Journal of Universal Computer Science, vol. 8, no. 11 (2002), 992-1015
submitted: 20/10/99, accepted: 15/11/02, appeared: 28/11/02 © J.UCS

Extensions of Affine Arithmetic: Application to Unconstrained Global Optimization

Frédéric Messine[1]
(Université de Pau et des Pays de l'Adour
UFR-Sciences et Techniques, Département d'Informatique
B.P. 1155, 64 013 Pau, France
Frederic.Messine@univ-pau.fr)

Abstract: Global optimization methods in connection with interval arithmetic permit to determine an accurate enclosure of the *global optimum*, and of all the corresponding *optimizers*. One of the main features of these algorithms consists in the construction of an interval function which produces an enclosure of the range of the studied function over a *box* (right parallelepiped).

We use here affine arithmetic in global optimization algorithms, in order to elaborate new inclusion functions. These techniques are implemented and then discussed. Three new affine and quadratic forms are introduced. On some polynomial examples, we show that these new tools often yield more efficient *lower bounds* (and *upper bounds*) compared to several well-known classical inclusion functions. The three new methods, presented in this paper, are integrated into various Branch and Bound algorithms. This leads to improve the convergence of these algorithms by attenuating some negative effects due to the use of interval analysis and standard affine arithmetic.

Key Words: Affine Arithmetic, Interval Arithmetic, Inclusion Functions, Global Optimization, Branch and Bound Algorithm.
Category: G.1.0 - General and G.1.6 - Optimization

1 Introduction

Interval arithmetic was introduced by Moore as a basic tool for the estimation of numerical errors in machine computations. Instead of approximating a real value x with a machine representable number (a floating point number), a pair of machine representable numbers is used in order to define the smallest interval which encloses x. Then, the basic arithmetic for real numbers is extended to define interval arithmetic [15].

Interval analysis connected with Branch and Bound techniques, permits to enclose with a given accuracy the global optimum and as well as all the solutions. The efficiency of these algorithms has already been validated in various domains such as in Chemical Process [11], or in Electromagnetism for the design of a slotless permanent magnet machine [13].

Such methods work with decomposition techniques and then with an exclusion principle. First, the initial search domain is iteratively divided into smaller

[1] This work was supported by the LEEI - Group EM^2 - CNRS-UMR 5828 laboratory. I would like to acknowledge the anonymous referees for their interest and help about this work.

and smaller parts (sub-boxes are generated). Then, a considered sub-box is eliminated when it can be proved that the global optimum cannot occur in it. For instance, when we consider a minimization problem, the exclusion principle occurs if a lower bound computed by interval arithmetic techniques over a sub-box is higher than the current best solution found so far.

As the algorithm proceeds, the current best solution is improved and sub-boxes are divided and discarded. The algorithms stop when the worse bound (over all the remaning sub-boxes) is close to the current best solution found (with a fixed accuracy). The global optimum is then precisely enclosed. For details on interval Branch and Bound algorithms, see [10, 11, 16].

Hence, it is of great importance to know, for a given function f, an enclosure of the *direct image (or range)* $f(X) = \{f(x) : x \in X\}$ over a box $X \subseteq \mathbb{R}^n$. An *inclusion function*, denoted by F, must satisfy this enclosure property for all considered boxes; thus, $f(X) \subseteq F(X)$, for all boxes X.

For a given closed interval $X \subseteq \mathbb{R}$, let us denote its *lower bound* by x^L and its *upper bound* by x^U, and then $X = [x^L, x^U]$. Furthermore, for an inclusion function F, we denote the bounds by $F(X) = [F^L(X), F^U(X)]$.

A *natural extension* into an interval of an expression of a given function f, is obtained by replacing each occurrence of a variable with its corresponding interval, and each standard operation with its corresponding interval operation. Let us denote this interval function by **NE**. Since **NE** is an inclusion function of f, we have $NE(X) \supseteq f(X)$.

Other inclusion function can be elaborated by considering Taylor expansions, see [8, 16]. For example, a truncated first-order Taylor expansion of f around x_0 yields:

$$T_1(x_0, X) = f(x_0) + (X - x_0)G(X), \forall x_0 \in X \tag{1}$$

where $G(X)$ is an enclosure of the gradient of f over the box X, and it can be obtained by interval automatic differentiation or interval formal differentiation, see [10, 11]. Usually, x_0 is fixed to the midpoint of the interval vector (the box) X: $x_0 = (\cdots, \frac{x_i^L + x_i^U}{2}, \cdots)$. In this case, the corresponding inclusion function is denoted by $T_1(X)$.

In 1988, Baumann gave an analytic solution of the following optimization problem [3]:

$$T_B(X) = \max_{x \in X} T_1^L(x, X) \text{ (or } \min_{x \in X} T_1^U(x, X)) \tag{2}$$

This solution only depends on the interval vector X and the enclosure of the gradient of f, $G(X)$.

However, the computations which directly use interval arithmetic lack precision even if the considered box is very small. Also, some techniques were introduced in order to limit these negative effects: on one hand by using Taylor expansions and on the other hand by using affine arithmetic [2, 4, 6, 7] which also permits to deal with some other difficulties.

1.1 Dependency Problem

The *dependency problem* is well-known in interval computations. It is generated by the fact that each occurrence of a same variable, in a function expression, is taken into account independently. Therefore, the enclosure result can be very wide and thus, uninteresting.

For instance, let us consider the elementary function $f(x) = x - x,$ with $x \in X = [0, 1]$. In this example, the natural extension $\mathbf{NE}(X) = X - X$ produces the enclosure result $[-1, 1]$, although the interval $[0, 0]$ would have been expected.

1.2 Clustering Problem

The *clustering problem* is further described. At the end of an interval Branch and Bound algorithm, when the remaining sub-boxes are very small, it is still impossible to determine with accuracy where the global optimizers are. This is due to the fact that the inclusion function used is not efficient enough in order to produce enclosures close to the range of the considered function (even if the remaining sub-boxes are very small) [5].

Definition 1. The α-*convergence* of an inclusion function (or of other methods which only compute lower bounds) of f over a box X, denoted by $F^L(X)$, is defined by:

$$f^* - F^L(X) \leq k(w(X))^\alpha \tag{3}$$

where f^* is the global minimum, k and α are constants, and $w(X) = \max\limits_{i=\{1,\cdots,n\}} x_i^U - x_i^L$ is the width of the interval vector $X = (\cdots, [x_i^L, x_i^U], \cdots)$.

Generally, the following definition is used: $w(F(X)) \leq k(w(X))^\alpha$.

Du and Kearfott proved in 1992 that in the unconstrained case, without the use of the monotonicity test [8, 16], the clustering problem (if there exists) is either:

- strongly reduced, if an inclusion function with an $\alpha-$convergence at the order 2 is used,
- or eliminated, if an inclusion function with an $\alpha-$convergence at the order 3 or more is used.

But unfortunately, no inclusion function with an $\alpha-$convergence at the order 3 or more, is known.

E. Hansen proved that the centered forms using Taylor expansions at the first order and at the second order, have an $\alpha-$convergence at the order 2 [8].

Remark. Generally, a natural extension into an interval of an expression of a given function, produces an $\alpha-$convergence at the order 1.

1.3 Used Global Optimization Algorithm

Some new efficient inclusion functions based on affine arithmetic are presented in this paper. In order to prove that the *dependency problem* and the *clustering problem* are strongly reduced, we choose to consider a classical Branch and Bound algorithm due to Ichida-Fujii [9] (*cut off test* and *middle point test*). This algorithm can be used to solve global unconstrained minimization problems which can be formulated as follow:

$$\min_{x \in D} f(x), \text{ with } D \subseteq \mathbb{R}^n \tag{4}$$

Few assumptions about f are needed, see [8, 10, 15, 16]. In this paper, only polynomial functions will be considered.

Definition 2. 1. *Cut off Test*:

If $\overline{f} < F^L(X)$ then the search of the global minimum in the sub-box X can be stopped (where $\overline{f}$ represents the current best solution, and $F^L(X)$ denotes a lower bound of f over the box X).

