# Affine Arithmetic
(extended abstract)

Marcus Vinicius Alvim Andrade[1]
João Luiz Dihl Comba[2]
Jorge Stolfi[1]

[1]Computer Science Dept.
University of Campinas, Brazil
{stolfi,marcus}@dcc.unicamp.br

[2]Computer Graphics Lab. – COPPE
Federal University of Rio de Janeiro
comba@lcg.ufrj.br

## 1 Summary

We propose here a new model for numerical computation, which we call *affine arithmetic* (AA).

Like standard interval analysis (IA), affine arithmetic keeps track automatically of the round-off and truncation errors affecting each computed quantity. In addition, AA keeps track of *correlations* between those quantities. Thanks to this extra information, affine arithmetic is able to provide much tighter intervals than IA, especially in long computation chains.

As one may expect, the AA model is more complex and expensive than ordinary interval arithmetic. However, we believe that its higher accuracy will be worth the extra cost in many applications.

## 2 Error growth in standard IA

The arithmetic operations of standard IA always assume that the values of the operands may vary *independently* over the given intervals. If this assumption is not valid, the result interval returned by the IA operation may be much wider than the true range of the result quantity.

For example, consider the formula $y = x(10-x)$, where $x$ ranges over $\bar{x} = [4 \mathinner{..} 6]$. Evaluating $y$ with standard IA gives the interval $[4\cdot 4 \mathinner{..} 6\cdot 6] = [16 \mathinner{..} 36]$; whereas the true range of $y$ is $[24 \mathinner{..} 25]$.

Note that the computed interval is 20 times wider than the true range. The the reason for this discrepancy is that the interval multiplication procedure is blind to the inverse relation between the quantities $x$ and $10 - x$.

Clearly, if this phenomenon happens a few times times in a long computation chain, the resulting interval may be millions of times wider than the true range. Moreover, when this "error explosion" happens, it cannot be avoided by subdividing the input intervals, because the relative accuracy of an IA computation is generally independent of the width of the input intervals. (That is, IA has basically first-order approximation error.)

## 3 Affine arithmetic

Affine arithmetic (AA) is a computation model that attempts to retain the advantages of IA while ameliorating the error explosion problem. The key feature of AA is an extended encoding of quantities from which one can determine, in addition to their ranges, also some of their mutual dependencies—such as the relation between $x$ and $10 - x$ in the previous example.

Thanks to this extra information, individual AA operations are usually affected only by *second-order* approximation errors. Moreover, even these errors may cancel out in later steps — unlike standard IA operations, where the errors always accumulate.

### 3.1 Affine forms

Specifically, in AA we represent a partially unknown quantity $x$ by an *affine form* $\hat{x}$, which is a first-degree polynomial

$$\hat{x} = x_0 + x_1\varepsilon_1 + x_2\varepsilon_2 + \cdots + x_n\varepsilon_n \qquad (1)$$

Here the $x_i$ are known real coefficients (stored as floating-point numbers), and the $\varepsilon_i$ are symbolic variables whose values are unknown but assumed to lie in the interval $\mathbf{U} = [-1 \mathinner{..} +1]$. We call $x_0$ the *central value* of the form; the coefficients $x_i$ are the *partial deviations*, and the $\varepsilon_i$ are the *noise symbols*.

Each noise symbol $\varepsilon_i$ stands for an independent source of uncertainty (measurement error, indeterminacy, numerical approximation, etc.) that adds to the total uncertainty of the quantity $x$. The corresponding coefficient $x_i$ gives the magnitude of that contribution.

The key feature of affine arithmetic is that the same noise symbol $\varepsilon_i$ may be shared by two affine forms $\hat{x}, \hat{y}$, indicating some partial dependency between the underlying quantities $x, y$. The magnitude and sign of the dependency is determined by the corresponding coefficients $x_i, y_i$.

For example, suppose two quantities $x, y$ are represented by the affine forms

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

From this data we can tell that $x$ lies in the interval $[6 \mathinner{..} 14]$, and $y$ lies in $[12 \mathinner{..} 28]$. However, since they both include the same noise symbols $\varepsilon_1$ and $\varepsilon_4$, they are not entirely independent of each other. In fact, the range of the point $(x, y)$ is a convex region of the plane that covers only about one half of the rectangle $[6 \mathinner{..} 14] \times [12 \mathinner{..} 28]$.

