Parabola Volume 59, Issue 1 (2023)

Using a reciprocal base in number systems

Moosa Nasir[1]

Bases

In number systems, the base represents the number of different digits a number can use when representing a quantity. Each place in a number can be representing as $b^p \times d$, where b is the base, p is the number place with $p = 0$ to the right of the decimal point, and d is the digit. For example, as shown in Figure 1, the decimal number 2743.54 can be represented as

$$2743.54_{10} = 10^3 \times 2 + 10^2 \times 7 + 10^1 \times 4 + 10^0 \times 3 + 10^{-1} \times 5 + 10^{-2} \times 4\,.$$

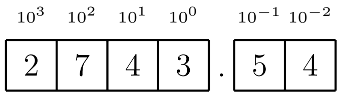


Figure 1: The decimal number 2743.54_{10} written as a base 10 number.

With this rule in mind, nothing stops us from using any value for b. Using the example in Figure 1, we can change b to $\frac{1}{10}$, the reciprocal of 10. Following the same rule for p, 2743.54_{10} now becomes $453.472_{\frac{1}{10}}$ as shown in Figure 2. A decided rule that must stay the same no matter what base a number uses is that the decimal place must be to the right side of $b^0 \times d$, the "ones place".

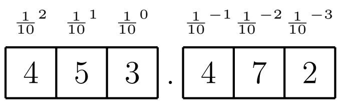
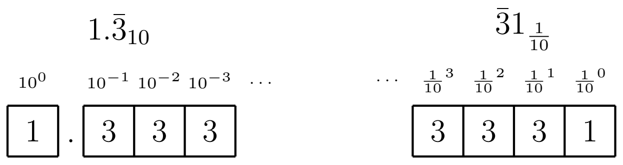


Figure 2: The decimal number 2743.54_{10} written as a base $\frac{1}{10}$ number.

Non-terminating numbers

The example shown in Figure 1 has a terminating decimal. If we were to have a non-terminating decimal number such as $1.\bar{3}_{10}$, then, in base $\frac{1}{10}$, it would look like $\bar{3}1_{\frac{1}{10}}$, as shown in Figure 3. This observation tells us that in reciprocal bases, non-terminating decimals will have a bar to the right of the decimal point after the ones place. Otherwise, the reciprocal base number would be have an infinite number of digits.

[1]Moosa Nasir is a senior student at Saint Francis Xavier Secondary School

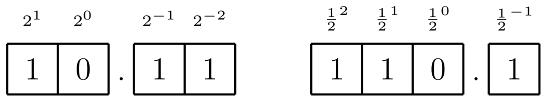


$1.\bar{3}_{10}$
$\bar{3}1_{\frac{1}{10}}$
10^0 10^{-1} 10^{-2} 10^{-3} $\cdots$
$\cdots$ $\frac{1}{10}^3$ $\frac{1}{10}^2$ $\frac{1}{10}^1$ $\frac{1}{10}^0$
1 . 3 3 3
3 3 3 1

Figure 3: The fraction $\frac{4}{3}$ written in base 10 and in base $\frac{1}{10}$.

Conversion

After looking at a few examples, we can devise an algorithm to convert from base b to its corresponding base $\frac{1}{b}$. The simplest algorithm is to $1)$ reverse the digits of the number and $2)$ move the decimal point down one place. This can be seen in Figure 4.



2^1 2^0 2^{-1} 2^{-2}
$\frac{1}{2}^2$ $\frac{1}{2}^1$ $\frac{1}{2}^0$ $\frac{1}{2}^{-1}$
1 0 . 1 1
1 1 0 . 1

Figure 4: Converting the number 2.75 from base 2 (10.11_2) to base $\frac{1}{2}$ $(110.1_{\frac{1}{2}})$.

Arithmetic

Arithmetic done for numbers in reciprocal bases is very similar to arithmetic for whole base numbers, with the only difference being that the direction of arithmetic procedures are reversed; for instance, left becomes right and vice versa.

Addition

Addition is by far the simplest arithmetic operation done in reciprocal bases. While in whole base arithmetic carrying is done to the left, carrying in reciprocal base arithmetic is done to the right, as shown in Figure 5. In the example, we first align the numbers to their one to one number places, then we start from the left and sum the digits at each column, carrying one to the right if the sum is greater than the largest possible digit d, and moving the sum subtract d down to the final sum.

$$\begin{array}{rl} & \overset{1}{7}.2_{\frac{1}{10}} \\ + & 9.5_{\frac{1}{10}} \\ \hline & 6.8_{\frac{1}{10}} \end{array}$$

Figure 5: Addition in base $\frac{1}{10}$.

Subtraction

Subtraction is also a fairly easy operation in reciprocal bases. The steps will be done to the right instead of to the left like whole base arithmetic, as shown in Figure 6. In the example, we first try to subtract 9 from 7. Since this would give us a negative difference, we will carry 1 from the tenth's place, and subtract 1 from it as well. Now we subtract 9 from 17, which gives 8. The next digit is the difference of $7 - 1$ and 5, which gives¬1.

$$\begin{array}{r}
{\scriptstyle 1}\;\;{\scriptstyle -1}\phantom{_{\frac{1}{10}}} \\
7.7_{\frac{1}{10}} \\
-\;\; 9.5_{\frac{1}{10}} \\
\hline
8.1_{\frac{1}{10}}
\end{array}$$

Figure 6: Example of subtracting two base $\frac{1}{10}$ numbers.