2. *Middle Point Test*:

$$\overline{f} \longleftarrow \min\{\overline{f}, f(mid(X))\}$$

At each step of the Branch and Bound algorithm, the function f is evaluated at the center of a considered sub-box X and then, we can improve the best current solution $\overline{f}$ (which overestimates the minimum).

After a short introduction of standard affine arithmetic due to Andrade, Stolfi, Comba, de Figueiredo and Van Iwaarden, [2, 4, 6, 7], two new affine forms and one quadratic form, will be presented here. Then, we will observe how these new forms will permit to deal with the difficulties generated by the use of interval analysis and also standard affine arithmetic.

As a conclusion regarding the efficiency of these new methods, we can note that numerical results on polynomial examples will validate the interest of the use of these three new affine and quadratic forms. Comparisons will be performed by considering other standard inclusion functions **NE**, $\mathbf{T}_1$, $\mathbf{T}_B$ [1, 3, 15, 16], standard affine arithmetic [7], and a method based on the computation of affine hyperplanes intersections, denoted by **AS** [14].

2 Standard Affine Arithmetic

Affine Arithmetic is a new approach which makes it possible to compute an enclosure of the range of a given function f over a box X. It was introduced in 1994 by Andrade, Comba and Stolfi [2] and was first applied in computer graphic problems [4] and surface intersections [6]. Recently, these methods were introduced in unconstrained interval Branch and Bound algorithms [7].

Like generalized interval arithmetic introduced by Hansen [8], these techniques are developed to take into account the *dependency problem* (due to the use of interval arithmetic), see Subsection 1.1.

2.1 Notation

Affine forms are denoted by **AF**, and are represented by:

$$\widehat{x} = x_0 + \sum_{i=1}^{n} x_i \varepsilon_i. \tag{5}$$

Here, x_i are real coefficients fixed (stored floating-point numbers), and ε_i are symbolic variables, the values of which are unknown, but lie in the interval $[-1, 1]$.

Each ε_i derives from independent source of approximation, error or uncertainty (original uncertainty or round-off and truncation errors performed during the computation of $\widehat{x}$).

2.2 Conversions

- Interval $\longrightarrow$ Affine Form:

$$\begin{aligned} &X = [x^L, x^U] \\ &\longrightarrow \widehat{x} = \frac{x^L + x^U}{2} + \frac{x^L - x^U}{2}\varepsilon_k. \end{aligned} \tag{6}$$

  where ε_k symbolizes the uncertainty of the value of x. k is the new index of the new symbolic variable so generated (after each conversion, one is added to k).

- Affine Form $\longrightarrow$ Interval:

$$\begin{aligned} &\widehat{x} = x_0 + \sum_{i=1}^{n} x_i \varepsilon_i \\ &\longrightarrow X = x_0 + \left(\sum_{i=1}^{n} |x_i| \right) \times [-1, 1] \end{aligned} \tag{7}$$

As in interval arithmetic, all standard operators $(+, -, \times, \div)$ and other functions $(x^2, \sqrt{x}, |x|)$ are redefined for affine forms [2].

2.3 Classical Operations

$$
\begin{aligned}
\widehat{x} \pm \widehat{y} &= (x_0 \pm y_0) + \sum_{i=1}^{n} (x_i \pm y_i)\varepsilon_i, \\
a \pm \widehat{x} &= (a \pm x_0) \pm \sum_{i=1}^{n} x_i \varepsilon_i, \\
a \times \widehat{x} &= a x_0 + \sum_{i=1}^{n} a x_i \varepsilon_i,
\end{aligned}
\tag{8}
$$

where $\widehat{x}$ and $\widehat{y}$ are affine forms and a is a real number.

Multiplication (non-affine operation):

It is clear that the multiplication, the division, the square, and the square root are neither affine operations nor affine functions. Therefore, the results have not an affine form, and thus an affine approximation must be introduced. Another extra term $z_k \varepsilon_k$ must be appended to represent the error due to the affine approximation (in the Chebyshev's sense to minimize the maximum error) [2, 4].

$$
\begin{aligned}
\widehat{x} \times \widehat{y} &= \left(x_0 + \sum_{i=1}^{n} x_i \varepsilon_i \right) \times \left(y_0 + \sum_{i=1}^{n} y_i \varepsilon_i \right) \\
&= x_0 y_0 + \sum_{i=1}^{n} (x_0 y_i + x_i y_0)\varepsilon_i + \left(\sum_{i=1}^{n} x_i \varepsilon_i \right) \left(\sum_{i=1}^{n} y_i \varepsilon_i \right).
\end{aligned}
$$

The *best affine approximation* which conserves the enclosure property is:

$$
\widehat{x} \times \widehat{y} = x_0 y_0 + \sum_{i=1}^{n} (x_0 y_i + x_i y_0)\varepsilon_i + \left(\sum_{i=1}^{n} |x_i| \times \sum_{i=1}^{n} |y_i| \right) \varepsilon_{n+1}. \tag{9}
$$

A new noise symbolic variable ε_{n+1} $(k = n + 1)$ is introduced with its corresponding real coefficient given by $\sum_{i=1}^{n} |x_i| \times \sum_{i=1}^{n} |y_i|$, which is an upper bound due to the affine approximation.

2.4 Round-off Errors

As in rounded interval arithmetic [1, 15], the purpose is to discard the numerical errors due to the floating arithmetic operations, caused by the floating coefficients in the affine forms.

Thus, the main idea is to add some other terms $x_k \varepsilon_k$ which are generated after each affine operation [2]. The positive value x_k depends upon the other coefficients x_i. In this case, the conversion into interval must be performed with the *rounded interval arithmetic* [15]. Let us consider an affine form $\widehat{x} = x_0 + \sum_{i=1}^{n} x_i \varepsilon_i$, then the corresponding interval becomes:

$$x_0 + \sum_{i=1}^{n} [-|x_i|, |x_i|], \text{ using rounded interval arithmetic.}$$

Nevertheless, performing these affine computations, a numerical error may occur. These numerical difficulties are just decreased and then not avoided.

Remark. Another idea to construct "rigorous" affine forms, is to convert them into interval affine forms such that the real coefficients x_i are replaced with the interval $X_i = [\underline{x_i}, \overline{x_i}]$ where $\underline{x_i}$ and $\overline{x_i}$ are the floating lower and upper numbers the closest to the real coefficient x_i. Therefore all the computations are performed by the use of rounded interval analysis. In this way, a *reliable affine arithmetic* can be elaborated.

In order to simplify the last part of this paper, one should not take into account numerical problems. The new representations which are presented below, could be easily extended to deal with the numerical difficulties.

2.5 Difficulties due to Non-affine Operations ($\times$)

1. Affine forms grow by performing non-affine operations:

   p non-affine operations generate p new symbolic variables $\varepsilon_k, k = n+1, \cdots, n+p$.

   This produces some technical implementation difficulties because the number of non-affine operations is rarely known and then, this number must be arbitrarily fixed.

2. Square (or even power) of affine form:

   Consider the square function x^2 with $x \in X = [-2, 2]$

   - Interval Arithmetic gives $X^2 = [0, 4] \neq X \times X = [-4, 4]$.
   - Affine Arithmetic gives $\widehat{x} = 2\varepsilon_1$ and $\widehat{x}^2 = \widehat{x} \times \widehat{x} = 4\varepsilon_2$.

     The new symbolic variable ε_2 is introduced to produce an affine approximation of the multiplication between two affine forms.

     The conversion into interval gives $[-4, 4]$.

In order to decrease these difficulties, new forms and their corresponding computations are introduced here. One of the purposes of this paper, is to show that many representations based on affine forms could be analogously generated.

3 New Affine Forms

The following two new affine forms are introduced to decrease the difficulties due to the use of standard affine arithmetic. The first form allows to discard the first difficulty (i.e. the fact that affine forms grow by performing non-affine operations), whereas the second form which is an extension of the first new affine form, permits to decrease significantly the second difficulty (i.e. the fact that an affine approximation of the even power is not accurate).

3.1 First Affine Form AF1

AF1 is introduced to avoid the first difficulty which is the fact that affine forms grow by performing non-affine operations, refer to Section 2.5.

