Humboldt University of Berlin

Einführung in das wissenschaftliche Rechnen

Floating Point Arithmetic

Christian Parpart & Kei Thoma

May 29, 2019

Contents

1 Introduction

It is perhaps the most profound realization

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

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

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 positive value in the defined floating point arithmetic, we proceed similary to example 2.1. Set the exponent as small as possible and fill the mantissa with zero but the last place. We have in binary

$$x_{\min} = (0.00000001)_2 \times 2^{-5}$$

which in decimal representation translates to

$$\begin{aligned} x_{\min} &= (-1)^\nu \cdot 2^N \cdot \sum_{i=1}^{t} x_i \beta^{-i} \\ &= 2^{-5} \times \frac{1}{256} \\ &= \frac{1}{8192} \end{aligned}$$

Example 2.3. 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.

$$\mathrm{rd}_8(z_1) = (1.000011)_2 \times 2^6$$

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 2.4. 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 rounded to a floating point number. Previously **??**, we have shown that $x_{\max} = 255$ which is another reason why $z_2 > x_{\max}$ cannot be expressed as a floating point number in this context. Most trivially, both absolute and relative error are also undefined for 287.0.

Example 2.5. 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 three digits to the left and we have

$$z_3 = 10.625 = (1.010101 \times 2^3)_2.$$

Example 2.6. 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.

$$\begin{aligned} 0.01 \times 2 &= 0.02 + 0 \\ 0.02 \times 2 &= 0.04 + 0 \\ 0.04 \times 2 &= 0.08 + 0 \\ 0.08 \times 2 &= 0.16 + 0 \\ 0.16 \times 2 &= 0.32 + 0 \\ 0.32 \times 2 &= 0.64 + 0 \\ 1.28 \times 2 &= 0.28 + 1 \\ 0.28 \times 2 &= 0.56 + 0 \\ 0.56 \times 2 &= 0.12 + 1 \\ 0.12 \times 2 &= 0.24 + 0 \end{aligned}$$

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

$$z_4 = 1.01 \approx (1.0000001010 \times 2^0)_2$$

which is already normalized.

Example 2.7. 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

$$z_5 = 0.0002 \approx (1.1010001101 \times 2^{-13})_2$$

Example 2.8. For the more mathematically minded, we have last but not least $z_6 = \frac{1}{3}$.

$$\begin{aligned}
\frac{1}{3} \times 2 &= \frac{2}{3} + 0 \\
\frac{2}{3} \times 2 &= \frac{1}{3} + 1
\end{aligned}$$

We already see a patern here; further calculation is not needed. We simply have

$$z_6 = \frac{1}{3} \approx (1.0101010101 \times 2^{-2})_2$$

For posterity and stripped from tedious calculation, in the following is a table summerizing the results of **??**.

decimal representation	normalized binary representation
67.0	1.000011×2^6
287.0	1.00011111×2^8
10.625	1.010101×2^3
1.01	1.0000001010×2^0
0.0002	$1.1010001101 \times 2^{-13}$
$\frac{1}{3}$	$1.0101010101 \times 2^{-2}$

Example 2.9.

3 Documentation

4 Reality

5 Bibliography