U}$ to each of the noise variables in use that makes the value of every affine form $\hat{x}$ equal to the true value of the corresponding ideal quantity x [10].

3.1. Conversions between IA and AA

Every affine form $\hat{x} = x_0 + x_1\varepsilon_1 + \cdots + x_n\varepsilon_n$ implies an interval bound for the corresponding ideal quantity x: namely, $x \in \bar{x} = [x_0 - r \_\_ x_0 + r]$, where r is the *total deviation* of $\hat{x}$, $\sum_{i=1}^{n} |x_i|$. This is the smallest interval that contains all possible values of $\hat{x}$, assuming that each ε_i ranges independently over the interval $\mathbb{U} = [-1 \_\_ +1]$. Note that the bounds of $\bar{x}$ must be rounded outwards, and that this conversion discards all the correlation information present in $\hat{x}$.

Conversely, every ordinary interval bound $\bar{x} = [a \_\_ b]$ for an ideal quanity x can be replaced by an affine form $\hat{x} = x_0 + x_k\varepsilon_k$, where x_0 is the midpoint $(a + b)/2$ of $\bar{x}$, x_k is the half-width $(b - a)/2$, and ε_k is a new noise symbol, not occurring in any other existing affine form. The new symbol ε_k represents the uncertainty in the value of x that is implicit in its interval representation $\bar{x}$. Again, note that x_0 and x_k must be carefully rounded, and that the new affine form, like the interval $\bar{x}$, carries no correlation informtion.

3.2. Joint range of affine forms

The key feature of AA is that the same noise symbol ε_i may contribute to the uncertainty of two or more quantities (inputs, outputs, or intermediate results) $\hat{x}$ and $\hat{y}$ arising in the evaluation of an expression. The sharing of noise symbols indicates some partial dependency between the underlying quantities x and y, determined by

the corresponding coefficients x_i and y_i. Note that the signs of these coefficients are not meaningful in themselves, because the sign of ε_i is arbitrary; but the relative sign of x_i and y_i defines the direction of the correlation. For example, suppose that the quantities x and y are represented by the affine forms

$$\begin{aligned} \hat{x} &= 20 - 4\varepsilon_1 + 2\varepsilon_3 + 3\varepsilon_4 \\ \hat{y} &= 10 - 2\varepsilon_1 + 1\varepsilon_2 - 1\varepsilon_4 \end{aligned}$$

From this data, we can tell that x lies in the interval $\bar{x} = [11 \_\_ 29]$ and y lies in $\bar{y} = [6 \_\_ 14]$; i.e., the pair (x, y) lies in the grey rectangle of figure 2. However, since the two affine forms include the same noise variables ε_1 and ε_4 with non-zero coefficients, they are not entirely independent of each other. In fact, the pair (x, y) must lie in the dark grey region of figure 2, which is the set of all possible values of $(\hat{x}, \hat{y})$ when the noise variables $\varepsilon_1, .. \varepsilon_4$ are independently chosen in $\mathbb{U}$. This set is the *joint range* of the forms $\hat{x}$ and $\hat{y}$, denoted $\langle \hat{x}, \hat{y} \rangle$.

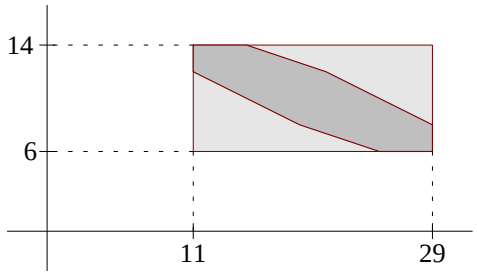

Figure 2: Joint range $\langle \hat{x}, \hat{y} \rangle$ of two partially dependent quantities as implied by their affine forms $\hat{x} = 20 - 4\varepsilon_1 + 2\varepsilon_3 + 3\varepsilon_4$ and $\hat{y} = 10 - 2\varepsilon_1 + 1\varepsilon_2 - 1\varepsilon_4$.

As can be inferred from figure 2, the set $\langle \hat{x}, \hat{y} \rangle$ is a convex polygon, symmetric around its center (x_0, y_0). If the forms depend on n noise symbols $\varepsilon_1, .. \varepsilon_n$, the joint range has $2n$ sides; each ε_i corresponds to a pair of opposite sides, which are parallel and congruent to the segment with endpoints (x_i, y_i) and $(-x_i, -y_i)$. In fact, the joint range $\langle \hat{x}, \hat{y} \rangle$ is the Minkowski sum [16] of all those segments with the point (x_0, y_0). The $2n$ vertices of $\langle \hat{x}, \hat{y} \rangle$ are the corners of the convex hull of the 2^n points $(\hat{x}, \hat{y})$ that are obtained by setting each ε_i to -1 or $+1$, in all possible combinations.

Similarly, the joint range $\langle \hat{x}, \hat{y}, \hat{z} \rangle$ of three affine forms $\hat{x}$, $\hat{y}$ and $\hat{z}$ is a convex polyhedron, center-symmetric around the point (x_0, y_0, z_0), with $\Theta(n^2)$ vertices, edges and faces in the worst case.