Remark. Affine representations were used to retain the knowledge of the variables x_i by using the introduction of corresponding symbolic variables ε_i. Thus, affine information is conserved during the computations. However, all the other symbolic variables $\varepsilon_k, k \in \{n+1, \cdots, n+p\}$ which are generated by performing p non-affine operations, are completely independent. Therefore, all these symbolic variables $\varepsilon_k, k \in \{n+1, \cdots, n+p\}$ are simply stored and then all these terms cannot be added or subtracted.

Hence, without losing affine information, the absolute value of all these p terms can be added in the same single new symbolic variable, denoted ε_{n+1}.

3.1.1 Representation of AF1

$$\widehat{x} = x_0 + \sum_{i=1}^{n} x_i \varepsilon_i + x_{n+1}\varepsilon_{n+1}, \tag{10}$$

where ε_{n+1} is the new unique symbolic variable which represents all the errors (due to affine approximations after performing non-affine operations). We can remark that x_{n+1} is a positive number.

3.1.2 Operations with AF1

$$\begin{aligned}
\widehat{x} \pm \widehat{y} &= (x_0 \pm y_0) + \sum_{i=1}^{n} (x_i \pm y_i)\varepsilon_i + (x_{n+1} + y_{n+1})\varepsilon_{n+1}, \\
a \pm \widehat{x} &= (a \pm x_0) \pm \sum_{i=1}^{n} x_i \varepsilon_i + |x_{n+1}|\varepsilon_{n+1}, \\
a \times \widehat{x} &= (a x_0) + \sum_{i=1}^{n} a x_i \varepsilon_i + |a| x_{n+1}\varepsilon_{n+1},
\end{aligned} \tag{11}$$

$$\widehat{x} \times \widehat{y} = x_0 y_0 + \sum_{i=1}^{n}(x_0 y_i + x_i y_0)\varepsilon_i + \left(|x_0| y_{n+1} + |y_0| x_{n+1} + \sum_{i=1}^{n+1}|x_i| \times \sum_{i=1}^{n+1}|y_i|\right)\varepsilon_{n+1}.$$

Remark. The conversions between interval and **AF1** are the same than with **AF**, refer to (6), (7).

3.2 Second Affine Form AF2

A logical extension of **AF1** consists in the use of two new symbolic variables $\varepsilon_{n+2} \in [0,1]$ and $\varepsilon_{n+3} \in [-1,0]$. This way, the second difficulty (even power) can be attenuated, refer to Section 2.5.

3.2.1 Representation of AF2

$$\widehat{x} = x_0 + \sum_{i=1}^{n} x_i \varepsilon_i + x_{n+1}\varepsilon_{n+1} + x_{n+2}\varepsilon_{n+2} + x_{n+3}\varepsilon_{n+3}, \qquad (12)$$

where $\varepsilon_{n+1}, \varepsilon_{n+2}, \varepsilon_{n+3}$ are new symbolic variables and $x_{n+1}, x_{n+2}, x_{n+3}$ must always be positive numbers.

3.2.2 Operations with AF2

$$\begin{aligned}
\widehat{x} + \widehat{y} &= (x_0 + y_0) + \sum_{i=1}^{n}(x_i + y_i)\varepsilon_i + (x_{n+1} + y_{n+1})\varepsilon_{n+1} + (x_{n+2} + y_{n+2})\varepsilon_{n+2} \\
&\quad +(x_{n+3} + y_{n+3})\varepsilon_{n+3}, \\
\widehat{x} - \widehat{y} &= (x_0 - y_0) + \sum_{i=1}^{n}(x_i - y_i)\varepsilon_i + (x_{n+1} + y_{n+1})\varepsilon_{n+1} + (x_{n+2} + y_{n+3})\varepsilon_{n+2} \\
&\quad +(x_{n+3} + y_{n+2})\varepsilon_{n+3}, \\
a + \widehat{x} &= (a + x_0) + \sum_{i=1}^{n} x_i\varepsilon_i + x_{n+1}\varepsilon_{n+1} + x_{n+2}\varepsilon_{n+2} + x_{n+3}\varepsilon_{n+3}, \qquad (13) \\
a - \widehat{x} &= (a - x_0) - \sum_{i=1}^{n} x_i\varepsilon_i + x_{n+1}\varepsilon_{n+1} + x_{n+3}\varepsilon_{n+2} + x_{n+2}\varepsilon_{n+3}, \\
a \times \widehat{x} &= \begin{cases} ax_0 + \sum_{i=1}^{n} ax_i\varepsilon_i + ax_{n+1}\varepsilon_{n+1} + ax_{n+2}\varepsilon_{n+2} + ax_{n+3}\varepsilon_{n+3} \text{ if } a > 0, \\ ax_0 + \sum_{i=1}^{n} ax_i\varepsilon_i + |a|x_{n+1}\varepsilon_{n+1} + |a|x_{n+3}\varepsilon_{n+2} + |a|x_{n+2}\varepsilon_{n+3} \text{ else.} \end{cases}
\end{aligned}$$

The multiplication is somewhat more difficult than for **AF1**:

$$\widehat{x} \times \widehat{y} = x_0 y_0 + \sum_{i=1}^{n}(x_0 y_i + x_i y_0)\varepsilon_i + K_1\varepsilon_{n+1} + K_2\varepsilon_{n+2} + K_3\varepsilon_{n+3}. \qquad (14)$$

Computation of the three positive values K_1, K_2, K_3

– Initialization:

$$K_1 := |x_0|y_{n+1} + |y_0|x_{n+1}$$
$$K_2 := \begin{cases} x_0y_{n+2} + y_0x_{n+2} \text{ if } x_0 > 0 \text{ and } y_0 > 0 \\ x_0y_{n+2} - y_0x_{n+3} \text{ if } x_0 > 0 \text{ and } y_0 < 0 \\ -x_0y_{n+3} + y_0x_{n+2} \text{ if } x_0 < 0 \text{ and } y_0 > 0 \\ -x_0y_{n+3} - y_0x_{n+3} \text{ if } x_0 < 0 \text{ and } y_0 < 0 \end{cases}$$
$$K_3 := \begin{cases} x_0y_{n+3} + y_0x_{n+3} \text{ if } x_0 > 0 \text{ and } y_0 > 0 \\ x_0y_{n+3} - y_0x_{n+2} \text{ if } x_0 > 0 \text{ and } y_0 < 0 \\ -x_0y_{n+2} + y_0x_{n+3} \text{ if } x_0 < 0 \text{ and } y_0 > 0 \\ -x_0y_{n+2} - y_0x_{n+2} \text{ if } x_0 < 0 \text{ and } y_0 < 0 \end{cases}$$

– Iterations:

$$K_1 := K_1 + \sum_{i=1}^{n+3} \sum_{j=1, j \neq i}^{n+3} |x_i y_j|,$$
$$K_2 := K_2 + \sum_{i=1, x_i y_i > 0}^{n+3} x_i y_i,$$
$$K_3 := K_3 + \sum_{i=1, x_i y_i < 0}^{n+3} |x_i y_i|.$$

Remark. One must be careful because sometimes terms are permuted, the subtraction is an example: between ε_{n+2} and ε_{n+3}. Conversions between interval and the affine form **AF2** are similar to conversions between interval and the standard affine form. Nevertheless, ε_{n+2} is replaced with the interval $[0, 1]$ and ε_{n+3} with $[-1, 0]$.

3.3 Example

Let us consider the univariate function $f(x) = x(10 - x)$ for $x \in [4, 6]$, the following enclosure results are obtained:

– with the *natural extension* into interval $\mathbf{NE}([4, 6]) = [4, 6](10 - [4, 6]) = [16, 36]$,

– with *another expression* $f(x) = 10x - x^2$, $\mathbf{NE}_2([4, 6]) = 10 \times [4, 6] - [4, 6]^2 = [4, 44]$,

– with *the standard affine form* **AF** or the *first new affine form* **AF1**:

$$\begin{aligned}\widehat{x} &= 5 + \varepsilon_1 \\ \widehat{x}(10 - \widehat{x}) &= (5 + \varepsilon_1)(5 - \varepsilon_1) \\ &= 25 + 0\varepsilon_1 + \varepsilon_2.\end{aligned}$$

The enclosure corresponding to the use of the affine representation **AF1**, is equal to $[24, 26]$. Observe that one new term ε_2 is generated. The obtained value is very close to the exact range of f over $[4, 6]$ which is $f([4, 6]) = [24, 25]$.

