Calculations & Rounding Errors

Joseph Pousada - Sept 15th, 2013

While humans who use mathematics will grapple with number systems that expand different levels of infinity such as Z, < , = & C, machines and computers have limitations on the numbers they are able to handle. (These data types are usually referred to as primitive data types.) With modern computers available to the public we have the following basic data structures and their numerical limitations:

"Name Range Storage Size

byte -128 to 127 8-bit signed

short -32768 to 32767 16-bit signed

int -2147483648 to 2147483647 32-bit signed

long -9223372036854775808 to 9223372036854775807 64 bit signed

float -3.4E38 to 3.4E38 (6 to 7 digits of accuracy) 32 bit IEEE 754

double -1.7E308 to 1.7E308 (14 to 15 significant digits of accuracy) 64-bit IEEE 754" (Liang, Y. D. ,pg 50,(2002))

Since the different data structures have limitations when calculations are performed there is a certain level of error that occurs. Let us take a look at Maple and have it calculate the famous number e to 50 digits.

e;

$$e \tag{1}$$

$$\xrightarrow{\text{at 50 digits}}$$

$$2.7182818284590452353602874713526624977572470937000 \tag{2}$$

So if we consider 5 digit chopping method we go to the 5th digit and then just chop off the remainder. For e this would be 2.7182 . If we use the rounding method then we go to the 5th digit and see that the next digit (6th one) is the number 8. Since this is greater than 5 we round up and get 2.7183.

Now we can compare the effects of chopping off and rounding off with a simple example. Let's look at the effect of evaluating e^3 using both methods to the 10th digit 5 digit and 2 digit chopping and round off methods.

Using the 10th digit chopping method:

$(2.718281828)\cdot(2.718281828);$

$$7.389056096 \quad \textbf{(3)}$$

$7.389056096\cdot(2.718281828);$

$$20.08553691 \quad \textbf{(4)}$$

Since we would round down the 11th digit using the rounding method we would wind up with the same 10 digit number and would yield he same result of 20.08553691 .

Using the 5 digit chopping method:

$(2.7182)\cdot(2.7182);$

$$7.38861124 \quad \textbf{(5)}$$

The result would yield 7.3886.
$7.3886\cdot(2.7182);$

$$20.08369252 \quad \textbf{(6)}$$

The result would yield 20.083.

Using the 5 digit rounding method:

$(2.7183)\cdot(2.7183);$

$$7.38915489$$

$7.3892\cdot(2.7183);$

$$20.08606236 \quad \textbf{(8)}$$

The result would yield 20.086

Using the 2 digit chopping method:

$(2.7)\cdot(2.7);$

$$7.29 \quad \textbf{(9)}$$

$7.2\cdot(2.7);$

$$19.44 \quad \textbf{(10)}$$

And our result with the two digit chopping method would be 19.

If we chose the two digit rounding method our results would be:
$(2.7)\cdot(2.7);$

$$7.29 \quad \textbf{(11)}$$

$7.3\cdot(2.7);$

$$19.71 \quad \textbf{(12)}$$

Since we would round up the 3rd digit using the rounding method we would wind up with 20.

To gauge the level of error let us look first at the absolute error which is $|p-p^*|$ and the weighted error referred to as relative error which is $\frac{|p-p^*|}{|p|}$. First let us look at the 10 digit calculations (the results were the same for chopping and round off methods).