In general, any m affine forms $\hat{x}^1, .. \hat{x}^m$ determine a *joint range* $\langle \hat{x}^1, .. \hat{x}^m \rangle \subseteq \mathbb{R}^m$, defined as the set of all tuples $(x^1, .. x^m)$ of values for the corresponding ideal quantities that are simultaneously compatible with those affine forms. Note that $\langle \hat{x}^1, .. \hat{x}^n \rangle$ is the parallel projection on $\mathbb{R}^m$ of the hypercube $\mathbb{U}^n$ by the affine map $(\hat{x}^1, .. \hat{x}^m)$. The projection is a *zonotope*, a center-symmetric convex polytope in $\mathbb{R}^m$, whose faces are themselves center-symmetric. It is the Minkowski sum of the point $(x_0^1, .. x_0^m)$ and the n segments s_i with endpoints $(x_i^1, .. x_i^n)$ and $(-x_i^1, .. -x_i^n)$.

3.3. Computing with AA

In order to evaluate a formula with AA, we must replace each elementary operation $z \leftarrow f(x, y)$ on real quantities x and y by a corresponding procedure $\hat{z} \leftarrow \hat{f}(\hat{x}, \hat{y})$, which takes affine forms of those quantities and returns an affine form for the result z. By definition, there are (unknown) values of $\varepsilon_1, .. \varepsilon_n \in \mathbb{U}^n$ such that

$$x = x_0 + x_1\varepsilon_1 + .. x_n\varepsilon_n \tag{3.2}$$

$$y = y_0 + y_1\varepsilon_1 + .. y_n\varepsilon_n \tag{3.3}$$

Therefore, the exact result z is a function of the unknowns ε_i, namely

$$\begin{aligned} z &= f(x, y) \\ &= f(x_0 + x_1\varepsilon_1 + .. x_n\varepsilon_n,\ y_0 + y_1\varepsilon_1 + .. y_n\varepsilon_n.) \end{aligned} \tag{3.4}$$

The challenge now is to replace $f(x, y)$ by an affine form

$$\hat{z} = z_0 + z_1\varepsilon_1 + .. z_m\varepsilon_m$$

for some $m \geq n$, that preserves as much information as possible about the constraints between x, y, and z that are implied by (3.2–3.4). Note that this approach can be applied to operations with any number of arguments.

3.3.1. Affine operations

If the operation f itself is an affine function of its arguments x and y, then formula (3.4) can be expanded and rearranged into an affine combination of the noise symbols ε_i. Namely, for any given constants $\alpha, \beta, \zeta \in \mathbb{R}$, the computation $z \leftarrow \alpha x + \beta y + \zeta$ can be carried out as

$$\hat{z} \leftarrow \alpha\hat{x}+\beta\hat{y}+\zeta = (\alpha x_0+\beta y_0+\zeta) + (\alpha x_1+\beta y_1)\varepsilon_1 + \cdots + (\alpha x_n+\beta y_n)\varepsilon_n. \tag{3.5}$$

Except for roundoff errors and overflows, the affine form $\hat{z}$ above, together with $\hat{x}$ and $\hat{y}$, captures all the information about the quantities x, y, and z that can be deduced from the given affine forms $\hat{x}$ and $\hat{y}$, and the equation $z = \alpha x + \beta y + \zeta$.

Observe that the computation of $\hat{x} - \hat{x}$ by formula (3.5) yields exactly zero. Since the two operands use the same noise symbols with the same coefficients, the AA subtraction procedure "knows" that they are actually the same ideal quantity, and not just two independent quantities that happen to have the same range. By the same token, linear identities such as $(\hat{x} + \hat{y}) - \hat{x} = \hat{y}$ or $(3\hat{x}) - \hat{x} = 2\hat{x}$, which do not hold in IA, *do* hold in AA (except for floating-point roundoff errors).

Thus, an AA computation that uses only affine operations with known constant coefficients will usually give an almost-exact range for the ideal result. In fact, it will give an almost-exact explicit formula for the ideal result in terms of the input variables — more precisely, in terms of the ε_i that occur in the input forms.

3.3.2. Handling roundoff errors

Formula (3.5) ignores floating-point roundoff errors, which must be taken into account in order to preserve the fundamental invariant of AA (see section 3). One might think that (as in IA) it suffices to round each coefficient z_i of the result $\hat{z}$ in the "safe" direction, namely away from zero. However, if the noise variable ε_i occurs in some other affine form $\hat{w}$, then any error in z_i — in either direction — would imply a different correlation between the quantities z and w, and would falsify the fundamental invariant.

The correct way to handle roundoff errors is to extend the resulting form $\hat{z}$ with an additional term $z_k\varepsilon_k$, where z_k is an upper bound for the sum of all absolute errors d_i incurred in the computation of the coefficients z_i, and ε_k is a noise symbol that does not occur in any other affine form. Note that we *are* allowed to round z_k conservatively (away from zero), since that term, which uses a brand-new symbol ε_k, does not imply any correlations.