– with **AF2**:

$$\begin{aligned}\widehat{x} &= 5 + \varepsilon_1 + 0\varepsilon_2 + 0\varepsilon_3 + 0\varepsilon_4 \\ \widehat{x}(10 - \widehat{x}) &= (5 + \varepsilon_1 + 0\varepsilon_2 + 0\varepsilon_3 + 0\varepsilon_4)(5 - \varepsilon_1 + 0\varepsilon_2 + 0\varepsilon_3 + 0\varepsilon_4) \\ &= 25 + 0\varepsilon_1 + 0\varepsilon_2 + 0\varepsilon_3 + 1\varepsilon_4\end{aligned}$$

The enclosure corresponding to the use of the second affine representation **AF2**, gives $[24, 25]$ which is exactly the range of f over the interval $[4, 6]$.

Remark. If the second expression of f is used ($f(x) = 10x - x^2$) then the same results are obtained by using affine or quadratic forms.

In this article, two new affine forms **AF1**, **AF2** have been introduced to improve standard affine arithmetic. By using the same principle, other affine forms could also be generated. For instance, all symbolic variables in $[0, 1]$ or in [-1,0].

4 Some Extensions of Affine Forms

4.1 Quadratic Form QF

A logical extension of **AF2**, consists in keeping the three added symbolic variables, and introducing new square symbolic variables $\varepsilon_i^2 (= \varepsilon_{n+i})$. The three symbolic variables are denoted in this case by $\varepsilon_{2n+1} \in [-1, 1]$, $\varepsilon_{2n+2} \in [0, 1]$, $\varepsilon_{2n+3} \in [-1, 0]$. Thus the second difficulty presented in Section 2.5 will be reduced again.

4.1.1 Representation of QF

$$\widehat{x} = x_0 + \sum_{i=1}^{n} x_i\varepsilon_i + \sum_{i=1}^{n} x_{i+n}\varepsilon_i^2 + x_{2n+1}\varepsilon_{2n+1} + x_{2n+2}\varepsilon_{2n+2} + x_{2n+3}\varepsilon_{2n+3}, \quad (15)$$

where $\varepsilon_{2n+1}, \varepsilon_{2n+2}, \varepsilon_{2n+3}$ are new symbolic variables and $x_{2n+1}, x_{2n+2}, x_{2n+3}$ must always be positive numbers as with **AF2**. In order to simplify the notation, ε_i^2 is denoted ε_{n+i}, with $\varepsilon_i^2 = \varepsilon_{i+n} \in [0, 1]$.

4.1.2 Operations with QF

$$\widehat{x}+\widehat{y}=(x_0+y_0)+\sum_{i=1}^{2n}(x_i+y_i)\varepsilon_i+(x_{2n+1}+y_{2n+1})\varepsilon_{2n+1}+(x_{2n+2}+y_{2n+2})$$
$$\varepsilon_{2n+2}+(x_{2n+3}+y_{2n+3})\varepsilon_{2n+3},$$
$$\widehat{x}-\widehat{y}=(x_0-y_0)+\sum_{i=1}^{2n}(x_i-y_i)\varepsilon_i+(x_{2n+1}+y_{2n+1})\varepsilon_{2n+1}+(x_{2n+2}+y_{2n+3})$$
$$\varepsilon_{2n+2}+(x_{2n+3}+y_{2n+2})\varepsilon_{2n+3},$$
$$a+\widehat{x}=(a+x_0)+\sum_{i=1}^{2n}x_i\varepsilon_i+x_{2n+1}\varepsilon_{2n+1}+x_{2n+2}\varepsilon_{2n+2}+x_{2n+3}\varepsilon_{2n+3}, \tag{16}$$
$$a-\widehat{x}=(a-x_0)-\sum_{i=1}^{2n}x_i\varepsilon_i+x_{2n+1}\varepsilon_{2n+1}+x_{2n+3}\varepsilon_{2n+2}+x_{2n+2}\varepsilon_{2n+3},$$
$$a\times\widehat{x}=\begin{cases} ax_0+\sum_{i=1}^{2n}ax_i\varepsilon_i+ax_{2n+1}\varepsilon_{2n+1}+ax_{2n+2}\varepsilon_{2n+2}+ax_{2n+3}\varepsilon_{2n+3} \text{ if } a>0,\\ ax_0+\sum_{i=1}^{2n}ax_i\varepsilon_i+|a|x_{\,2n+1}\varepsilon_{2n+1}+|a|x_{\,2n+3}\varepsilon_{2n+2}+|a|x_{\,2n+2}\varepsilon_{2n+3} \text{ else.}\end{cases}$$

The multiplication is somewhat more difficult:

$$\widehat{x}\times\widehat{y}=x_0y_0+\sum_{i=1}^{n}(x_0y_i+x_iy_0)\varepsilon_i+\sum_{i=1}^{n}(x_0y_{i+n}+x_{i+n}y_0+x_iy_i)\varepsilon_{i+n}$$
$$+K_1\varepsilon_{2n+1}+K_2\varepsilon_{2n+2}+K_3\varepsilon_{2n+3}. \tag{17}$$

Computation of the three positive values K_1, K_2, K_3

– Initialization:

$$K_1:=|x_0|y_{2n+1}+|y_0|x_{2n+1}$$
$$K_2:=\begin{cases} x_0y_{2n+2}+y_0x_{2n+2} \text{ if } x_0>0 \text{ and } y_0>0\\ x_0y_{2n+2}-y_0x_{2n+3} \text{ if } x_0>0 \text{ and } y_0<0\\ -x_0y_{2n+3}+y_0x_{2n+2} \text{ if } x_0<0 \text{ and } y_0>0\\ -x_0y_{2n+3}-y_0x_{2n+3} \text{ if } x_0<0 \text{ and } y_0<0\end{cases}$$
$$K_3:=\begin{cases} x_0y_{2n+3}+y_0x_{2n+3} \text{ if } x_0>0 \text{ and } y_0>0\\ x_0y_{2n+3}-y_0x_{2n+2} \text{ if } x_0>0 \text{ and } y_0<0\\ -x_0y_{2n+2}+y_0x_{2n+3} \text{ if } x_0<0 \text{ and } y_0>0\\ -x_0y_{2n+2}-y_0x_{2n+2} \text{ if } x_0<0 \text{ and } y_0<0\end{cases}$$

– Iterations:

$$K_1:=K_1+x_{2n+1}y_{2n+1}+\sum_{i=1}^{n}(|x_i|(y_{2n+1}+y_{2n+2}+y_{2n+3})+|y_i|(x_{2n+1}$$

$$
\begin{aligned}
&+x_{2n+2}+x_{2n+3})+|x_{i+n}|y_{2n+1}+|y_{i+n}|x_{2n+1})\\
&+\sum_{i=1}^{n}\sum_{j=1}^{n}(|x_iy_{j+n}|+|y_ix_{j+n}|)+\sum_{i=1}^{n}\sum_{j=1,i\neq j}^{n}x_iy_i,\\
K_2 := {}& K_2+x_{2n+2}y_{2n+2}+x_{2n+3}y_{2n+3}+\sum_{i=1,x_iy_{2n+2}>0}^{n}x_iy_{2n+2}\\
&+\sum_{i=1,y_ix_{2n+2}>0}^{n}y_ix_{2n+2}+\sum_{i=1,x_iy_{2n+3}>0}^{n}x_iy_{2n+3}\\
&+\sum_{i=1,y_ix_{2n+3}>0}^{n}y_ix_{2n+3}+\sum_{i=1}^{n}\sum_{j=1,x_{i+n}y_{j+n}>0}^{n}x_{i+n}y_{j+n},\\
K_3 := {}& K_3+x_{2n+2}y_{2n+3}+x_{2n+3}y_{2n+2}+\sum_{i=1,x_iy_{2n+2}<0}^{n}|x_iy_{2n+2}|\\
&+\sum_{i=1,y_ix_{2n+2}<0}^{n}|y_ix_{2n+2}|+\sum_{i=1,x_iy_{2n+3}<0}^{n}|x_iy_{2n+3}|\\
&+\sum_{i=1,y_ix_{2n+3}<0}^{n}|y_ix_{2n+3}|+\sum_{i=1}^{n}\sum_{j=1,x_{i+n}y_{j+n}<0}^{n}|x_{i+n}y_{j+n}|.
\end{aligned}
$$

Remark. One must be careful because some terms are permuted (see the subtraction between ε_{n+2} and ε_{n+3} as with **AF2**). The conversions between interval and this quadratic form are similar to the standard affine form. However, the symbolic variables $\varepsilon_{i+n} = \varepsilon_i^2$ are replaced with the interval $[0, 1]$, the symbolic variable ε_{2n+1} with $[-1, 1]$, the symbolic variable ε_{2n+2} with $[0, 1]$, and ε_{2n+3} with $[-1, 0]$.

