Humboldt University of Berlin

Einführung in das wissenschaftliche Rechnen

Floating Point Arithmetic

Christian Parpart & Kei Thoma

May 31, 2019

Contents

1 Introduction

Consider a hotel with an infinite number of rooms. Each room is currently occupied, but a new guest arrives. So the question is, is it possible to accommodate the newly arrived guest? Intuition would say no, the hotel is fully booked; however, if the receptionist is keen enough, they will come to the conclusion that the solution is surprisingly simple. Every guest in the hotel is moved to the next room i.e. the guest in room one moves to two and the guest in room two moves to three and so on. This will open up the first room for the new guest to take. Now, this paradox by David Hilbert is bogus in the sense that there are no hotels with infinitely many rooms. Infinity is a concept we as mathematicians often take for granted (even the very first set we've learned about, the set of natural numbers, is infinite), but in the real physical world, there are no such things as infinity. Every amount and every measurement is ultimately finite and even if we try to find infinity in the realm of infinitesimals, we will eventually hit the wall of Planck constant, the smallest physically possible unit of length. Therefore, funnily enough, the set of the real numbers is nothing more than a misnomer. This is especially troublesome for computers which uses zero and ones to represent data. No amounts of memory are enough for a machine to truly grasp the infinite spirals pi's decimal places generate. Even simple calculations between a large and a small number proved to be challenging for a computer. Not all is lost, however. While our computers do not know the endless ocean of the real numbers, they do know a number system known as the floating point arithmetic. This number system is vastly more limited (finite that is) than the real numbers, but they are useful enough for us to navigate the computer through any calculations. As with every tool, we just have to use them right. With this goal in mind, we present in this article an introduction into floating point arithmetic. We will first cover the theoretical aspect by giving important lemmas and in the second part, these theories are verified through the output of a Python application.

2 Theory

For all following examples, let the mantissa length be $t = 8$ and the exponent of the floating point arithmetic be bounded by $N_{\min} = -5$ and $N_{\max} = 8$.

Example 2.1. Given the context as defined above, the largest number that can be represented is $x_{\max} = 255$. The calculation is fairly simple, choose the largest exponent possible and fill every digit of the mantissa with ones. In binary, this would be

$$x_{\max} = (0.11111111)_2 \times 2^8 = (11111111)_2,$$

or in decimal

$$\begin{aligned} x_{\max} &= (-1)^\nu \cdot 2^N \cdot \sum_{i=1}^{t} x_i \beta^{-i} \\ &= 2^8 \cdot \left(\frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \frac{1}{16} + \frac{1}{32} + \frac{1}{64} + \frac{1}{128} + \frac{1}{256} \right) \\ &= 255. \end{aligned}$$

Example 2.2. To find the smallest possible normalized positive value in the defined floating point arithmetic, we proceed similary to the example 2.1. Set the exponent as small as possible and fill the mantissa with zeros but the first place. We have in binary

$$x_{\text{norm. min}} = (0.10000000)_2 \times 2^{-4}$$

which in decimal this translates to

$$x_{\text{norm. min}} = (-1)^\nu \cdot 2^N \cdot \sum_{i=1}^{t} x_i \beta^{-i} = 2^{-4} \cdot \frac{1}{2} = \frac{1}{32} = 0.03125$$

Example 2.3. If we do not require the value to be normalized, the smallest possible positive value is much smaller. To find $x_{\min}$, we again set N to -4 and fill the mantissa with 0 except for the last place. We have in binary representation

$$x_{\min} = (0.00000001)_2 \times 2^{-4}$$

and decimal would be

$$x_{\min} = (-1)^\nu \cdot 2^N \cdot \sum_{i=1}^{t} x_i \beta^{-i} = 2^{-4} \times \frac{1}{256} = \frac{1}{4096} = 0.000244140625$$

Example 2.4. We want to find the margin for the absolute and the relative error. To find the largest possible absolute error, set the exponent to the maximum value and consider two neighboring floating point numbers such as $(1.00000000)_2 \times 2^8$ and $(1.00000001)_2 \times 2^8 = 1$. Worst case scenario, the given number is right in the middle of these two numbers; therefore, the maximum absolute error is $(0.00000001)_2 \times 2^7 = \frac{1}{2}$. This result is also verified by the lemma **??**. The same lemma gives us the boundaries for the relative error, 2^{-8}. To conclude, we have

$$\begin{aligned} 0 \leq e_{\text{abs}} &\leq \frac{1}{2} \\ 0 \leq e_{\text{rel}} &\leq \frac{1}{256} \end{aligned}$$

Example 2.5. Let $z_1 = 67.0$. We want to find the normalized binary form of this integer with ten decimal place accuracy. According to lemma **??**, we have

$$\begin{aligned} 67.0 \div 2 &= 33.0 + 1 \\ 33.0 \div 2 &= 16.0 + 1 \\ 16.0 \div 2 &= 8.0 + 0 \\ 8.0 \div 2 &= 4.0 + 0 \\ 4.0 \div 2 &= 2.0 + 0 \\ 2.0 \div 2 &= 1.0 + 0 \\ 1.0 \div 2 &= 0.0 + 1. \end{aligned}$$