3.3.3. Non-affine operations

Consider now a generic non-affine operation $z \leftarrow f(x, y)$. If x and y are represented by the affine forms $\hat{x}$ and $\hat{y}$, then the ideal quantity z is given by the formula

$$\begin{aligned} z &= f(x_0 + x_1\varepsilon_1 + .. \, x_n\varepsilon_n, \;\; y_0 + y_1\varepsilon_1 + .. \, y_n\varepsilon_n) \\ &= f^*(\varepsilon_1, .. \, \varepsilon_n), \end{aligned} \tag{3.6}$$

where f^* is a function from $\mathbb{U}^n$ to $\mathbb{R}$. If f^* is not affine, then z cannot be expressed exactly as an affine combination of the noise symbols ε_i. In that case, we must choose some affine function of the ε_i,

$$f^{\mathrm{a}}(\varepsilon_1, .. \, \varepsilon_n) = z_0 + z_1\varepsilon_1 + \cdots + z_n\varepsilon_n \tag{3.7}$$

that approximates $f^*(\varepsilon_1, .. \, \varepsilon_n)$ reasonably well over its domain $\mathbb{U}^n$, and then add to it an extra term $z_k\varepsilon_k$ to represent the error introduced by this approximation. That is, we return

$$\begin{aligned} \hat{z} &= f^{\mathrm{a}}(\varepsilon_1, .. \, \varepsilon_n) + z_k\varepsilon_k \\ &= z_0 + z_1\varepsilon_1 + \cdots + z_n\varepsilon_n + z_k\varepsilon_k. \end{aligned} \tag{3.8}$$

The term $z_k\varepsilon_k$ will represent the *approximation error* $f^{\mathrm{e}}(\varepsilon_1, .. \, \varepsilon_n) = f^*(\varepsilon_1, .. \, \varepsilon_n) - f^{\mathrm{a}}(\varepsilon_1, .. \, \varepsilon_n)$. As in section 3.3.2, the noise symbol ε_k must be distinct from all other noise symbols that already appeared in the same computation, and the coefficient z_k must be an upper bound on the absolute magnitude of the approximation error f^{e}; that is,

$$|z_k| \geq \max \{ \, |f^*(\varepsilon_1, .. \, \varepsilon_n) - f^{\mathrm{a}}(\varepsilon_1, .. \, \varepsilon_n)| \, : \, \varepsilon_1, .. \, \varepsilon_n \in \mathbb{U} \, \}. \tag{3.9}$$

The residual coefficient z_k must also take into account any roundoff errors incurred in the computation of the other coefficients $z_0, .. \, z_n$.

Note that the substitution of $z_k\varepsilon_k$ for $f^{\mathrm{e}}(\varepsilon_1, .. \varepsilon_n)$ in formula (3.8) entails a loss of information: the noise symbol ε_k is actually a function of $\varepsilon_1, .. \varepsilon_n$, but this constraint is not recorded, so subsequent computations will assume that ε_k is an independent source of variation. Therefore, we can take $|z_k|$ as a measure of this information loss, i.e., the approximation error of the operation $\hat{z} \leftarrow \hat{f}(\hat{x}, \hat{y})$.

3.3.4. Example: multiplication

To illustrate this general principle, let's consider the multiplication of two affine forms, $\hat{z} \leftarrow \hat{x}\hat{y}$, where $\hat{x} = 30 - 4\varepsilon_1 + 2\varepsilon_2$ and $\hat{y} = 20 + 3\varepsilon_1 + 1\varepsilon_3$. Note that the operands are partially correlated through the shared noise symbol ε_1. We can collect the terms of the product $\hat{x}\hat{y}$, into an affine part A and a pure quadratic residue Q, where

$$\begin{aligned} A(\varepsilon_1, .. \varepsilon_n) &= 600 + 10\varepsilon_1 + 40\varepsilon_2 + 30\varepsilon_3 \\ Q(\varepsilon_1, .. \varepsilon_n) &= (-4\varepsilon_1 + 2\varepsilon_2)(3\varepsilon_1 + 1\varepsilon_3) \end{aligned}$$

The two factors of Q, considered independently, have ranges $[-6 \_\_ +6]$ and $[-4 \_\_ +4]$; therefore, a quick estimate for the range of Q is $\bar{Q} = [-24 \_\_ +24]$, i.e. $0 + 24\varepsilon_4$ in affine format. Using this estimate, we can choose for z the affine form

$$\hat{z} = 600 + 10\varepsilon_1 + 40\varepsilon_2 + 30\varepsilon_3 + 24\varepsilon_4.$$

which implies that the range of z is contained in $600 \pm 104 = [496 \_\_ 704]$.