4.2 Taylor Inclusion Functions with Affine and Quadratic Forms

Taylor inclusion functions require the computation of enclosures of the derivatives of the considered function over a box.

These enclosures can be computed by using one of affine or quadratic forms. This leads to construct the following inclusion functions based on $\mathbf{T}_1$ and $\mathbf{T}_B$ which are denoted by:

– ${\mathbf{T}_1}^{AF}$ and ${\mathbf{T}_B}^{AF}$, where an enclosure of the gradient is computed by using the standard affine arithmetic **AF**,

– ${\mathbf{T}_1}^{AF1}$ and ${\mathbf{T}_B}^{AF1}$, where an enclosure of the gradient is computed by using the first new affine representation **AF1**,

– ${\mathbf{T}_1}^{AF2}$ and ${\mathbf{T}_B}^{AF2}$, where an enclosure of the gradient is computed by using the second new affine representation **AF2**,

- $\mathbf{T}_1{}^{QF}$ and $\mathbf{T}_B{}^{QF}$, where an enclosure of the gradient is computed by using the new quadratic representation **QF**,

This approach will be validated below in some numerical tests.

4.3 Conversions between all these Forms

Proposition 3. *All these affine and quadratic forms* **AF***,* **AF1***,* **AF2***,* **QF** *can be converted between them.*

Proof. All these forms can be converted into an interval and reciprocally. Therefore, each form can be converted into another form via the utilization of a conversion into an interval.

However, every conversion can be directly performed as follows (the underlined terms represent the differences between the two considered representations):

- **AF** $\longrightarrow$ **AF1**

$$\begin{aligned}\hat{x} =&\ x_0 + \sum_{i=1}^{n} x_i\varepsilon_i + \sum_{k=n+1}^{K} x_k\varepsilon_k \\ \longrightarrow&\ \hat{x} = x_0 + \sum_{i=1}^{n} x_i\varepsilon_i + \underline{\left(\sum_{k=n+1}^{K} |x_k|\right)\varepsilon_{n+1}},\end{aligned}$$

- **AF1** $\longrightarrow$ **AF**

$$\underline{K = n+1},$$

- **AF1** $\longrightarrow$ **AF2**

$$\begin{aligned}\hat{x} =&\ x_0 + \sum_{i=1}^{n} x_i\varepsilon_i + x_{n+1}\varepsilon_{n+1} \\ \longrightarrow&\ \hat{x} = x_0 + \sum_{i=1}^{n} x_i\varepsilon_i + x_{n+1}\varepsilon_{n+1}\underline{+0\varepsilon_{n+2} + 0\varepsilon_{n+3}},\end{aligned}$$

- **AF2** $\longrightarrow$ **AF1**

$$\begin{aligned}\hat{x} =&\ x_0 + \sum_{i=1}^{n} x_i\varepsilon_i + x_{n+1}\varepsilon_{n+1} + x_{n+2}\varepsilon_{n+2} + x_{n+3}\varepsilon_{n+3} \\ \longrightarrow&\ \hat{x} = x_0 + \sum_{i=1}^{n} x_i\varepsilon_i\underline{+ \left(x_{n+1} + \max\{x_{n+2}, x_{n+3}\}\right)\varepsilon_{n+1}},\end{aligned}$$

– **AF2** $\longrightarrow$ **QF**

$$\begin{aligned}\hat{x} = \ & x_0 + \sum_{i=1}^{n} x_i\varepsilon_i + x_{n+1}\varepsilon_{n+1} + x_{n+2}\varepsilon_{n+2} + x_{n+3}\varepsilon_{n+3} \\ \longrightarrow\ & \overline{x} = x_0 + \sum_{i=1}^{n}(x_i\varepsilon_i \underline{+0\varepsilon_i^2}) + x_{n+1}\varepsilon_{2n+1} + x_{n+2}\varepsilon_{2n+2} + x_{n+3}\varepsilon_{2n+3},\end{aligned}$$

– **QF** $\longrightarrow$ **AF2**

$$\begin{aligned}\overline{x} = \ & x_0 + \sum_{i=1}^{n}(x_i\varepsilon_i + x_{n+i}\varepsilon_i^2) + x_{2n+1}\varepsilon_{2n+1} + x_{2n+2}\varepsilon_{2n+2} + x_{2n+3}\varepsilon_{2n+3} \\ \longrightarrow\ & \hat{x} = x_0 + \sum_{i=1}^{n} x_i\varepsilon_i \underline{+ x_{2n+1}\varepsilon_{n+1} + \left(x_{2n+2} + \sum_{i=1, x_{n+i}\geq 0}^{n} x_{n+i}\right)\varepsilon_{n+2}} \\ & \underline{+ \left(x_{2n+2} + \sum_{i=1, x_{n+i}<0}^{n} |x_{n+i}|\right)\varepsilon_{n+3}}.\end{aligned}$$

Remark. Thus, by combining further different affine or quadratic forms, new inclusion functions could be constructed.

4.4 Other Non-affine Operations

It is easy to understand that all the non-affine operations can be extended by using conversions into standard affine form or into interval. For example, let us consider the division $\frac{\widehat{x}}{\widehat{y}}$:

> The inverse function $\frac{1}{\widehat{y}}$ is computed by replacing $\widehat{y}$ with its corresponding interval, denoted $[\widehat{y}]$. Thus, the inversion is done by using interval arithmetic and then the resulting interval is converted into the corresponding affine or quadratic form.
>
> *Remark.* A new symbolic variable is generated by using standard affine form. With **AF1** and **AF2**, the symbolic variable ε_{n+1} can directly be used, respectively ε_{2n+1} for **QF**.
>
> The division $\frac{\widehat{x}}{\widehat{y}} := \widehat{x} \times \frac{1}{\widehat{y}}$, $\frac{1}{\widehat{y}}$ is first performed and then the multiplication by $\widehat{x}$ is done with the inverse result converted into the corresponding form.

And so on for all other functions such as the square root, the absolute value or the sinus, etc.

Remark. If some function (like the square root) get a computation using the standard affine arithmetic then the conversions into interval is replaced by the conversions into standard affine form **AF** .

5 Inclusion Functions and Global Optimization

As with interval arithmetic, the purpose of all these affine and quadratic forms is to construct new inclusion functions.

5.1 Affine or Quadratic Inclusion Functions

Theorem 4. *Affine or Quadratic inclusion functions of a function f over a box X (an interval vector) are obtained by the following steps:*

$$X = (X_1, \cdots, X_n)^T, X_i = [x_i^L, x_i^U]$$

1. *Convert all the interval X_i into an affine or quadratic form* **AF**, **AF1**, **AF2**, **QF**
$$\widehat{x} = (\widehat{x}_1, \cdots, \widehat{x}_n) \text{ with } \widehat{x}_i = \frac{x_i^L + x_i^U}{2} + \frac{x_i^U - x_i^L}{2}\varepsilon_i$$
2. *A calculation of the resulting affine or quadratic form is given by replacing each occurrence of the variable x_i in an expression of f, with its corresponding affine or quadratic form* **AF**, **AF1**, **AF2**, **QF** *(denoted by $\widehat{x}_i$), and the standard operations by the corresponding affine or quadratic operations.*
3. *Convert the resulting affine or quadratic expression into an interval.*

The resulting interval encloses the range of f over the box X.

