

# *Tutorial on Interval Computation*

*Rolando Flores, Olga Garay, Jorge Santillan, Christian Servin,  
and Carlos Silva*

## **Abstract**

The aim of this tutorial is to show how interval arithmetic can be used to compute the error bounds. We will also show that there are other extensions of interval arithmetic (IA), due to its drawbacks. IA has a very serious drawback known as the dependency problem. The dependency problem or the overestimation problem as we also know of it is when a computed interval can be much larger than the exact range. In order to fix this such problem, we will show you examples in which natural extensions and the Horner's extension reduces this situation of interval arithmetic [3].

**Keywords:** Interval Computation, Interval Arithmetic, Real Interval Arithmetic, Interval Extensions, Floating-Point Arithmetic, The Horner's Extension.

## **1 Introduction**

Measurements are used today in many applications. The problem is that they are not as precise as we want them to be. This is the reason why we use interval computations as opposed to using real numbers. An example of such a problem is when we are measuring our weight, and another example is when we are measuring distances between objects. Let's assume that when asking a person what his/her weight is, they answer that they weight 160 pounds, so how sure is that person that he/she weighs that much. So how can we be sure that the measurement is exact? There is a possibility that the scale used in this example said that the weight was 160, but in this case the scale had an accuracy range  $[-4, +4]$ . As a consequence, the person's weight will not be exactly 160 pounds, but it will be in the range of  $[156, 164]$ . Any number between this range can be the actual person's weight. As you can see the data used here is not as exact as we wanted it to be, even though the results in this example are not critical, but in another situation it can be a bit different. Now lets take another example into consideration. Let's assume you want to measure the distance between an airplane and its landing field, let's assume the distance is 100 meters. What if the accuracy range in this case is from  $[-100, +100]$  meters? In this case, you cannot rely on these values because there is a big chance that the airplane will crash.

The problem in this situation is "how to obtain the inaccuracy bound of these measurements". In many cases we have the accuracy bound, but in many other cases we don't. These are two simple examples on how interval computation is used in simple situations, but more than we often know about there is interval computation involved.

There has been many hearsay that computers are always right, unfortunately, computers are far from being right and far from being reliable machines. At least in ways in which common everyday users use them and the way in which they are applied to common applications. A way to guarantee reliability in computations performed by an everyday computer is to replace all floating-point computations for interval computations. In other words, the way it works is that every real number is replaced in computations by the smallest interval that it contains. This way, by following some rules on the bounds of the interval, interval computations are guaranteed to always contain the expected real quality. This is the reason why we will be talking about interval computations as opposed to floating-point computations.

Unfortunately, when gaining reliability of computations, we jeopardize somehow the accuracy of computations. And in case we need to reach a given accuracy, it is at the expense of time, because interval computations have this bad habit of overestimating the quantities that are supposed to be computed. So now we can be reliable but at the expense of performance. However, critical applications require reliable and performance systems. Therefore, we will be using symbolic identities and the Horner's Extension to reduce this situation. Later on in the tutorial we will go more into detail on symbolic identities and Horner's Extension.

## 2 Intervals

### 2.1 Real Interval Arithmetic

Interval Arithmetic (IA) was proposed by Ramon E. Moore [1] in the late sixties in order to model uncertainty, and to tackle rounding errors of numerical computations. It is actually a tool for performing arithmetic on intervals. Interval Arithmetic has three different properties which consist of associativity, commutativity, and sub-distributivity. Notice that the distributive law is not included in the three properties of interval arithmetic, the reason being is that it just does not hold with respect for addition. The associative law, the commutative law, and the sub-distributive law are pre-

served over real intervals. For all  $x, y, z \in \mathbb{IR}$ , we have:

**associativity:**  $(x + y) + z = x + (y + z)$

**commutativity:**  $(x + y) = (y + x)$

**sub-distributivity:**  $x \times (y + z) \subseteq x \times y + x \times z$

The following rules that will be listed below are the rules that are followed in interval arithmetic. Given two intervals  $x = [a, b]$ , and  $y = [c, d]$ , we have:

$$\left\{ \begin{array}{ll} x + y & = [a + c, b + d] \\ x - y & = [a - d, b - c] \\ x \times y & = [\min\{ac, ad, bc, bd\}, \max\{ac, ad, bc, bd\}] \\ x \div y & = [\min\{a/c, a/d, b/c, b/d\}, \max\{a/c, a/d, b/c, b/d\}] \\ & \quad [-\infty, +\infty] \quad \text{if } 0 \in [c, d] \\ x^n & = [a^n, b^n] \quad \text{if } n \text{ is an odd natural number} \\ & \quad [0, \max\{|a|, |b|\}^n] \quad \text{if } n \text{ is even} \end{array} \right.$$

For example, we define two intervals A and B. Assume  $A = [-1, 2]$  and  $B = [1, 2]$ . What is  $A + B$ ,  $B - A$ ,  $B \times A$ ,  $A/B$ ,  $A^2$  ?