A detailed analisis shows that the actual range of $\hat{x} \cdot \hat{y}$ is $[528 \_\_ 675]$; therefore, the range implied by AA is only $(704 - 496)/(675 - 528) = 1.42$ times wider than the exact range. For comparison, computing the product $z \leftarrow xy$ with standard IA would yield the interval $\bar{z} = [24 \_\_ 36] \cdot [16 \_\_ 24] = [384 \_\_ 864]$, which is $(864 - 384)/(675 - 528) = 3.26$ times wider than the actual range. The large discrepancy is due to the negative correlation between x and y, implied by the shared symbol ε_1. The correlated terms $-80\varepsilon_1$ and $+90\varepsilon_1$ nearly cancel out in the AA computation, but are added with the same sign in the IA computation.

Note, moreover, that the affine form $\hat{z}$ also traces most of the uncertainty in the AA result to uncertainty in the original data, represented by the noise variables ε_1, ε_2, and ε_3; the approximation error (loss of information) introduced by the AA operation itself is only the residual term $24\varepsilon_4$. In contrast, the IA result $[384 \_\_ 864] = 624 \pm 240$ does not specify the source of its uncertainty, and therefore all of it must be viewed as approximation error of that operation.

3.4. Selecting the affine approximation

There are $n + 1$ degrees of freedom in the choice of the affine approximation f^{a} in equation (3.10). The AA model leaves the choice to the implementor, provided that the approximation error $|z_k|$ in equation (3.8) is asymptotically as good as one could hope for. In particular, if f is twice differentiable, the error term must depend quadratically on the radius of the joint range of the affine forms $\hat{x}, \hat{y}, \ldots$.

In the interest of simplicity and efficiency, we consider only *normal* approximations f^{a}, that are themselves affine combinations of the input forms $\hat{x}$ and $\hat{y}$; i. e.,

$$f^{\mathrm{a}}(\varepsilon_1, .. \varepsilon_n) = \alpha\hat{x} + \beta\hat{y} + \zeta. \tag{3.10}$$

Thus, we have only three parameters to determine, instead of $n + 1$. In particular, for a unary operation $z \leftarrow f(x)$, a normal approximation is a straight line on the x–z plane. After the operation, the joint range $\langle \hat{x}, \hat{z} \rangle$ of the affine forms $\hat{x}, \hat{z}$ will be a parallelogram with vertical sides of length $2\,|z_k|$, bisected by that line. The constraint on $|z_k|$ (equation (3.9)) ensures that the range $\langle \hat{x}, \hat{z} \rangle$ encloses the graph of $f(x)$, restricted to the interval $\bar{x}$. See figure 3 (left).

Similarly, for a binary operation $z \leftarrow f(x, y)$, a normal approximation f^{a} defines a plane $z = \alpha x + \beta y + \zeta$ in the (x, y, z) space. The joint range $\langle \hat{x}, \hat{y}, \hat{z} \rangle$ will then be a prism with vertical walls and oblique bases, parallel to and equidistant from this plane. This prism has vertical extent $2\,|z_k|$, and its projection on the x–y plane is the polygon $\langle \hat{x}, \hat{y} \rangle$. Condition (3.9) ensures that the prism encloses the graph of $f(x, y)$, restricted to $\langle \hat{x}, \hat{y} \rangle$. See Figure 3 (right).

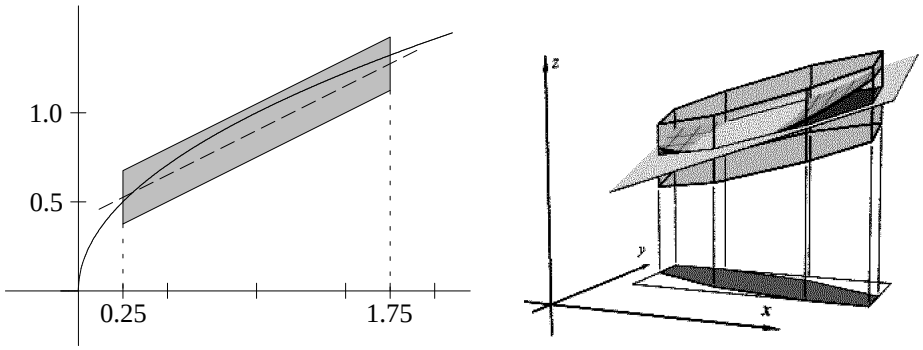

Figure 3: **Left:** Normal approximation of the function $z = \sqrt{x}$ (solid curve), given $\hat{x} = 1 + 0.75\varepsilon_1$, by the affine function $f^{\mathrm{a}}(x) = 0.5x + 0.4 \pm 0.15$ (dashed line), yielding $\hat{z} = 0.5 + 0.4\varepsilon_1 + 0.15\varepsilon_2$. **Right:** Normal approximation of the function $z = 10 + (x^2 + y^2)/80$ (curved surface), given the affine forms $\hat{x} = 15 - 4\varepsilon_1 + 3\varepsilon_3 + 5\varepsilon_4$, $\hat{y} = 15 + 2\varepsilon_1 + 2\varepsilon_2 - \varepsilon_4$, by the affine function $f^{\mathrm{a}}(z) = 0.3281x + 0.3750y + 5.5469 \pm 3.2815$ (plane), yielding $\hat{z} = 16.0938 - 0.5625\varepsilon_1 + 0.7500\varepsilon_2 + 0.9844\varepsilon_3 + 1.2656\varepsilon_4 + 3.2815\varepsilon_5$.