Reading the reminders on the left from bottom to top yields $z_1 = 67.0 = (1000011)_2$. To normalize this number, we move the decimal point seven digits to the left. Since z_1 only has seven digits, we do not need to cut off any digits. We have

$$z_1 = 67.0 = (0.1000011)_2 \times 2^7$$

If one wants to check the validity of the conversion from decimal to binary above, we can check the solution by applying the formula from the other way.

$$(-1)^\nu \cdot 2^N \cdot \sum_{i=1}^{t} x_i \beta^{-i} = 2^7 \cdot \left(\frac{1}{2} + \frac{1}{64} + \frac{1}{128} \right) = 128 \cdot \frac{67}{128} = 67$$

Now, let's consider the floating point number of 67.0. $N = 7$ is between $N_{\min} = -5$ and $N_{\max} = 8$, also 67.0 has 7 digits in binary form; therefore, there is no rounding to do which means that 67.0 can be represented with the given floating point arithmetic without loss of precision.

$$\mathrm{rd}_8(z_1) = (0.1000011)_2 \times 2^7$$

Since there is no loss of precision, one can easily conclude that the absolute and relative error of 67.0 and $\mathrm{rd}_8(67.0)$ is zero.

Example 2.6. Let $z_2 = 287.0$. To find the normalized binary form with ten decimal place accuracy, we have

$$\begin{aligned} 287.0 \div 2 &= 143.0 + 1 \\ 143.0 \div 2 &= 71.0 + 1 \\ 71.0 \div 2 &= 35.0 + 1 \\ 35.0 \div 2 &= 17.0 + 1 \\ 17.0 \div 2 &= 8.0 + 1 \\ 8.0 \div 2 &= 4.0 + 0 \\ 4.0 \div 2 &= 2.0 + 0 \\ 2.0 \div 2 &= 1.0 + 0 \\ 1.0 \div 2 &= 0.0 + 1, \end{aligned}$$

therefore, $z_2 = 287.0 = (100011111)_2$. Again, there is no need to round any digits. Its normalized binary form is

$$z_2 = 287.0 = (0.100011111)_2 \times 2^9$$

In this example, we have an exponent $N = 9$ which is greater than $N_{\max} = 8$. This means that with the given floating point arithmetic, we have an overflow and 287.0 cannot be sensibly rounded to a floating point number (instead, computing with the given arithmetic, z_2 is the same as $+\infty$). In example 2.1, we showed that $x_{\max} = 255$ which is another reason for a overflow. According to IEEE 754 standard, both absolute and relative error are also infinity for 287.0.

Example 2.7. For a non-integer example, let $z_3 = 10.625$. To find the binary form of this number, we first separate $z_3 = 10.0 + 0.625$ and apply the algorithm of **??** on each summand. For 10.0 we have

$$\begin{aligned} 10.0 \div 2 &= 5.0 + 0 \\ 5.0 \div 2 &= 2.0 + 1 \\ 2.0 \div 2 &= 1.0 + 0 \\ 1.0 \div 2 &= 0.0 + 1 \end{aligned}$$

and for 0.625 we will multiply it with 2 until we get 0

$$\begin{aligned} 0.625 \times 2 &= 0.25 + 1 \\ 0.25 \times 2 &= 0.5 + 0 \\ 0.5 \times 2 &= 0.0 + 1 \end{aligned}$$

Combining both results together, we get $z_3 = (1010.101)_2$. To normalize, we move the decimal place four digits to the left and we have

$$z_3 = 10.625 = (0.1010101 \times 2^4)_2.$$

Again we see that the exponent is between $N_{\min} = -5$ and $N_{\min} = 8$. The mantissa is also short enough; therefore, z_3 is already a floating point number and we have

$$\mathrm{rd}_8(z_3) = (1.010101 \times 2^3)_2.$$

Needless to say, the absolute and relative errors are both zero.

Example 2.8. Perhaps a more interesting example is needed. Let $z_4 = 1.01$. As we did in **??**, we will separate z_4 in two parts; however, we immediately see that 1 is 1 in both

decimal and binary system. We will therefore consider 0.01.

$$\begin{aligned}
0.01 \times 2 &= 0.02 + 0 \\
0.02 \times 2 &= 0.04 + 0 \\
0.04 \times 2 &= 0.08 + 0 \\
0.08 \times 2 &= 0.16 + 0 \\
0.16 \times 2 &= 0.32 + 0 \\
0.32 \times 2 &= 0.64 + 0 \\
1.28 \times 2 &= 0.28 + 1 \\
0.28 \times 2 &= 0.56 + 0 \\
0.56 \times 2 &= 0.12 + 1
\end{aligned}$$