$e^3 - 20.08553691;$

$$e^3 - 20.08553691 \tag{13}$$

at 10 digits →

$$1.\,10^{-8} \tag{14}$$

$\frac{(e^3 - 20.08553691)}{e^3};$

$$\frac{e^3 - 20.08553691}{e^3}$$

at 10 digits →

$$4.978706838\,10^{-10} \tag{16}$$

Now we can look at the 5 digit chopping method

$e^3 - 20.083;$

$$e^3 - 20.083 \tag{17}$$

at 10 digits →

$$0.00253692 \tag{18}$$

$\frac{(e^3 - 20.083)}{e^3};$

$$\frac{e^3 - 20.083}{e^3}$$

at 10 digits →

$$0.0001263058095 \tag{20}$$

Now we can look at the 5 digit rounding method

$e^3 - 20.086;$

$$e^3 - 20.086 \tag{21}$$

at 10 digits →

$$-0.00046308 \tag{22}$$

$$\frac{(e^3 - 20.086)}{e^3};$$

$$\frac{e^3 - 20.086}{e^3}$$

$$\xrightarrow{\text{at 10 digits}}$$

$$-0.000023055395 62 \tag{24}$$

Now, let us look at the 2 digit chopping method

$$e^3 - 19;$$

$$e^3 - 19 \tag{25}$$

$$\xrightarrow{\text{at 10 digits}}$$

$$1.08553692 \tag{26}$$

$$\frac{(e^3 - 19)}{e^3};$$

$$\frac{e^3 - 19}{e^3}$$

$$\xrightarrow{\text{at 10 digits}}$$

$$0.05404570086 \tag{28}$$

Now we can look at the two digit rounding method.

$$e^3 - 20;$$

$$e^3 - 20 \tag{29}$$

$$\xrightarrow{\text{at 5 digits}}$$

$$0.086 \tag{30}$$

$$\frac{(e^3 - 20)}{e^3};$$

$$\frac{e^3 - 20}{e^3}$$

$$\xrightarrow{\text{at 10 digits}}$$

$$0.004258632485 \tag{32}$$

As we can see there was a low level of error for the 10 digit calculations. For the five digit calculations there was a slightly higher error for the chop off method versus the rounding method but both had higher errors than the 10 digit calculations but, the level of error was moderate. For the 2 digit calculations the error was quite high. Naturally, one might wonder why programmers don't use the double primitive type all the time. The answer lies in what the application they are using is going to do. Many times a long or a float might be very sufficient for the application they are developing and in order to save data space they pick the appropriate size of data type that gives them the level of accuracy needed. One quick example would be a mom and pop shop that sells pizza and ice pops. All monetary denominations end at a penny so the level of accuracy would be to two digits and chances are that it will not exceed the limit of let's pick a float - even in one years's time for accounting purposes.

Many times the level of error gets enhanced with complex calculations so the rounding error gets compounded. Here we will look at a five digit rounding up for $ex^2 + \sqrt{17}\,x + 1 = 0$ and take a look at the level of error that takes place in solving this quadratic equation.

e;

$$e \tag{33}$$

at 10 digits →

$$2.718281828 \tag{34}$$

$\sqrt{17}$;

$$\sqrt{17} \tag{35}$$

at 10 digits →

$$4.123105626 \tag{36}$$

The five digit rounding values are
$a = 2.7183$, $b = 4.1231$, $c = 1.0000$

Using the standard quadratic equation we have:

$$x_1 = \frac{\left(-b + \sqrt{(b)^2 - 4\,ac}\right)}{2\,a} \;\&\; x_2 = \frac{\left(-b - \sqrt{(b)^2 - 4\,ac}\right)}{2\,a}$$

Using the five digit rounded numbers we get:

$b^2 = 4.1231 \cdot 4.1231$;

$$b^2 = 16.99995361 \tag{37}$$

Five digit rounded is 17.000
(After squaring the five digit rounded root of 17 you can see that it gets us close to 17 - but there is error included and it would be evident if we used the five digit chopping

method here.)

$4\,ac = -4(2.7183)\cdot(1.0000);$

$$4\,ac = -10.87320000 \quad \textbf{(38)}$$

Five digit rounded is -10.873

$2\,a = 2\cdot 2.7183;$

$$2\,a = 5.4366 \quad \textbf{(39)}$$

Five digit rounded is 5.4366

$b^2 - 4\,ac$ is equal to :
$(17.000) - 10.873;$

$$6.127 \quad \textbf{(40)}$$

Five digit rounded is 6.1270

$\sqrt{(b^2) - 4\,ac}$ becomes:

$\sqrt{6.1270};$

$$2.475277762 \quad \textbf{(41)}$$

Five digit rounded is 2.4753

So for x_1 we consider the following:
$\left(-b + \sqrt{(b)^2 - 4\,ac}\right)$ becomes:

$-4.1231 + (2.4753);$

$$-1.6478 \quad \textbf{(42)}$$

Five digit rounded is -1.6478

$x_1 = \dfrac{\left(-b + \sqrt{(b)^2 - 4\,ac}\right)}{2\,a}$ becomes

$\dfrac{-1.6478}{5.4366};$

$$-0.3030938454 \quad \textbf{(43)}$$

Rounding we get -.30309
So for x_2 we consider the following:
$\left(-b - \sqrt{(b)^2 - 4\,ac}\right)$ becomes:

$-4.1231 - (2.4753);$

$$-6.5984 \quad \textbf{(44)}$$

Five digit rounded is -6.5984

$x_2 = \frac{\left(-b - \sqrt{(b)^2 - 4\,ac}\right)}{2\,a}$ becomes:

$\frac{-6.5984}{5.4366}$;

$$-1.213699739 \quad \textbf{(45)}$$

-1.2136

The exact values are:

$x_1 = \frac{\left(-b + \sqrt{(b)^2 - 4\,ac}\right)}{2\,a}$

$\frac{\left(-\sqrt{17} + \sqrt{\left(\sqrt{17}\right)^2 - (4\cdot(e)\cdot(1))}\right)}{2\cdot(1)}$;

$$-\frac{1}{2}\sqrt{17} + \frac{1}{2}\sqrt{17 - 4\,e} \quad \textbf{(46)}$$

at 10 digits →

$$-0.823926790 \quad \textbf{(47)}$$

$x_2 = \frac{\left(-b - \sqrt{(b)^2 - 4\,ac}\right)}{2\,a}$

$\frac{\left(-\sqrt{17} - \sqrt{\left(\sqrt{17}\right)^2 - (4\cdot(e)\cdot(1))}\right)}{2\cdot(1)}$;

$$-\frac{1}{2}\sqrt{17} - \frac{1}{2}\sqrt{17 - 4\,e} \quad \textbf{(48)}$$

at 10 digits →

$$-3.299178836 \quad \textbf{(49)}$$

The absolute errors are:

$x_1 = |p - p^*|$

$\left(\sqrt{-0.823926790 - (-.30309)}\right)^2$;

$$-0.5208367900 \quad \textbf{(50)}$$

$x_2 = |p - p^*|$

$$\left(\sqrt{(-3.299178836)-(-1.2136)}\right)^2;$$

$$-2.085578835 \quad \textbf{(51)}$$

The relative errors are:

$$x_1 = \frac{|p-p^*|}{|p|}$$

$$\frac{\left(\sqrt{(-0.823926790)-(-0.30309)}\right)^2}{\left(\sqrt{(-0.823926790)}\right)^2};$$

$$0.6321396468 \quad \textbf{(52)}$$

$$x_2 = \frac{|p-p^*|}{|p|}$$

$$\frac{\left(\sqrt{(-3.299178836)-(-1.2136)}\right)^2}{\left(\sqrt{(-3.299178836)}\right)^2};$$

$$0.6321508896 \quad \textbf{(53)}$$

Here we see that when performing a complex computation that the relative error can compound and be quite moderate. One way of trying to mitigate this level of error is to find techniques that would modify how the calculation is done so as to lower the level of error. One option we can investigate is to rationalize the quadratic formula and investigate if this would lower the relative error.The rationalized quadratic equation is:

$$x_1 = \frac{-2c}{b+\left(\sqrt{b^2-4ac}\right)} \;\&\; x_2 = \frac{-2c}{b-\left(\sqrt{b^2-4ac}\right)}$$

Using the four digit rounded numbers we get:

$$2c = 2\cdot 1.0000;$$

$$2c = 2.0000 \quad \textbf{(54)}$$

Four digit rounded is 2.0000

Then considering $x_1 = \frac{-2c}{b+\left(\sqrt{b^2-4ac}\right)}$ becomes:

$\left(b+\sqrt{(b)^2-4ac}\right)$ becomes:

$$4.1231+2.4753;$$

6.5984 **(55)**

$\frac{-2\,c}{b+\left(\sqrt{b^2-4\,ac}\right)}$ becomes:

$\frac{-2.0000}{6.5984}$;

-0.3031037827 **(56)**

Rounded to -.30310

Then considering $x_2=\frac{-2\,c}{b-\left(\sqrt{b^2-4\,ac}\right)}$ becomes:

$\left(b\text{-}\sqrt{(b)^2-4\,ac}\right)$ becomes:

$4.1231-2.4753$;