Multiplication

Multiplication is a little tricky to understand but, after a while, it will become easy. It is very similar to whole base arithmetic, as shown in Figure 7. In the example, we can see that instead of moving our 2 to the left, we move to the right, past the decimal point. This makes sense because $1 \times \frac{1}{10}^{0} + 2 \times \frac{1}{10}^{-1}$ is equal to 21, which is the base 10 result of 7×3.

$$\begin{array}{r}
7\;\;\;_{\frac{1}{10}} \\
\times\;\; 3\;\;\;_{\frac{1}{10}} \\
\hline
1.2_{\frac{1}{10}}
\end{array}$$

Figure 7: Multiplication in base $\frac{1}{10}$.

Division

Division is not much different from multiplication in terms of difficulty. We can apply the same rules and thought process. As long as you know division in base b, it will be trivial to do it in base $\frac{1}{b}$. As shown in Figure 8, the process is extremely similar to division in base b.

$$\begin{array}{rr} & 7_{\frac{1}{10}} \\ \times & \bar{3}0_{\frac{1}{10}} \\ \hline & \bar{3}2_{\frac{1}{10}} \end{array}$$

Figure 8: Dividing $7_{\frac{1}{10}}$ by $3_{\frac{1}{10}}$, by multiplying by $(3_{\frac{1}{10}})^{-1} = \bar{3}0_{\frac{1}{10}}$.

Scientific notation

Scientific Notation is also possible in base $\frac{1}{b}$. For example, in base 10, the scientific notation of 743 is 7.43×10^2. In base $\frac{1}{10}$, $3.47_{\frac{1}{10}}$ (which is 743 in base $\frac{1}{10}$) becomes

$$347_{\frac{1}{10}} \times 0.1^2_{\frac{1}{10}}$$

as shown in Figure 9. In base $\frac{1}{b}$, scientific notation allows for never needing the decimal point in a number.

$$34.7_{\frac{1}{10}} = 347_{\frac{1}{10}} \times 0.1^2_{\frac{1}{10}}$$

Figure 9: Example of Scientific Notation in a base $\frac{1}{10}$ number.

Conclusion

We were able to define what reciprocal bases are, how they function, how to convert between bases and reciprocal bases, and their arithmetic algorithms. Reciprocal bases might have some application. They could be used to represent extremely small numbers without dealing with decimal points. They can also be used to define and to display decimal numbers as whole numbers.

Implementation

A simple implementation in C is shown below which shows addition and subtraction in base $\frac{1}{b}$ for numbers less than b. It should help someone get started in implementing their own reciprocal base system.

```
#include <iostream>
#include <string>

using namespace std;

#define PRECISION 100

struct RecipricalB
```

```
{
   int base;
   int precision;
   int * buffer;
};

string ToString(RecipricalB n);

RecipricalB Create(int base, string str, short precision = PRECISION)
{
   RecipricalB n;
   n.base = base;
   n.precision = precision;
   n.buffer = new int[precision];

   for(int i = str.length() - 1; i >= 0; i--)
   {
      n.buffer[str.length() - i - 1] = str[i] - '0';
   }

   return n;
}

bool Add(RecipricalB n1, RecipricalB n2, RecipricalB &n3)
{
   if (n1.base != n2.base) { cout << "numbers don't have same base";
      return false; }
   if (n1.precision != n2.precision) { cout << "numbers don't have
      same precision"; return false; }

   n3 = Create(n1.base, "");

   for (int i = n1.precision; i >= 0; i--)
   {
      n3.buffer[i] = n1.buffer[i] + n2.buffer[i];
   }

   for (int i = n1.precision; i >= 0; i--)
   {
      if (n3.buffer[i] >= n3.base)
      {
         n3.buffer[i] -= n3.base;
         n3.buffer[i-1]++;
      }
   }

   return true;
```

```
}

bool Subtract(RecipricalB n1, RecipricalB n2, RecipricalB &n3)
{
   if (n1.base != n2.base) { cout << "numbers don't have same base";
      return false; }
   if (n1.precision != n2.precision) { cout << "numbers don't have
      same precision"; return false; }

   n3 = Create(n1.base, "");

   for (int i = n1.precision; i >= 0; i--)
   {
      n3.buffer[i] = n1.buffer[i] - n2.buffer[i];
   }

   for (int i = n1.precision; i >= 0; i--)
   {
      if (n3.buffer[i] < 0)
      {
         n3.buffer[i-1]--;
         n3.buffer[i] += n3.base;
      }
   }

   return true;
}

string ToString(RecipricalB n)
{
   string str = "";
   bool start = false;
   for (int i = n.precision - 1; i >= 0; i--)
   {
      if (start == false)
      {
         if (n.buffer[i] != 0)
         {
            str += to_string(n.buffer[i]);
            start = true;
         }
      }
      else
      {
         str += to_string(n.buffer[i]);
      }
   }
   return str;
}

int main()
{
   RecipricalB n1 = Create(10,"12");
   RecipricalB n2 = Create(10,"10");
   RecipricalB n3;
   Add(n1, n2, n3);
   cout << ToString(n1) << " + " << ToString(n2) << " = " <<
      ToString(n3) << endl;


   return 1;
}
```