1.  $A + B = [-1+1, 2+2] = [0, 4]$
2.  $A - B = [-1-2, 2-1] = [-3, 1]$
3.  $B \times A = [\min\{(-1)(1), (-1)(2), (2)(1), (2)(2)\}, \max\{(-1)(1), (-1)(2), (2)(1), (2)(2)\}]$   
 $= [\min\{(-1), (-2), (2), (4)\}, \max\{(-1)(2)(2)(4)\}]$   
 $= [\min(-2), \max(4)]$   
 $= [-2, 4]$
4.  $A \div B = [\min\{-1/1, -1/2, 2/1, 2/2\}, \max\{-1/1, -1/2, 2/1, 2/2\}]$   
 $= [\min\{-1, -1/2, 2, 1\}, \max\{-1, -1/2, 2, 1\}]$   
 $= [\min(-1/2), \max(2)]$   
 $= [-1/2, 2]$
5.  $A^2 = [0, 4]$

Interval arithmetic is important to us because real world problems engage real numbers or floating point numbers. The major problem is that floating-point numbers are logically disconnected from science and engineering, because of these disconnections, floating point numbers contain no accuracy

information. As a result we have crucial situations such as the example we gave above for the airplane landing. In this example, recall that when computing the speed, unpredicted risks and results can lead to drastic situations. Therefore, we use interval arithmetic to create a tight logical connection between computing and engineering projects. Using interval computation, you compute correct results the first time, and we are guaranteed the accuracy of the computed results. The width of intervals can be used to represent error in an approximation. As a result, Interval Arithmetic gives us a whole new look at things.

## 2.2 Interval Extensions

Interval Extensions [1] are computed using interval arithmetic, when a variable occurs only once in the function. When working with interval extensions, you just have to apply the same interval arithmetic rules as the examples given above with the intervals A and B.

(Interval Extension): The interval extension  $F(X)$  includes all values of  $f(x)$  for  $x \in X$ :  $F(X) \supseteq \{ f(x) \mid x \in X \}$

On the contrary, when a variable occurs more than once in a function, just using the interval extension is not enough, because the results obtained using interval arithmetic are not as precise as we want. Just using the interval extension would yield us to obtain inaccurate error bounds. Therefore, another extension must be enforced in order to deal with the dependency problem. This extension is called the natural extension.

The following expression as you can see has two occurrences of the variable  $x$ , the first example will not use the natural extension obviously not eliminating the occurrences. The second example will use the natural extension therefore evaluating the dependency problem.

Example 1:

$$f(x) = x / (x - 1)$$

evaluated for  $x = [2,3]$

$$\begin{aligned} F([2,3]) &= [2,3] / ([2,3] - 1) \\ &= [2,3] / [1,2] \end{aligned}$$

$$= [1,3]$$

Now we will use the same expression, but evaluated even further, and using the natural extension.

Example 2:

$$f(x) = x / (x - 1) = 1 + 1 / (x - 1),$$

$$\begin{aligned} F([2,3]) &= 1 + 1 / ([2,3] - 1) \\ &= 1 + 1 / [1,2] \\ &= 1 + [0.5,1] \\ &= [1.5,2] \end{aligned}$$

The results from Example 1 and Example 2, although they may seem similar, are fairly different. Example 2 is much more accurate and is in a tighter interval. As a result, using the natural extension is very useful when two or more occurrences of the same variable occur.

In Figure 1, we see that two functions define interval extensions of  $f$ . However, one function is clearly better.

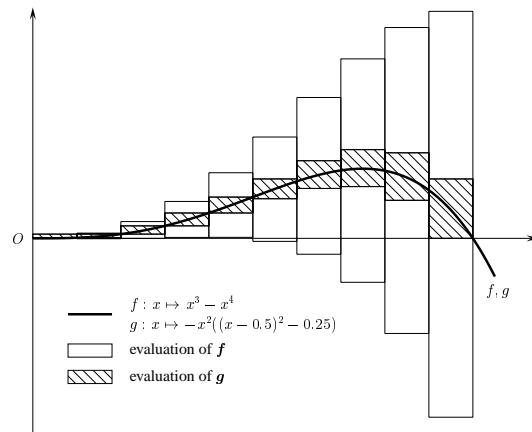


Figure 1: Natural interval evaluations of two expressions of a real function  $f$ .