## 4 Computing with Affine Forms

To evaluate a formula in the AA model, one needs to replace each elementary operation $z \leftarrow f(x, y)$ on real numbers by a corresponding operation $\hat{z} \leftarrow \hat{f}(\hat{x}, \hat{y})$ on affine forms.

If $f$ is an affine operation (such as sum, difference, or multiplication by a real number), then $f(\hat{x}, \hat{y})$ is itself an affine form. Implementing $\hat{f}$ is then trivial (except for the handling of arithmetic roundoff errors, discussed later on).

If $f$ is not affine, the quantity $f(\hat{x}, \hat{y})$ cannot be expressed as an affine function of the noise symbols $\varepsilon_i$. In that case, we replace $f$ by an affine function $f^{\mathrm{a}}$ that approximates $f$ well enough in the region of interest, and then compute the affine form $f^{\mathrm{a}}(\hat{x}, \hat{y})$. To this we must add an extra term $z_k \varepsilon_k$, which accounts for the difference between $f$ and $f^{\mathrm{a}}$. The coefficient $z_k$ must be an upper bound to the approximation error in the region of interest; and the noise symbol $\varepsilon_k$ must be distinct from all other noise symbols appearing in the same computation.

### 4.1 Multiplication

Let's illustrate this approach with the multiplication of affine forms. (Most other non-affine elementary operations can be handled in a similar way.)

For concreteness, let's evaluate the expression $z = (10 + x + r) \cdot (10 - x + s)$ for $x \in [-2 \mathinner{..} +2]$, $r \in [-1 \mathinner{..} +1]$, $s \in [-1 \mathinner{..} +1]$. Converting the ordinary intervals to affine forms, we get

$$\begin{aligned} x &= 0 + 2\varepsilon_1 \\ r &= 0 + 1\varepsilon_2 \\ s &= 0 + 1\varepsilon_3 \\ 10 + x + r &= 10 + 2\varepsilon_1 + 1\varepsilon_2 \\ 10 - x + s &= 10 - 2\varepsilon_1 + 1\varepsilon_3 \end{aligned}$$

therefore

$$z = 100 + 10\varepsilon_2 + 10\varepsilon_3 + (2\varepsilon_1 + 1\varepsilon_2)(-2\varepsilon_1 + 1\varepsilon_3)$$

We approximate this quadratic formula by the affine form consisting of its first three terms. The last term (containing all the quadratic monomials) will then be the approximation error.

Computing the exact maximum of the error term is not trivial. However, a quick estimate can be obtained by observing that each factor may range between $-3$ and $+3$; therefore, their product lies in $[-9 \mathinner{..} +9]$. Using this estimate, we return

$$\hat{z} = 100 + 10\varepsilon_2 + 10\varepsilon_3 + 9\varepsilon_4$$

Notice that the uncertainty associated to the noise symbol $\varepsilon_1$, shared by both operands, canceled out (to first order) in the final result. As a consequence, the range of $z$ implied by $\hat{z}$ is

$$[100 - 29 \mathinner{..} 100 + 29] = [71 \mathinner{..} 129]$$

which is very close to the true range $[71 \mathinner{..} 121]$. By comparison, standard IA would return $[49 \mathinner{..} 169]$, which is more than twice as wide.

In practice, all the coefficient computations above will be carried out in floating-point arithmetic, and hence will be affected by roundoff error. Those errors must be estimated, and added to the error term $z_k$. (This correction applies even to affine operations like sum or difference.)

## 5 Conclusions

There are several theoretical reasons why we expect AA to be more precise than IA, and less prone to error explosion. First, linear correlations among the operands are detected and accounted for. Second, the approximation errors are quadratic, rather than linear. And, finally, those errors are individually traced through the computation, and may cancel out in latter steps.

We have implemented the basic operations of affine arithmetic ($+$, $-$, $\times$, $\div$, $\sqrt{\ }$) in C for the Sun SPARCstation. Numerical experiments with this prototype implementation generally confirm these expectations.

Of course, affine arithmetic is more expensive than IA, but not prohibitively so. By using a sparse representation for the affine forms, the cost of each operation is proportional to the number of non-zero partial deviations appearing in its operands; and the cost of evaluating an expression with $m$ operations grows like $O(m^2)$. We believe that affine arithmetic will be worth this cost, in many fields where IA is presently not usable because of its poor accuracy.