Proof. The proof is not given here because it involves formal difficulties. Since all these forms are elaborated to construct new inclusion functions, we can easily understand that such new forms of arithmetic should be correct.

Some elements of proof can be made available in [12].

In order to give an idea of the proof, let us consider the previous example $f(x) = x(10-x)$, with $x \in [4, 6]$, presented in Subsection 3.3. The corresponding affine form was $\widehat{x} = 5 + \varepsilon_1$.

Hence, $\forall y \in [4, 6]$, it exists $\alpha_1 \in [-1, 1]$ such that $y = 5 + \alpha_1$ and $10 - y = 5 - \alpha_1$.

The multiplication gives:

$$\begin{aligned} f(y) = y(10-y) &= (5+\alpha_1)(5-\alpha_1) \\ &= 25 + 0\alpha_1 - \alpha_1^2 \end{aligned} \tag{18}$$

But, since α_1 is in $[-1,1]$ then α_1^2 is in $[0,1]$. Furthermore, since there exists $\alpha_2 = \alpha_1^2 \in [0,1] \subset [-1,1]$ such that $f(y) = 25 - \alpha_1^2 = 25 - \alpha_2 \in 25 + [-1,0] \subset 25 + [-1,1]$, for all $y \in [4,6]$. Thus on this example, an inclusion function of f over a box is obtained by using **AF1**, **AF2** or **QF**.

Conjecture 5. *For polynomial functions, the inclusion functions constructed with affine or quadratic forms have an α-convergence at the order 2.*

5.2 Branch and Bound Algorithm

In this paper, the classical interval Branch and Bound algorithm due to Ichida and Fujii is used [9]. The computation of lower bounds uses the following different methods **NE**, $\mathbf{T}_1$, $\mathbf{T}_B$, **AS**, and **AF**, **AF1**, **AF2**, **QF**. All these inclusion functions are applied to perform the *Cut-Off test*, presented in Definition 1. The *middle point test* is computed by using rounded interval analysis [15].

6 Numerical Results

The purpose of these experiments is firstly to show the efficiency of the utilization of the new representations **AF1**, **AF2**, **QF** versus standard affine form **AF** and some other well-known methods **NE**, $\mathbf{T}_1$, $\mathbf{T}_B$, **AS**. Secondly, the application to global optimization permits to show the interest of these new approaches.

All the following algorithms were developed in Fortran 90 and performed on a DIGITAL Alpha-Server 8200 5/625 quadriprocessors with 2Gb of memory (in order to get the same results on a Pentium 233MHz with 64Mb of memory, the obtained CPU-times must be multiply by approximately 8).

6.1 Tests on Inclusion Functions

Firstly, the accuracy of the computations of lower bounds is only discussed. To perform this, ten thousand polynomial functions using the following shape are randomly generated:

$$\begin{aligned} P(x,y) = \ & a_1x^4 + a_2x^3 + a_3x^2 + a_4x + a_5y^4 + a_6y^3 + a_7y^2 + a_8y + a_9xy \\ & + a_{10}xy^2 + a_{11}xy^3 + a_{12}x^2y + a_{13}x^2y^2 + a_{14}x^3y + a_{15} \end{aligned}$$

where the real coefficients a_i are randomly taken in the interval $[-10,10]$.

These polynomial functions are evaluated on different boxes centered on a random point in $[-1,1]$ with a fixed width $10, 5, 1, \cdots$ until 0.01. The columns of the following table 1 represent the average of the ten thousand lower bounds: $\displaystyle\sum_{i=1}^{10000} \frac{P_i^L(X,Y)}{10000}$, where $P_i^L(X,Y)$ represents a lower bound of the i^{th} random polynomial function, computed with **NE**, $\mathbf{T}_1$, $\mathbf{T}_B$, **AS**, **AF**, **AF1**, **AF2** or **QF**. +AD signifies that an interval automatic differentiation code is used [10, 11]. The last column represents the total CPU-times in seconds for all the 10000 evaluation of the corresponding inclusion function.

Remark. The fact that the center of the random point is in $[-1, 1]$ decreases the efficiency of affine methods because of the second difficulty, see Section 2.5. This emphasizes the interest of the following numerical results.

Widths :	10	5	1	0.5	0.1	0.05	0.01	Time(s)
NE	-18639.08	-1865.682	-43.198	-17.4519	-3.1372	-1.6864	$-18.3194\ 10^{-2}$	0.318
AF	-26838.49	-2704.460	-45.109	-12.9097	-1.3646	-0.7309	$1.6715\ 10^{-2}$	0.795
AF1	-26838.47	-2704.457	-45.109	-12.9097	-1.3646	-0.7309	$1.6731\ 10^{-2}$	0.237
AF2	-21387.10	-2222.359	-36.943	-10.9904	-1.2883	-0.7119	$1.7494\ 10^{-2}$	0.954
QF	-20993.61	-2123.686	-32.997	-10.0127	-1.2492	-0.7020	$1.7884\ 10^{-2}$	0.833
$\mathbf{T}_1$+AD	-85986.55	-7295.003	-74.934	-18.8363	-1.5890	-0.7873	$1.4451\ 10^{-2}$	6.487
$\mathbf{T}_1$	-85986.52	-7295.000	-74.934	-18.8363	-1.5890	-0.7873	$1.4451\ 10^{-2}$	0.603
${\mathbf{T}_1}^{AF}$	-91266.38	-7788.866	-65.766	-15.0183	-1.3672	-0.7290	$1.6847\ 10^{-2}$	1.245
${\mathbf{T}_1}^{AF1}$	-91266.36	-7788.862	-65.766	-15.0183	-1.3672	-0.7290	$1.6847\ 10^{-2}$	0.584
${\mathbf{T}_1}^{AF2}$	-87018.78	-7241.939	-59.965	-14.2453	-1.3609	-0.7282	$1.6853\ 10^{-2}$	1.252
${\mathbf{T}_1}^{QF}$	-83960.52	-6858.191	-56.908	-13.8616	-1.3578	-0.7278	$1.6857\ 10^{-2}$	1.172
$\mathbf{T}_B$+AD	-75813.97	-5849.806	-50.776	-12.0081	-1.1821	-0.6803	$1.8965\ 10^{-2}$	6.199
$\mathbf{T}_B$	-75813.94	-5849.804	-50.776	-12.0081	-1.1821	-0.6803	$1.8965\ 10^{-2}$	0.653
${\mathbf{T}_B}^{AF}$	-91147.86	-7728.425	-53.801	-9.6267	-1.1399	-0.6745	$1.9012\ 10^{-2}$	1.134
${\mathbf{T}_B}^{AF1}$	-91147.81	-7728.420	-53.801	-9.6267	-1.1399	-0.6745	$1.9012\ 10^{-2}$	0.767
${\mathbf{T}_B}^{AF2}$	-82005.94	-6558.010	-43.811	-8.6828	-1.1391	-0.6745	$1.9012\ 10^{-2}$	1.297
${\mathbf{T}_B}^{QF}$	-78938.30	-6171.462	-40.732	-8.4274	-1.1389	-0.6744	$1.9012\ 10^{-2}$	1.275
AS+AD	-76467.97	-5964.837	-51.640	-12.0879	-1.1823	-0.6803	$1.8965\ 10^{-2}$	6.926
AS	-76467.95	-5964.835	-51.640	-12.0879	-1.1823	-0.6803	$1.8965\ 10^{-2}$	1.137

Table 1: Quality Tests for Different Inclusion Functions

These numerical results emphasize the real interest of affine representations. If one compares the standard affine form with the first new affine representation, we can note that the CPU time is divided by three with exactly the same results. This proves the interest to discard the first difficulty due to the use of standard affine arithmetic. The second affine form and the quadratic form confirm that the second difficulty, inherent to the use of affine arithmetic, can be really attenuated (about 25% of improvement). Thus, the quadratic representation **QF** gives always the best lower bounds compared to **AF**, **AF1**, **AF2**, with an interesting CPU-time close to **AF**.