## 2.3 Floating Point Arithmetic

When floating point intervals are used, the set of real intervals  $\mathbb{IR}$  is replaced with the set of floating point intervals  $\mathbb{IF}$ . This means that a floating point interval is a real interval bounded by floating point numbers. The main difference between them is that computations over  $\mathbb{IF}$  need to be rounded. Let  $E$  be the set of machine representable IEEE (The Institute of Electrical and Electronics Engineers) floating point numbers, which is a finite subset of the rational numbers. Given a floating point number  $b$ , let  $b^+$  denote  $\min \{x \in \mathbb{F} \mid x > b\}$  and  $b^-$  denote  $\max \{x \in \mathbb{F} \mid x < b\}$ .

Now, given a real number  $y \in \mathbb{R}$  we can define the downward and upward rounding as follows:

Downward rounding:  $\lfloor y \rfloor = \max \{x \in \mathbb{F} \mid x \leq y\}$ .

Upward rounding:  $\lceil y \rceil = \min \{x \in \mathbb{F} \mid x \geq y\}$ .

$\mathbb{IR}$  are mapped then on  $\mathbb{F}$  (floating point intervals) by the outward bound as follows:

$$[a, b] \in \mathbb{IR} \rightsquigarrow [\lfloor a \rfloor, \lceil b \rceil] \in \mathbb{IF}$$

Floating point interval arithmetic belongs to the Real interval arithmetic where all the intermediate results of interval computations are outward rounded.

### 2.3.1 Overestimation Intervals

Overestimation interval theorem states that the Horner's extension over  $\mathbb{IR}$ , exactly evaluates the range of the real function, only if the domain of  $x$  does not intersect with the overestimation interval. In other words,  $x$  cannot be zero.

In order to obtain the overestimation interval we denote this: Let  $p: \mathbb{R} \rightarrow \mathbb{R}$ , be a polynomial, where the overestimation interval of  $p$  is the Hull of the set of roots of the intermediate polynomials of  $h_p$ , plus 0,  $O_p = \{0\} \cup \mathbf{Hull}(\{x \in \mathbb{R} \mid \exists i \in \{0, \dots, n\}, p_i(x) = 0\})$ .

Now, lets consider a polynomial  $p: \mathbb{R} \rightarrow \mathbb{R}$ .

Then, for all  $x \in \mathbb{R}$ . We get:  $\overset{\circ}{x} \cap O_p = \emptyset \implies \mathbf{h}_p(\mathbf{x}) = \mathbf{p}^u(\mathbf{x})$ , where  $\mathbf{x}$  is the interior of  $\overset{\circ}{x}$  interval. This theorem can be illustrated as follows:

The respective overestimation interval [1]  $O_p$  is  $[0,1]$ . In this case, we will examine an example according to the intersection  $\mathbf{x} \cap O_p$ .

$\mathbf{x} = [0.5,0.6]$ . In this case, there is a strict inclusion  $\mathbf{p}^u(\mathbf{x}) \subset \mathbf{h}_p(\mathbf{x}) \subset \mathbf{p}(\mathbf{x})$ .

### 2.3.2 Overestimation Problems

Horner's rule can be applied to ordered sequences of powers with no problem. However, the decomposition of even powers can be problematic to interval evaluations.

## 3 The Horner Extension

### 3.1 Horner's Rule

The Horner's Rule[1] is a rule for polynomial computation which both reduces the number of necessary multiplications and results in less numerical instability due to potential subtraction of one large number from another. The rule simply factors out powers of  $x$ , giving:

$$a_n x^n + a_{n-1} x^{n-1} + \dots + a_0 = (\dots(\mathbf{a}_n \mathbf{x} + a_{n-1})\mathbf{x} + \dots)\mathbf{x} + a_0$$

Horner's rule, rediscovered by William G. Horner five hundred years after Chu Shih-Chieh.