3.4.1. Chebyshev approximations

Among normal approximants, the best choice for f^{a} — in the sense of minimizing the error term $|z_k|$ — is the one which minimizes the maximum absolute error $|\alpha x + \beta y + \zeta - f(x, y)|$, when x and y range over the polygon $\langle \hat{x}, \hat{y} \rangle$. This is called the *Chebyshev* (or *minimax*) *affine approximation* of f over $\langle \hat{x}, \hat{y} \rangle$.

In particular, for a unary operation $z \leftarrow f(x)$, the Chebyshev criterion ensures that the joint range $\langle \hat{x}, \hat{z} \rangle$ of the input and output forms will be the *minimum-area* parallelogram with vertical sides that encloses the graph of f over the interval $\bar{x}$

implied by $\hat{x}$. See figure 4 (left). For a binary operation $z \leftarrow f(x, y)$, the Chebyshev criterion implies that the range $\langle \hat{x}, \hat{y}, \hat{z} \rangle$ is the *minimum-volume* prism, with vertical sides and parallel oblique bases, that encloses the surface $z = f(x, y)$ when (x, y) is restricted to the polygon $\langle \hat{x}, \hat{y} \rangle$. See figure 4 (right).

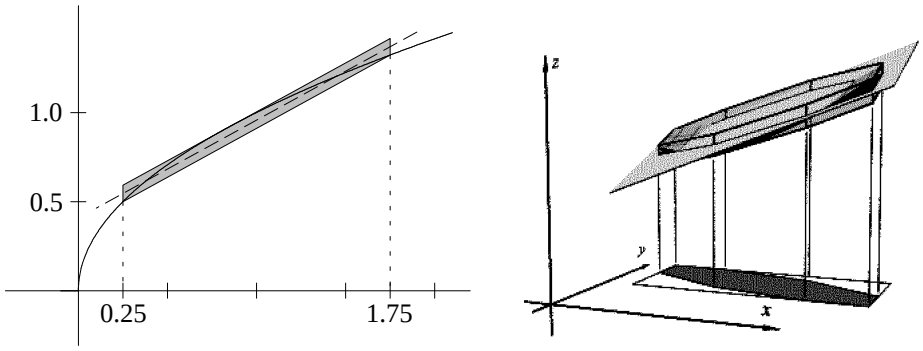

Figure 4: **Left:** Chebyshev affine approximation of $z = \sqrt{x}$, given the affine form $\hat{x} = 1 + 0.75\,\varepsilon_1$, by the form $f^{\mathrm{a}}(z) = 0.5486x + 0.4093 \pm 0.0466$ yielding $\hat{z} = 0.9579 + 0.41145\varepsilon_1 + 0.0466\varepsilon_2$. **Right:** Chebyshev affine approximation of $z = 10 + (x^2 + y^2)/80$, given $\hat{x} = 15 - 4\varepsilon_1 + 3\varepsilon_3 + 5\varepsilon_4$, $\hat{y} = 15 + 2\varepsilon_1 + 2\varepsilon_2 - \varepsilon_4$, by the affine function $f^{\mathrm{a}}(z) = 0.375x + 0.385y + 5.432 \pm 1.058$ (plane), yielding $\hat{z} = 16.6812 - 0.7500\varepsilon_1 + 0.7500\varepsilon_2 + 1.1250\varepsilon_3 + 1.5000\varepsilon_4 + 1.0581\varepsilon_5$.

3.5. Implementing affine arithmetic

In some applications, roundoff errors are known to be negligible compared to other sources of uncertainty, such as argument variation and truncation errors. In such cases, affine arithmetic is fairly easy to implement; see for example the C++ library published by O. Gay [13]. Indeed, it appears that many researchers have created similar AA libraries for their own use. (See the bibliography). For a version that takes roundoff errors into account, see the C library made available by J. Stolfi [46].

3.6. Using affine arithmetic

Many applications of IA use some sort of adaptive domain subdivision, where an initial argument interval is recursively partitioned into smaller intervals, until the result intervals satisfy some criterion such as small size or definite sign [10]. One can substitute AA for IA in each sub-interval, simply by converting the input intervals to affine forms, and reducing the computed AA form to an interval. Since AA yields aymptotically tighter ranges than IA in many cases, the replacement is worth considering — at least, in situations where the input intervals are small enough for the asymptotic analysis to become relevant. See figure 5, which should be compared to figure 1. Specifically, if the IA computation has n arguments and overestimates the result's range by a factor σ, the use of AA may reduce the number of function evaluations by a factor $1/\sigma^n$. Since σ is often much larger than 1, this gain can easily offset the higher cost of AA operations [10].

