Periodic Properties of Expansions for Fractions into any Base

by

Aidan Bowman
Aidan.Bowman@socprep.org
Socrates Preparatory School
and
Jonathan H. Yu
drysdg@gmail.com
Homeschooled in Frederick, Maryland

Sponsors: Kristina Vuong and Neal Gallagher, Ph.D.
Socrates Preparatory School
kristina.vuong@socprep.org
neal.gallagher@socprep.com

Abstract: Multiple methods are brought together here. Change of base transformation for fractions, continued products, mixed radix representations, the binary Spigot Algorithm, and repeating decimals are all used to compute a million binary digits of π, and to investigate interesting properties of repeating decimals in arbitrary bases. An Excel spreadsheet utilizes the Spigot algorithm to compute binary digits of π. A Java program listing is also included that can be used to compute a million binary digits of π.

Introduction

Often interesting number properties have analogs as the base system for expressing the number changes. For example, we know that a decimal integer is divisible by 9 if the sum of its digits is divisible by 9. In octal, base 8, an integer is divisible by 7, if the sum of its digits is divisible by 7, and a similar property holds for other number base systems. This paper deals with interesting base dependent properties for fractions.

All rational numbers can be written as the division of two integers, a numerator and denominator. $$x = \frac{p}{q}.$$

If $p < q$, this makes x a proper fraction. To convert x into a base b number, we write;

$$x = 0.\, b_1 b_2 b_3 \ldots,$$

where the digits $b_n < b$, for all n. The digits can be found as the integer part of the recursive relation involving number b times x as follows, where *int*() means the integer part:

$$b_1 = int(bx),$$

$$x_1 = bx - b_1$$

and so on
$$b_n = int(bx_{n-1} - b_{n-1}) \qquad (1)$$
$$x_n = bx_{n-1} - b_n, n \geq 2$$

Equation (1) has two steps. First, slide the new digit b_n to the units position by multiplying by b, then subtract off, leaving a new fraction. The digits drip off the number like drops off a spigot.

To illustrate how this recursion works, investigate converting the decimal 0.625 into base 8, and also base 2. First, base 8:

$$b_1 = int(bx) = int(8 \cdot 0.625) = int(5.000) = 5,$$
$$x_1 = bx - b_1 = 8 \cdot 0.625 - 5 = 0$$

The digit "5" is the first digit after the octal point. To compute succeeding digits we discover:

$$b_2 = int(bx_1 - b_1) = 0,$$

and
$$x_2 = 0.$$

All the following digits are zero.

$$b_3, b_4, \ldots = 0.$$

The decimal 0.625 in base 8 or octal is 0.5 or 5/8. Now for base 2 or binary representation,

$$b_1 = int(bx) = int(2 \cdot 0.625) = int(1.25) = 1,$$

and
$$x_1 = 2x - b_1 = 0.25 .$$

The remaining digits are computed:

$$b_2 = int(2x_1 - b_1) = int(0.5 - 1) = 0,$$
$$x_2 = 2x_1 - b_2 = 0.5,$$
$$b_3 = int(2x_2 - b_2) = int(1 - 0) = 1,$$

and
$$x_3 = 2 \cdot 0.5 - 1 = 0.$$

Continuing, it is easy to show $b_4, b_5, \ldots = 0$, resulting in the binary expansion

$$(0.625)_{10} = (0.101)_{2,} \qquad (2)$$

where $(\ )_b$ is used to indicate the base of the number. In base 10 we use a decimal point; in base 2, a binary point. In general it is called a radix point. The numbers in equation (2) can be written as continued products

$$(0.625)_{10} = \frac{1}{10}\left(6 + \frac{1}{10}\left(2 + \frac{1}{10}(5)\right)\right) = \frac{1}{2}\left(1 + \frac{1}{2}\left(0 + \frac{1}{2}(1)\right)\right) = (0.101)_2.$$

In Appendix A, we have listed a MATLAB R2018a script for taking a number's integer part and decimal part converting from base 10 to any base. Note that a finite length decimal fraction may convert to an infinite length term in some other base.

Note that the repeated term 1/10 in the continued product indicates that the number is base 10 while the term 1/2 on the right of the expression indicates base 2. The process can be complicated when a radix expansion does not terminate. One base expansion may terminate, while the other does not such as $1/3 = (0.\bar{3})_{10} = (0.1)_3$. So sometimes extra digits need to be included in the computation to allow for carries in the continued product multiplication. Once we have the continued product form in any base of a fraction, there is a way to convert the representation into any other base. This is true even when converting from a mixed radix, or mixed base format. This is investigated in Section 3**.** In Section 2, using the familiar mixed base measure of time as an example, the continued fraction notation is studied as a way of representing mixed radix numbers. Using a continued product in mixed bases, Section 4 contains an explanation for the computation of a million binary digits of π. In Section 5, we use continued product notation to explain the pattern of repeating decimals described by a Numberphile video[1] and also extend this property of repeating decimals to any number base.