In comparison to other inclusion functions **NE**, $\mathbf{T}_1$, $\mathbf{T}_B$, **AS**, we can remark that (according to these numerical results):

– when the studied box is wide, **NE** is the best method, however **QF** produces similar results,

– when the studied box has not too small a width, **QF** produces the best results,

– when the studied box has a small width (less than 0.01), $\mathbf{T}_B$ or **AS** produces the best results. However, these results can be improved by computing an enclosure of the gradient by using affine or quadratic forms. Furthermore, **QF** gives directly equivalent results.

In order to summarize, **QF** produces (on these polynomial examples) efficient enclosures which are always very close to the best method. Table 1 shows the difference of inclusion function behaviors between on one hand **NE** which generally has an $\alpha-$convergence at the order 1 and on the other hand $\mathbf{T}_1$ which has an $\alpha-$convergence at the order 2.

Remark. **AS** (which is the most efficient method when the widths of the box are very small) computes only lower bounds. That is not the case for all other methods (attention must be paid to $\mathbf{T}_B$ due to Baumann [3], because the lower bounds are optimal but not the upper bounds and conversely). Observe that, the computation of the gradient by using automatic differentiation (denoted by +AD) is very expensive in CPU-time.

¿From these first numerical results, the interest of the improvements of the standard affine arithmetic by the introduction of new representations is numerically shown. Thus, for small boxes, the new inclusion function constructed by using quadratic forms **QF**, produces enclosures very close to $\mathbf{T}_1$, $\mathbf{T}_B$ and **AS** methods which require the expensive computation of an enclosure of the gradient. That seems to confirm the above Conjecture 5.

6.2 Tests on Optimization Problems

In this paper, the global minimizers of polynomial functions of $2, 3$, or 4 variables with different degrees were searched with a great accuracy, less than 10^{-4}.

The stopping criterion of the Ichida-Fujii Branch and Bound algorithm [9] occurs when the maximal width of each remaining box, is less than 10^{-4} for all the considered following functions, except 10^{-3} for f_6.

Remark. In this paper, it is difficult to compare all these methods versus the efficient affine global optimization code developed in [7], because it depends on the compiler, the computer, the branch and bound algorithms: ordering of the elements in the list (lower bound, oldest box), the implementation of the list (using arrays, pointers). Thus, standard affine arithmetic **AF** has been re-implemented in this work to carry out remarks and comparisons between all these different techniques **NE**, $\mathbf{T}_1$, $\mathbf{T}_B$, **AS**, **AF**, **AF1**, **AF2** and **QF**.

The upper bounds produced by all these inclusion functions are not taken into account. The different methods are only compared with the *lower bounds* they produce.

$$
\begin{aligned}
f_1(x) &= 1+(x_1^2+2)x_2+x_1x_2^2,\ \text{with } X=[1,2]\times[-10,10],\\
f_2(x) &= 2x_1^2-1.05x_1^4+x_2^2-x_1x_2+\frac{1}{6}x_2^6,\ \text{with } X=[-2,4]^2,\\
f_3(x) &= (x_1-2x_2-7)^2+(2x_1+x_2-5)^2,\ \text{with } X=[-2.5,3.5]\times[-1.5,4.5]\\
f_4(x) &= [1+(x_1+x_2+1)^2(19-14x_1+3x_1^2-14x_2+6x_1x_2+3x_2^2)]\\
&\quad\times[30+(2x_1-3x_2)^2(18-32x_1+12x_1^2+48x_2-36x_1x_2+27x_2^2)]\\
&\quad \text{with } X=[-2,2]^2,\ \text{Golstein Price function},\\
f_5(x) &= (x_1-1)(x_1+2)(x_2+1)(x_2-2)x_3^2,\ \text{with } X=[-2,2]^3\\
f_6(x) &= 4x_1^2-2x_1x_2+4x_2^2-2x_2x_3+4x_3^2-2x_3x_4+4x_4^2+2x_1-x_2+3x_3\\
&\quad +5x_4,\ \text{with } X=[-1,3]\times[-10,10]\times[1,4]\times[-1,5]\\
f_7(x) &= 0.26(x_1^2+x_2^2)-0.48x_1x_2,\ \text{with } X=[-100,100]^2,\ \text{Matyas function},\\
f_8(x) &= 4x_1^2-2.1x_1^4+\frac{1}{3}x_1^6+x_1x_2-4x_2^2+4x_2^4,\ \text{with } X=[-1000,1000]^2,\\
&\quad \text{Ratschek function},\\
f_9(x) &= 12x_1^2-6.3x_1^4+x_1^6+6x_2(x_2-x_1),\ \text{with } X=[-100,100]^2,\\
&\quad \text{Three Hump function},\\
f_{10}(x) &= 6.94x_1^4+0.96x_1^3+9.68x_1^2+4.16x_1+7.53x_2^4-7.68x_2^3+8.21x_2^2-1.75x_2\\
&\quad -7.45x_1x_2+9.15x_1x_2^2+3.70x_1x_2^3-4.81x_1^2x_2-3.06x_1^2x_2^2-0.79x_1^3x_2\\
&\quad -0.18,\ \text{with } X=[-50,50]^2,\ \text{A random polynomial function.}
\end{aligned}
$$

Except for f_{10}, all these functions are taken in the literature [8, 14, 16].

In the following tables, these notations are used:

- **N**: Number of iterations,
- **T(s)**: CPU time in seconds,
- **C**: Number of elements which enclose the optimizers in the list L at the end of the algorithm,
- —: the algorithm does not give any solution after 100000 iterations, and then the average and the ratio are not computed.

– **Avg**: the mean value between the 10 considered functions,

– **Ratio**: the ratio between all the methods and $\mathbf{T}_1$.

– In the following Tables **NE**, $\mathbf{T}_1$, $\mathbf{T}_B$, **AS**, **AF**, **AF1**, **AF2** and **QF** denote the Ichida-Fujii optimization algorithm using respectively as inclusion function **NE**, $\mathbf{T}_1$, $\mathbf{T}_B$, **AS**, **AF**, **AF1**, **AF2** and **QF**.

Pbs	**NE**			$\mathbf{T}_1$			$\mathbf{T}_B$			**AS**		
	N	T(s)	C	N	T(s)	C	N	T(s)	C	N	T(s)	C
f_1	6333	0.70	1308	1363	0.15	206	1327	0.15	206	1327	0.18	206
f_2	4598	0.43	1152	4266	0.97	1073	4241	1.12	1073	4239	2.75	1073
f_3	926	0.04	154	741	0.08	117	710	0.05	117	712	0.07	117
f_4	—	—	—	6657	7.04	923	4581	3.81	914	4512	3.78	915
f_5	25087	17.60	4624	13311	8.79	2256	13015	9.47	2264	13015	9.58	2262
f_6	—	—	—	3949	1.68	210	980	0.39	109	832	0.36	113
f_7	13261	0.60	380	1629	0.16	42	539	0.11	14	353	0.08	8
f_8	—	—	—	4425	1.62	836	4012	0.57	832	1580	0.36	778
f_9	6657	0.69	10	3385	0.75	14	1408	0.18	6	759	0.13	6
f_{10}	73398	53.92	30049	1572	1.16	90	654	0.35	81	612	0.21	79
Avg	—	—	—	4130	2.24	577	3147	1.62	562	2794	1.75	556
Ratio	—	—	—	—	—	—	24%	28%	3%	32%	22%	4%

Table 2: Global Optimization using Standard Inclusion Functions

In table 2, **AS** [14] and $\mathbf{T}_B$ [3] produce generally the most interesting results. In comparison to $\mathbf{T}_1$ an improvement about 20% is noted. All these computations are performed by using interval automatic differentiation [10, 11].

Observe in table 2 that the clustering problem is strongly reduced if $\mathbf{T}_1$, $\mathbf{T}_B$ or **AS** are considered, see Section 1.2.

In table 3, the improvements of the standard affine arithmetic is shown. Generally, the quadratic representation produces the most interesting results. However, as these polynomial examples are often "simple" (low degree, few occurrences of the variables), the first or the second new affine forms are sufficient (there are not enough occurrences of each variables to produce interesting dependency problems).