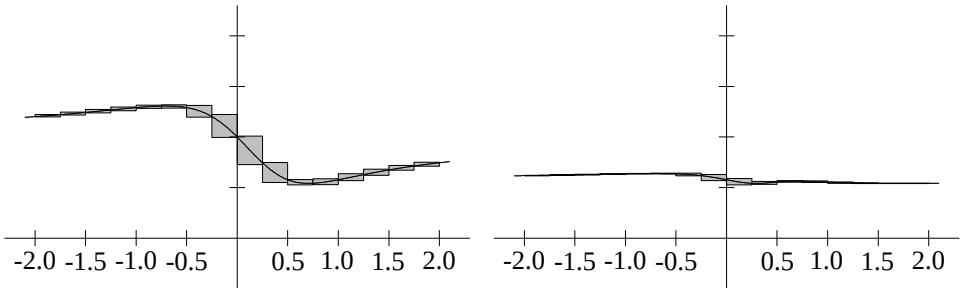

Figure 5: Avoiding error explosion in iterated functions with affine arithmetic. The plots show the result of evaluating $y = g(x) = \sqrt{x^2 - x + 1/2}/\sqrt{x^2 + 1/2}$ (left) and its iterate $y = h(x) = g(g(x))$ (right), over 16 equal intervals $\bar{x}$ of with 0.25 in $[-2 \_\_ +2]$. Each sub-interval was converted to an affine form, the function was evaluated with AA, and the result $\hat{y}$ was converted to an ordinaty interval $\bar{y}$ for plotting.

However, this naive use of AA — as a mere plug-in replacement for IA — fails to exploit its main advantage, namely that it yields first-order approximations to the function, with quadratic approximation errors. To take advantage of this feature, the application must be modified to use enclosures based on affine forms (zonotopes) instead of intervals (axis-aligned boxes). Then, one should be able to meet a specified tolerance δ with $O(1/\delta^{n/2})$ function evaluations, instead of $O(1/\delta^n)$ as required by IA [9]. See figure 6.

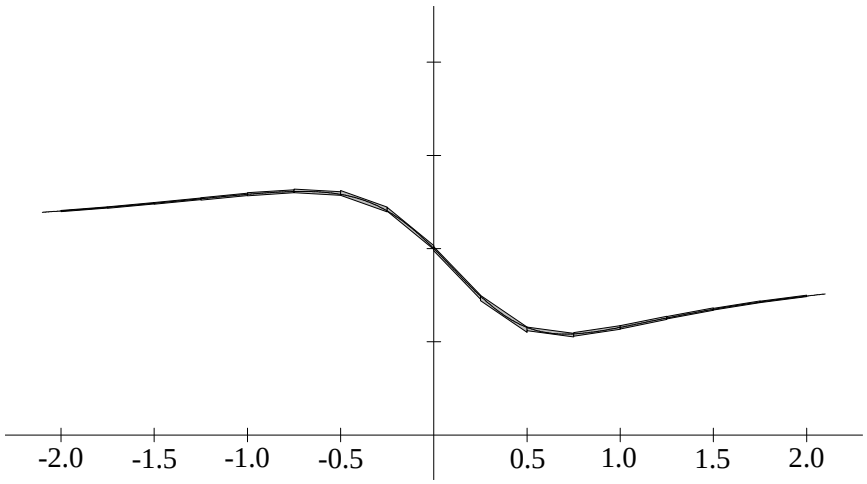

Figure 6: Effective use of affine arithmetic for function modeling. The plot shows the same function g of figures 1 (left) and 5 (left), evaluated ofver the same intervals. Here each parallelogram is the joint range $\langle \hat{x}, \hat{y} \rangle$ of an input sub-interval $\bar{x}$, converted to an affine form $\hat{x}$, and the the corresponding AA result $\hat{y} \leftarrow g^{\mathrm{a}}(\hat{x})$.

4. Conclusions

It goes without saying that the elementary operations of AA are substantially more expensive than those of standard IA. The internal representation of a quantity that depends on n input variables requires $\Omega(n)$ storage, and its computation requires $\Theta(n)$ instructions. However, the tighter range estimates provided by AA often lead to fewer function evaluations, so the total running time is actually reduced; and this advantage increases as the global error tolerance gets reduced. Indeed, AA has been found to be more efficient than IA in many applications [10, 22, 34, 33, 3, 25, 32, 50, 6, 7, 12, 15, 9].

There are other SV computation models that provide first-order approximations, including E. R. Hansen's *generalized interval arithmetic* [20] and its *centered form* variant [41], *first-order Taylor arithmetic* [41], and the *ellipsoidal calculus* of Chernousko, Kurzhanski, and Ovseevich [4, 31]. Although a systematic comparative analysis of those models is still lacking, affine arithmetic seems to have several advantages over them, including a wider range of applications and a more convenient programming interface.

Acknowledgements

We acknowledge the contribution of Joo L. D. Comba and Marcus V. A. Andrade in the initial development of affine arithmetic. Our research has benefited from work and comments by many people, including especially R. V. Iwaarden, A. Cusatis Jr., L. Velho, G. F. Corliss, and R. Chencarek.