1.6478 **(57)**

$\frac{-2.0000}{1.6478}$;

-1.213739531 **(58)**

Rounded to -1.2137

The exact values are:

$x_1=\frac{-2\,c}{b+\left(\sqrt{b^2-4\,ac}\right)}$

$\frac{-2\cdot(1)}{\sqrt{17}+\left(\sqrt{\left(\sqrt{17}\right)^2-(4\cdot(\mathrm{e})\cdot(1.0000))}\right)}$;

$-\frac{2}{\sqrt{17}+\sqrt{17-4.0000\,\mathrm{e}}}$

at 10 digits →

-0.3031057272 **(60)**

$x_2=\frac{-2\,c}{b-\left(\sqrt{b^2-4\,ac}\right)}$

$\frac{-2\cdot(1.0000)}{\sqrt{17}-\left(\sqrt{\left(\sqrt{17}\right)^2-(4\cdot(\mathrm{e})\cdot(1.0000))}\right)}$;

$$-\frac{2.0000}{\sqrt{17}-\sqrt{17-4.0000\,e}} \tag{61}$$

at 10 digits →

$$-1.213700067 \tag{62}$$

The absolute errors are:

$$x_1 = |p - p^*|$$

$$\left(\sqrt{(-0.3031057272)-(-.30310)}\right)^2;$$

$$-0.000005727200001 \tag{63}$$

$$x_2 = |p - p^*|$$

$$\left(\sqrt{(-1.213700067)-(-1.2137)}\right)^2;$$

$$-6.699999999\,10^{-8} \tag{64}$$

The relative errors are:

$$x_1 = \frac{|p - p^*|}{|p|}$$

$$\frac{\left(\sqrt{(-0.3031057272)-(-.30310)}\right)^2}{\left(\sqrt{(-0.3031057272)}\right)^2};$$

$$0.00001889505703 \tag{65}$$

$$x_2 = \frac{|p - p^*|}{|p|}$$

$$\frac{\left(\sqrt{(-1.213700067)-(-1.2137)}\right)^2}{\left(\sqrt{(-1.213700067)}\right)^2};$$

$$5.520309491\,10^{-8} \tag{66}$$

Now we can compare the standard quadratic formula and the rationalized quadratic formulas.

Relative Errors

	Standard Quadratic Formula	*Rationalized Quadratic Formula*
x_1	0.6321396468	$0.000018895057\overline{03}$
x_2	0.6321508896	$5.520309491\,10^{-8}$

For our purpose here we see that the rationalized quadratic formula significantly lowered our relative error. However, we need not rush to conclusions as such techniques may prove useful in only in limited ranges of calculations.

Many times with complicated calculations the calculated answer and the actual answer will converge at a limit. How quickly an answer converges is usually referred to as "big O" notation or the Greek letter theta which is Θ. As you may recall from Calculus II $\frac{1}{x^6}$ converges to zero much faster than $\frac{1}{x^3}$. So in big O notation we have $\Theta\left(\frac{1}{n^p}\right)$ or in the case of $\frac{1}{x^6}$ it converges in the order of $\Theta\left(\frac{1}{n^6}\right)$. The big O notation effectively classifies the functions into orders. By doing so we can effectively gauge how a complicated calculation will behave. This order of functions has many applications and is also used in programming. In programming however, big O notation is typically used to gauge the order of the function (or it's "asymptotic behavior"(Lafore, R., & Lafore, R.,pg186 (2003)) in terms of its' run time. So and example of this with programming would be an algorithm with a run time in the order of $\Theta(2^n)$ would be slow compared to an algorithm that produced the same computation but with an algorithm who's run time was in the order of $\Theta(n)$. Ideally, we would want a method of calculating a complex calculation that would minimize rounding errors by converging to the actual answer as quickly as possible, as well as the lowest run time as possible within the parameters that the algorithm selected will be used.

References

Liang, Y. D. (2002). Introduction to Java programming with Jbuilder 4. Upper Saddle River, N.J: Prentice Hall.

Lafore, R., & Lafore, R. (2003). Data structures & algorithms in Java. Indianapolis, Ind: Sams.

Faires, J. D., & Burden, R. L. (1998). Numerical methods. Pacific Grove, CA: Brooks/Cole Pub. Co.