The comparison with the two previous tables shows that the best CPU-time results are obtained by **AF1** and **QF**. A gain about 80% for the CPU-time and about 33% for the number of iterations is noted. We can remark that $\mathbf{T}_1$, $\mathbf{T}_B$, **AS**, **AF**, **AF1**, **AF2** and **QF** produce equivalent results when just the number of clusters are considered. Nevertheless a substantial gain for affine and quadratic forms is noted when the function f_4 is considered.

Pbs	AF			AF1			AF2			QF		
	N	T(s)	C	N	T(s)	C	N	T(s)	C	N	T(s)	C
f_1	1344	0.05	203	1342	0.04	203	1338	0.05	203	1339	0.05	204
f_2	4260	0.45	1077	4250	0.45	1070	4244	0.41	1071	4244	0.40	1073
f_3	724	0.02	116	724	0.02	116	719	0.02	116	719	0.02	116
f_4	2068	0.26	458	1774	0.22	453	1774	0.21	453	1678	0.23	454
f_5	13064	4.02	2255	13044	2.96	2238	13044	4.08	2242	13067	3.27	2264
f_6	2518	0.28	176	2443	0.21	145	1559	0.15	120	1596	0.17	149
f_7	1241	0.03	32	1241	0.02	32	985	0.02	26	985	0.02	26
f_8	3516	0.58	1046	2997	0.29	552	2108	0.10	588	2100	0.11	680
f_9	2264	0.13	14	2264	0.08	14	1387	0.06	8	1257	0.05	8
f_{10}	1161	0.15	138	1084	0.06	60	775	0.10	61	708	0.13	70
Avg	3216	0.60	552	3116	0.44	488	2793	0.52	489	2769	0.45	504
Ratio	22%	73%	4%	25%	81%	15%	32%	77%	15%	33%	80%	13%

Table 3: Global Optimization using Affine and Quadratic Forms

Therefore, in these polynomial examples, we can note that affine methods have the same behaviour as methods which have an α-convergence at the order 2. Thus, the Conjecture 5 seems to be numerically confirmed.

Remark. Sometimes, we do not get the same number of iterations between **AF** and **AF1**, that seems to be due to floating approximations and floating computations.

Remark. All these numerical tests have also been performed with $\mathbf{T}_1{}^{AF}$, $\mathbf{T}_1{}^{AF1}$, $\mathbf{T}_1{}^{AF2}$, $\mathbf{T}_1{}^{QF}$, $\mathbf{T}_B{}^{AF}$, $\mathbf{T}_B{}^{AF1}$, $\mathbf{T}_B{}^{AF2}$, $\mathbf{T}_B{}^{QF}$. Nevertheless, as the polynomial functions are "simple" (low degree and few occurrences), no significant improvement has been noted, except for f_4 where **QF** (used for the computation of enclosures of the gradient) improves considerably the three methods $\mathbf{T}_1$, $\mathbf{T}_B$ and also **AS**. Doing so, **AS** produces the best numerical result with 823 iterations during 0.15 seconds, and 198 clusters.

These numerical results show the efficiency of the three new inclusion functions. These approaches are validated on polynomial functions, but could be also easily extended to other cases even if the functions are not differentiable.

7 Conclusion

In this paper, we emphasize the interest of using affine arithmetic to reduce the dependency problem, presented in Section 1.1. Indeed, the new forms **AF1**, **AF2** and **QF**, permit to improve standard affine arithmetic by discarding and attenuating the two problems introduced in Section 2.5. Since all the conversions between all the representations (interval, **AF**, **AF1**, **AF2**, **QF**) can be carried out, many other new forms could be constructed analogously. Furthermore, extensions to non-polynomial functions are possible even if the considered function is not differentiable.

The numerical tests presented here show perfectly the improvements (with respect to both accuracy and CPU-time) provided by the methods using **AF1**, **AF2**, and **QF**, in comparison to the standard **NE**, $\mathbf{T}_1$, $\mathbf{T}_B$, **AS** and **AF**. Furthermore, the application of these techniques to global optimization leads to think that **AF**, **AF1**, **AF2** and **QF** have the same behavior as $\mathbf{T}_1$and $\mathbf{T}_B$, $\mathbf{T}_1$ and $\mathbf{T}_B$ having an α-convergence at the order 2. That seems to corroborate the above Conjecture 5, without needing enclosures of the gradient.

These new methods could also be applied at several stages of interval Branch and Bound algorithms:

1. for the construction of *Inclusion Function*, as we did in this paper,
2. for the computation of the enclosure of the gradient: improving thereby $\mathbf{T}_1$, $\mathbf{T}_B$ and **AS**, and also improving the monotonicity test,
3. for the computation of the enclosure of the Hessian: improving thereby $\mathbf{T}_2$, and the convexity test and also the *interval Newton steps*, refer to [8].

Finally, the approach presented in this paper can be implemented in an analogous manner in order to create other new forms yielding the construction of new inclusion functions perfectly adapted to a given problem.

References

1. G. Alefeld and J. Herzberger *Introduction to Interval Computations*, ACADEMIC PRESS, INC., 11 Fifth Avenue, New York, New York 10003 - (1983).
2. M.V.A. Andrade, J.L.D. Comba and J. Stolfi *Affine Arithmetic*, Interval'94, St. Petersburg (Russia), March 5-10 - (1994).
3. E. Baumann *Optimal Centerd Form*, BIT, Vol. 28, pp 80-87 - (1988).
4. J.L.D. Comba and J. Stolfi *Affine Arithmetic and its Applications to Computer Graphics*, Sibgrapi'93, Recife, PE (Brazil), October 20-22 - (1993).
5. K. Du, R. B. Kearfott *The Cluster Problem in Multivariate Global Optimization*, Journal of Global Optimization : 10 (27-32) - (1996).
6. L.H. De Figueiredo *Surface Intersection using Affine Arithmetic*, Proceedings of Graphics Interface'96, 168-175 - (1996).
7. L.H. De Figueiredo, R. Van Iwaarden and J. Stolfi *Fast Interval Branch and Bound Methods for Unconstrained Global Optimization with Affine Arithmetic*, Submitted in SIAM Journal of Optimization, available on ftp://ftp.icad.puc-rio.br/pub/lhf/doc/go.ps.gz - (1997).
8. E. Hansen *A generalized interval arithmetic*, in Interval Mathematics, K. Nickel, ed., no. 29 in Lecture Notes in Computer Science, Springer Verlag, 331-344 - (1975).
9. K. Ichida, Y. Fujii *An Interval Arithmetic Method for Global Optimization*, Computing : 23 (85-97) - (1979).
10. R. B. Kearfott, *Rigorous Global Search: Continuous Problems*, Kluwer Academic Publishers, Dordrecht, Boston, London - 1996.

11. F. Messine *Méthodes d'Optimisation Globale basées sur l'Analyse d'Intervalle pour la Résolution de Problèmes avec Contraintes*, Ph D Thesis, Polytechnic National Institute of Toulouse France - (1997).
12. F. Messine *Theorem on Affine or Quadratic Inclusion Functions*, Internal Report number R2I_01_04, available on www.univ-pau.fr/~messine/R2I.html, Computer Science Department of Pau France - (2001).
13. F. Messine, B. Nogarede, J.L. Lagouanelle *Optimal Design of Electromechanical Actuators: A New Method Based on Global Optimization*, IEEE Transactions on Magnetics, Vol. 34, No 1, pp 299-307 - (1998).
14. F. Messine, J.L. Lagouanelle *Enclosure Methods for Multivariate Differentiable Functions and Application to Global Optimization*, Journal of Universal Computer Science, Vol. 4, No 6, pp 589-603, Springer Verlag - (1998).
15. R. E. Moore *Interval Analysis*, Prentice Hall, Inc. Englewood Cliffs, N.J. - (1966).
16. H. Ratschek, J. Rokne *New Computer Methods for Global Optimization*, Ellis Horwood Limited Market Cross House, Cooper Street, Chichester, West Sussex, PO19 1EB, England - (1988).