Resumo. A aritmética afim (AA) é um modelo para computação auto-validada, que, assim como a aritmética intervalar (AI) padrão, produz envoltórias garantidas para os valores calculados, levando em conta tanto eventuais incertezas nos dados de entrada quanto os erros internos de truncamento e arredondamento. Ao contrário da AI padrão, as representações de grandezas usadas na AA são aproximações de primeira ordem, cujo erro tem geralmente dependência quadrática na largura dos intervalos de entrada. Em muitas aplicações, a maior precisão assintótica da aritmética afim mais do que compensa o custo maior de suas operações.

References

[1] R. Baker Kearfott, Algorithm 763: INTERVAL_ARITHMETIC — A Fortran 90 module for an interval data type, *ACM Transactions on Mathematical Software*, **22**, No. 4 (1996), 385–392.

[2] W. Barth, R. Lieger, and M. Schindler. Ray tracing general parametric surfaces using interval arithmetic. *The Visual Computer*, **10**, No. 7 (1994), 363–371.

[3] A. Bowyer, R. Martin, H. Shou and I. Voiculescu, Affine intervals in a CSG geometric modeller, in "Proc. Uncertainty in Geometric Computations", pp. 1–14. Kluwer Academic Publishers, July, 2001.

[4] F.L. Chernousko and A.I. Ovseevich, Method of ellipsoids: Guaranteed estimation in dynamical systems under uncertainties and control, in "Abstracts of the International Conference on Interval and Computer-Algebraic Methods in Science and Engineering (INTERVAL/94)", p. 66, St. Petersburg, Russia, April, 1994.

[5] J.L.D. Comba and J. Stolfi, Affine arithmetic and its applications to computer graphics, in "Anais do VI Simpósio Brasileiro de Computação Gráfica e Processamento de Imagens (SIBGRAPI'93)", pp. 9–18, Recife (Brazil), October, 1993.

[6] A. de Cusatis Jr., L.H. Figueiredo and M. Gattass, Interval methods for ray casting surfaces with affine arithmetic, in "Poceedings of SIBGRAPI'99 - the 12th Brazilian Symposium on Computer Graphics and Image Processing", pp. 65–71, 1999.

[7] L.H. de Figueiredo, Surface intersection using affine arithmetic, in "Proc. Graphics Interface '96", pp. 168–175, May, 1996.

[8] L.H. de Figueiredo and J. Stolfi, Affine arithmetic: Concepts and applications, *Numerical Algorithms*, (2004), to appear.

[9] L.H. de Figueiredo, J. Stolfi and L. Velho, Approximating parametric curves with strip trees using affine arithmetic, *Computer Graphics Forum*, **22**, No. 2 (2003), 171–179.

[10] L.H. de Figueiredo and J. Stolfi, "Self-Validated Numerical Methods and Applications", Brazilian Mathematics Colloquium monographs, IMPA/CNPq, Rio de Janeiro, Brazil, 1997.

[11] T. Duff, Interval arithmetic and recursive subdivision for implicit functions and constructive solid geometry, in "Proc. SIGGRAPH'92", pp. 131–138, July, 1992.

[12] N. Femia and G. Spagnuolo, True worst-case circuit tolerance analysis using genetic algorithm and affine arithmetic - Part I, *IEEE Trans. on Circuits and Systems*, **47**, No. 9 (2000), 1285–1296.

[13] O. Gay, Libaa - C++ affine arithmetic library for GNU/Linux, Available at `http://www.nongnu.org/libaa/`, April, 2003.

[14] M. Gleicher and M. Kass, An interval refinement technique for surface intersection, in "Proc. Graphics Interface '92", pp. 242–249, May, 1992.

[15] S. Goldenstein, C. Vogler and D. Metaxas, "Cue integration using affine arithmetic and gaussians", Technical Report MS-CIS-02-06, University of Pennsylvania, 2002.

[16] L.J. Guibas, L. Ramshaw and J. Stolfi, The kinetic framework for computational geometry. in "Proc. 24th Annual Symposium on Foundations of Computer Science (FOCS)", pp. 100–111, Tucson, Arizona, November, 1983.

[17] E. Hansen, Global optimization using interval analysis — The one-dimensional case, *Journal of Optimization Theory and Applications*, **29**, No. 3 (1979), 331–344.

[18] E. Hansen, Global optimization using interval analysis — The multi-dimensional case, *Numerische Mathematik*, **34**, No. 3 (1980), 247–270.

[19] E. Hansen, "Global Optimization using Interval Analysis", Number 165 in Monographs and Textbooks in Pure and Applied Mathematics, M. Dekker, New York, 1992.

[20] E.R. Hansen, A generalized interval arithmetic, in "Interval Mathematics" (K. Nickel, ed.), Lecture Notes in Computer Science 29, pp. 7–18, Springer, 1975.

[21] J. Hass, M. Hutchings and R. Schlafly, The double bubble conjecture, *Electronic Research Announcements of the American Mathematical Society*, **1** (1995), 98–102.

[22] W. Heidrich, Ph. Slusallek and H.-P. Seidel, Sampling procedural shaders using affine arithmetic, *ACM Transactions on Graphics (TOG)*, **17**, No. 3, (1998), 158–176.