We could go on, but since we only need to find the normalized binary form with respect to ten decimal places. We have

$$z_4 = 1.01 \approx (1.000000101)_2 \times 2^0$$

and in normalized form

$$z_4 = 1.01 \approx (0.1000000101)_2 \times 2^1$$

Using the formula from lemma **??**, we have the floating point number

$$\mathrm{rd}_8(z_4) = (-1)^\nu \cdot 2^N \cdot \sum_{i=1}^{t} x_i \beta^{-i} = 2 \cdot \left(\frac{1}{2} + \frac{1}{256} \right) = \frac{129}{128} = 1.0078125.$$

For the absolute and relative error we have

$$\begin{aligned}
e_{\text{abs}} &= |rd_8(1.01) - 1.01| \\
&= 0.0021875 \\
e_{\text{rel}} &= \left| \frac{rd_8(1.01) - 1.01}{rd_8(1.01)} \right| \\
&= \frac{7}{3225} \approx 0.00217
\end{aligned}$$

Example 2.9. As we already fell into the rabit hole of numbers which have endlessly long binary forms, let's continue with $z_5 = 0.0002$. For this example, we must stay

diligent and iterate many times over the algorithm.

$$
\begin{aligned}
0.0002 \times 2 &= 0.0004 + 0 \\
0.0004 \times 2 &= 0.0008 + 0 \\
0.0008 \times 2 &= 0.0016 + 0 \\
0.0016 \times 2 &= 0.0032 + 0 \\
0.0032 \times 2 &= 0.0064 + 0 \\
0.0064 \times 2 &= 0.0128 + 0 \\
0.0128 \times 2 &= 0.0256 + 0 \\
0.0256 \times 2 &= 0.0512 + 0 \\
0.0512 \times 2 &= 0.1024 + 0 \\
0.1024 \times 2 &= 0.2048 + 0 \\
0.2048 \times 2 &= 0.4096 + 0 \\
0.4096 \times 2 &= 0.8192 + 0 \\
0.8192 \times 2 &= 0.6384 + 1
\end{aligned}
$$

We got our first 1! Now we only have to find a maximum of 10 more digits.

$$
\begin{aligned}
0.6384 \times 2 &= 0.2768 + 1 \\
0.2768 \times 2 &= 0.5536 + 0 \\
0.5536 \times 2 &= 0.1072 + 1 \\
0.1072 \times 2 &= 0.2144 + 0 \\
0.2144 \times 2 &= 0.4288 + 0 \\
0.4288 \times 2 &= 0.8576 + 0 \\
0.8576 \times 2 &= 0.7152 + 1 \\
0.7152 \times 2 &= 0.4304 + 1 \\
0.4304 \times 2 &= 0.8608 + 0 \\
0.8608 \times 2 &= 0.7216 + 1
\end{aligned}
$$

Therefore, we have $z_5 = 0.0002 \approx (0.00000000000011010001101)_2$ and normalized we have

$$z_5 = 0.0002 \approx (0.1101000110)_2 \times 2^{-12}.$$

Following the formula from **??** we have with $n := N_{\min} - N = -5 + 12 = 7$

$$
\begin{aligned}
\mathrm{rd}_8(z_5) &= (-1)^{\nu} \cdot 2^{N_{\min}} \cdot \left(\sum_{j=n+1}^{t} x_{j-n} 2^{-j} + 2^{-t} \right) \\
&= 2^{-5} \cdot \left(\sum_{j=8}^{8} x_{j-7} \cdot 2^{-j} + 2^{-8} \right) \\
&= \frac{1}{4096} \approx 0.000244140625
\end{aligned}
$$

since $x_{t+1-7} = x_2 = 1$. We have for the absolute and the relative error

$$e_{\text{abs}} = \left|0.0002 - \frac{1}{4096}\right| = \frac{113}{2560000} \approx 4.4140 \times 10^{-5}$$
$$e_{\text{rel}} = \frac{\left|0.0002 - \frac{1}{4096}\right|}{\frac{1}{4096}} = \frac{113}{625} = 0.1808$$

Example 2.10. For the more mathematically minded, we have last but not least $z_6 = \frac{1}{3}$.

$$\frac{1}{3} \times 2 = \frac{2}{3} + 0$$
$$\frac{2}{3} \times 2 = \frac{1}{3} + 1$$

We already see a patern here; further calculations are not needed. We simply have

$$z_6 = \frac{1}{3} \approx (0.1010101010)_2 \times 2^{-1})$$

To find the floating point we have

$$\begin{aligned}\text{rd}_8(z_6) =& (-1)^\nu \cdot 2^N \cdot \sum_{i=1}^{t} x_i \beta^{-i} \\ =& 2^{-1} \cdot \left(\frac{1}{2} + \frac{1}{8} + \frac{1}{32} + \frac{1}{128}\right) = \frac{85}{256} = 0.33203125,\end{aligned}$$

and for its errors

$$e_{\text{abs}} = \left|\frac{1}{3} - \frac{85}{256}\right| = \frac{1}{768} \approx 0.001302083$$
$$e_{\text{rel}} = \frac{\left|\frac{1}{3} - \frac{85}{256}\right|}{\frac{85}{256}} = \frac{1}{255} \approx 0.00392156.$$

3 Documentation

4 Reality

5 Bibliography