Now, let's consider the following polynomial:

$$p(x) = a_0 + \sum_{i=1}^n a_i x^{\alpha_i}$$

Let  $\alpha_i$  be the hull of  $a_i$  for  $i = 0, 1, 2, \dots, n$ . Then, the Horner extension of  $p(x)$  over  $\mathbb{I}$  is defined as the natural extension of  $h_p(x)$  as follows:

$$h_p(x) = (\dots(\mathbf{a}_n \mathbf{x}^{\alpha_n} + a_{n-1})\mathbf{x}^{\alpha_{n-1}} + \dots)\mathbf{x}^{\alpha_1} + a_0$$

There is a problem with Horner extension, even though it performs evaluations on tight intervals on average, his proposition is not valid at all since the sub distributive law only holds for non-condensed extensions. (Which

are going to be discussed later).

Finally, we need to recall that the notion of Horner's extension [1] can be defined for quasipolynomials, which are the polynomials where the coefficients are defined as terms.

### 3.2 Condensed and non-condensed forms

Now, let  $p$  be a polynomial and lets consider the Horner extension of the polynomial  $p$

$$h_p(x) = (...(\mathbf{a}_n \mathbf{x}^{d_n} + a_{n-1}) \mathbf{x}^{d_{n-1}} + ...) \mathbf{x}^{d_1} + a_0$$

As we now, the sub-distributive law only guarantees the following inclusion for non-condensed expression as follows:

$$(..(\mathbf{a}_n \overbrace{\mathbf{x} \dots \mathbf{x}}^{d_n \text{ times}} + a_{n-1}) \overbrace{\mathbf{x} \dots \mathbf{x}}^{d_{n-1} \text{ times}} + ...) \overbrace{\mathbf{x} \dots \mathbf{x}}^{d_1 \text{ times}} + a_0 \subseteq + \sum_{i=1}^n a_i \overbrace{\mathbf{x} \dots \mathbf{x}}^{\beta_i \text{ times}}$$

Where  $\beta_i = d_1 + d_2 + d_3 + \dots + d_i$ . Then the products of  $(x \dots x)$  are replaced with the powers in the  $h_p(x)$ . As you can see we cannot say that is always better than for intervals evaluation.

Know, let's do some examples: Consider four terms  $p$ ,  $h_p(x)$ ,  $q$ , and  $r$  defining the same function  $f$ .

$$\begin{aligned} p(x) &= x + x^4 & h_p(x) &= x(x^3 + 1) \\ q(x) &= x + xxx & r(x) &= x(xxx + 1) \end{aligned}$$

Given  $\mathbf{x} = [-1,1]$ , we have:

$$\begin{aligned} \mathbf{p}(\mathbf{x}) &= [-2,17] & \mathbf{h}_p(\mathbf{x}) &= [-7,14] \\ \mathbf{q}(\mathbf{x}) &= [-10,17] & \mathbf{r}(\mathbf{x}) &= [-10,14] \end{aligned}$$

As you can see here, the sub-distributive law is preserved since  $\mathbf{r}(\mathbf{x}) \subset \mathbf{q}(\mathbf{x})$ . Even though, no expression is the best one. Despite the above mentioned limitation, Horner extension can provide with the exact range of the corresponding function over the reals in some particular cases. Which lead us to define the following:

## 4 Example of a function using Horner's Extension

Next is a simple example using Horner's Extension. First, we start with the first expression, we evaluate it and get a second function, evaluating that even further, we get the final function. As we are evaluating each function individually, you will notice that the final function is much easier to evaluate, because it takes less computations. As a result, the third function, has a smaller interval therefore we obtain a smaller range.

$$x^2 - x = x(x - 1) = (x - 1/2)^2 - 1/4$$

where  $x \in [0, 1]$

Functions

(1).  $x^2 - x$

(2).  $x(x - 1)$

(3).  $(x - 1/2)^2 - 1/4$

Function Number 1

$$\begin{aligned} & [0, 1]^2 - [0, 1] \\ &= ([0, 1] - [0, 1]) \\ &= [-1, 1] \end{aligned}$$

Function Number 2

$$\begin{aligned} & [0, 1]([0, 1] - 1) \\ &= [0, 1] * [-1, 0] \\ &= [-1, 0] \end{aligned}$$

Function Number 3

$$\begin{aligned} & ([0, 1] - 1/2)^2 - 1/4 \\ &= [-1/2, 1/2]^2 - 1/4 \\ &= [0, 1/4] - 1/4 \\ &= [-1/4, 0] \end{aligned}$$

## 5 Applications in Artificial Intelligence

### 5.1 Computer Graphics and Computer-Aided Design

Computer Graphics [2] is an application that uses interval computation. Interval computations are well-matched in such operations as surface intersection and hidden line removal, which require robustness in nonlinear equation solvers that can be provided by interval computations. As a result, the low-degree polynomial systems and constraints that arise in such operations are easy for interval solvers.