[23] K. Ichida and Y. Fujii, An interval arithmetic method for global optimization, *Computing*, **23** (1979), 85–97.

[24] IEEE standard for binary floating-point arithmetic, ANSI/IEEE Standard 754-1985, 1985.

[25] Y. Kanazawa and S. Oishi, A numerical method of proving the existence of solutions for nonlinear ODEs using affine arithmetic, in "Proc. SCAN'02 – 10th GAMM-IMACS International Symposium on Scientific Computing, Computer Arithmetic, and Validated Numerics", September, 2002.

[26] R.B. Kearfott, An interval branch and bound algorithm for bound constrained optimization problems, *Journal of Global Optimization*, **2** (1992), 259–280.

[27] O. Knüppel and T. Simenec, "PROFIL/BIAS extensions", Technical Report 93.5, Department of Computer Science III, Technical University of Hamburg-Harburg, November, 1993.

[28] O. Knüppel, "BIAS — Basic Interval Arithmetic Subroutines", Technical Report 93.3, Department of Computer Science III, Technical University of Hamburg-Harburg, July, 1993.

[29] O. Knüppel. "PROFIL — Programmer's Runtime Optimized Fast Interval Library", Technical Report 93.4, Department of Computer Science III, Technical University of Hamburg-Harburg, July, 1993.

[30] V. Kreinovich and D.J. Berleant, "Interval computations", WWW document at `http://www.cs.utep.edu/interval-comp/main.html`, 2002.

[31] A.B. Kurzhanski, Ellipsoidal calculus for uncertain dynamics, in "Abstracts of the International Conference on Interval and Computer-Algebraic Methods in Science and Engineering (INTERVAL/94)", p. 162, St. Petersburg, Russia, April, 1994.

[32] A. Lemke, L. Hedrich and E. Barke, Analog circuit sizing based on formal methods using affine arithmetic, in "Proc. ICCAD-2002 - International Conference on Computer Aided Design", pp. 486–489, November, 2002.

[33] F. Messine, Extentions of affine arithmetic: Application to unconstrained global optimization, *Journal of Universal Computer Science*, **8**, No. 11 (2002), 992–1015.

[34] F. Messine and A. Mahfoudi, Use of affine arithmetic in interval optimization algorithms to solve multidimensional scaling problems, in "Proc. SCAN'98 - IMACS/GAMM International Symposium on Scientific Computing, Computer Arithmetic and Validated Numerics", pp. 22–25, Budapest, Hungary, September, 1998, to appear in *Reliable Computing*.

[35] D. Michelucci and J.-M. Moreau, Lazy arithmetic, *IEEE Transactions on Computers*, **46**, No. 9 (1997), 961–975.

[36] D.P. Mitchell, Robust ray intersection with interval arithmetic, in "Proc. Graphics Interface '90", pp. 68–74, May, 1990.

[37] R. Moore, E. Hansen and A. Leclerc, Rigorous methods for global optimization, in "Recent Advances in Global Optimization", (C.A. Floudas and P.M. Pardalos, eds.), pp. 321–342. Princeton University Press, Princeton, NJ, 1992.

[38] R.E. Moore, "Interval Analysis", Prentice-Hall, 1966.

[39] R.E. Moore, "Methods and Applications of Interval Analysis", SIAM, Philadelphia, 1979.

[40] S.P. Mudur and P.A. Koparkar, Interval methods for processing geometric objects, *IEEE Computer Graphics & Applications*, **4**, No. 2 (1984), 7–17.

[41] A. Neumaier, Taylor forms: Use and limits. *Reliable Computing*, **9**, No. 1 (2003), 43–79.

[42] H. Ratschek and R.L.Voller, What can interval analysis do for global optimization? *Journal of Global Optimization*, **1**, No. 2 (1991), 111–130.

[43] H. Ratschek and J. Rokne, "New Computer Methods for Global Optimization", Ellis Horwood Ltd., 1988.

[44] Reliable computing, Kluwer Academic Publishers, 1997, formerly *Interval Computations*.

[45] J.M. Snyder, Interval analysis for computer graphics. in "Proc. SIGGRAPH'92", pp. 121–130, July 1992, special Issue of ACM Computer Graphics.

[46] J. Stolfi, "`LIBAA`: An affine arithmetic library in C", 1993, avaliable at `http://www.dcc.unicamp.br/~stolfi/`.

[47] K.G. Suffern and E.D. Fackerell, Interval methods in computer graphics. *Computers & Graphics*, **15**, No. 3 (1991), 331–340.

[48] G. Taubin, Rasterizing algebraic curves and surfaces, *IEEE Computer Graphics and Applications*, **14** (1994), 14–23.

[49] R. van Iwaarden, "An Improved Unconstrained Global Optimization Algorithm", PhD thesis, Department of Mathematics, University of Colorado at Denver, 1996.

[50] Q. Zhang and R.R. Martin, Polynomial evaluation using affine arithmetic for curve drawing, in "Proc. of Eurographics UK 2000 Conference", pp. 49–56, 2000.