Early, Mudur and Koparkar provide a review of interval analysis techniques that can be useful in computational geometry. Maekawa [2] uses interval-based techniques to robustly solve shape interrogation problems in computational geometry; such solutions are important in automated manufacturing of free-form objects.

In another application, that has used interval computations is computer-aided design [2]. Enger has shown how to use interval ray tracing to greatly speed up ray tracing algorithms without sacrificing image quality. The basic idea is to take care of regions in the image having nearly constant intensity with a single interval ray, rather than with many point rays. This is a brief explanation in which interval computations are used in computer design. For these reasons, there have been many applications in computer graphics and computer-aided design that use interval computations.

### 5.2 Remote Sensing and Geographic Information Systems

Remote Sensing and Geographic Information Systems [2] are also applications that use interval computation. Hager uses interval methods to take account of bounded errors in the data in decisions based on remote sensing. Lodwick uses interval methods in sensitivity analysis in geographic information systems.

### 5.3 Expert Systems

Yet, a final application that uses interval computation are expert systems [2]. Kohout develop interval-valued inference handle different logical properties of knowledge representations from different medical specialist fields. They apply interval-valued inference to CLINAID, a general medical diagnosis expert system.

## 6 Directions for our project

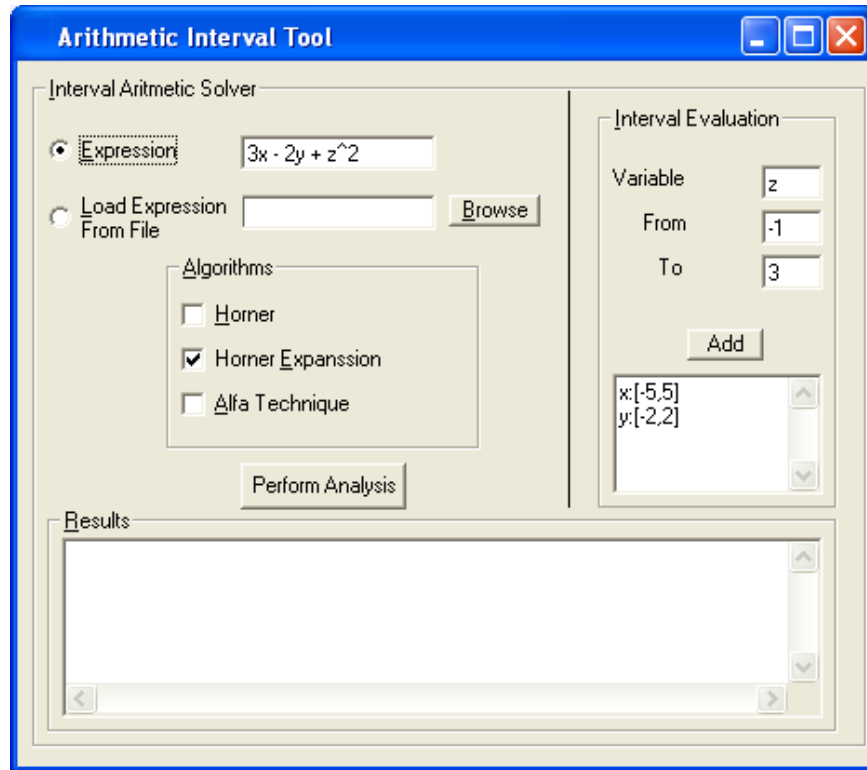


Figure 2: Our interface.

The Interval Arithmetic Solver is a tool that allows a user to enter the expressions, he/she wants evaluated in two different ways:

1. Manually: The user is allowed to input the expression manually.
2. Loading from a file: The user has the option of loading an expression that is already saved in a file.

The system also has an option where the user is allowed to set the desired intervals where the expressions will be evaluated.

The interface displays a list of algorithms that the user may select from that will allow the user to apply these algorithms in the evaluation of the

arithmetic expression.

Finally, there is a results section, where the system will print the best option after the evaluation.

This is a prototype of our system, as the semester goes on we will do modifications as needed.

## References

1. Martine Ceberio and Laurent Granvilliers. Horner's Rule for Interval Evaluation Revisited.
2. Interval Computations: Introduction, Uses, and Resources.
3. Applications of Interval Computations