Section 2: Measures of Time and Mixed Radix

Most of us use mixed radix representation on a daily basis to express time intervals. Yet, the use of mixed radix in the form of continued products can be confusing. The use of mixed radix in the representation of time can be helpful in understanding how it is employed in the following discussion. Suppose that we represent time using a number like .2.3.4.5, where 2 is in weeks, 3 in days, 4 in hours and 5 in minutes. To represent 10,000 minutes using a number like this, first divide 10,000 by 60 to convert minutes to hours.

$$\frac{10000}{60} = 166\, hours, remainder\ 40\, minutes.$$

Ten thousand minutes becomes 166 hours and 40 minutes. Now form

$$\frac{166}{24} = 6\ days, remainder\ 22\ hours.$$

Six days does not add up to one week. So,

$$10{,}000\ minutes = .0.6.22.40\ or\ 0\ w, 6\ d, 22\ h, 40\ m\ ,$$

or 0 weeks, 6 days, 16 hours, and 40 minutes in mixed radix. We might place other numbers to the left of the first radix point such as months, or years, or decades. The other radix points come in handy as a separation for the different radix value numbers. We could continue toward the right by adding seconds and tenths of a second. It is just a made up notation. Alternatively:

$$10{,}000\ minutes = 6 + \frac{1}{24}(22 + \frac{1}{60}(40))\ days.$$

Ten thousand minutes can be written as a decimal number or a mixed radix product on the right side that gives a number of days:

$$(6.944)_{10} = 6 + \frac{1}{24}(22 + \frac{1}{60}(40)). \qquad (3)$$

When working from the mixed radix form back to the decimal form work from right to left. First, change 40 minutes into hours:

$$\frac{40}{60} = 0.66667,$$

Then total hours:

$$22 + 0.6667 = 22.6667.$$

Then convert hours to days and total:

$$6 + \frac{22.6667}{24} = 6.944.$$

When working with these nested products, work from the least significant digit leftwards toward the most significant digit.

Section 3: Continued Products to Convert from Base to Base

In this section we discuss how to convert from a fraction from base 10 to any base using the continued product notation. This is important for the implementation of the Spigot algorithm to be discussed in the next section. For example, consider converting the decimal 0.65625 to base 2 or binary. The continued product form is shown below:

$$(0.65625)_{10} = \frac{1}{10}\left(6 + \frac{1}{10}\left(5 + \frac{1}{10}\left(6 + \frac{1}{10}\left(2 + \frac{1}{10}(5)\right)\right)\right)\right).$$

The factors 1/10 indicate every term in the number is in decimal. Let the binary version of the number be

$$0\,.\,b_1b_2b_3\ldots$$

To find, at once, the first two binary fraction digits multiply the base 10 number by $4 = 2^2$. This causes the first two binary digits to the right of the radix point to move to the left of the radix point. These numbers can be pulled off using an integer or floor operation:

$$(4 \cdot 0.65625)_{10} = 4 \cdot \frac{1}{10}\left(6 + \frac{1}{10}\left(5 + \frac{1}{10}\left(6 + \frac{1}{10}\left(2 + \frac{1}{10}(5)\right)\right)\right)\right);$$
$$(4 \cdot 0.65625)_{10} = \frac{1}{10}\left(24 + \frac{1}{10}\left(20 + \frac{1}{10}\left(24 + \frac{1}{10}\left(8 + \frac{1}{10}(20)\right)\right)\right)\right).$$

Go to the right of the expression. Twenty divided by ten is two with remainder zero, *i.e.,* 20/10 = 2 R 0. The expression becomes:

$$(4 \cdot 0.65625)_{10} = \frac{1}{10}\left(24 + \frac{1}{10}\left(20 + \frac{1}{10}\left(24 + \frac{1}{10}\left(8 + 2 + \frac{1}{10}(0)\right)\right)\right)\right).$$

So, adding 8+2:

$$(4 \cdot 0.65625)_{10} = \frac{1}{10}\left(24 + \frac{1}{10}\left(20 + \frac{1}{10}\left(24 + \frac{1}{10}\left(10 + \frac{1}{10}(0)\right)\right)\right)\right).$$

Continue in the same way:

$$(4 \cdot 0.65625)_{10} = \frac{1}{10}\left(24 + \frac{1}{10}\left(20 + \frac{1}{10}\left(25 + \frac{1}{10}\left(0 + \frac{1}{10}(0)\right)\right)\right)\right),$$

and $$(4 \cdot 0.65625)_{10} = \frac{1}{10}\left(24 + \frac{1}{10}\left(22 + \frac{1}{10}\left(5 + \frac{1}{10}\left(0 + \frac{1}{10}(0)\right)\right)\right)\right).$$

$$(4 \cdot 0.65625)_{10} = \frac{1}{10}\left(26 + \frac{1}{10}\left(2 + \frac{1}{10}\left(5 + \frac{1}{10}\left(0 + \frac{1}{10}(0)\right)\right)\right)\right),$$

or $$(4 \cdot 0.65625)_{10} = 2 + \frac{1}{10}\left(6 + \frac{1}{10}\left(2 + \frac{1}{10}\left(5 + \frac{1}{10}\left(0 + \frac{1}{10}(0)\right)\right)\right)\right).$$

While we are attempting to convert the number into a base 2 representation, the calculations are still done as decimals. So the digit 2 in the above expression is still a decimal. Convert the decimal two into a binary two. So, the first two digits are $(2)_{10} = (10)_2$. Multiply this by 1/4 to undo the initial multiplication by 4. To multiply a base 2 representation by $(1/4)_{10}$ simply move the binary point to the left by two places. The first two binary digits of the fraction are then $\frac{1}{4}(10)_2 = (0.10)_2$ To find the next two binary digits, multiply the remaining fraction term again by four.

$$\begin{aligned} 4 \cdot \frac{1}{10}\left(6 + \frac{1}{10}\left(2 + \frac{1}{10}\left(5 + \frac{1}{10}\left(0 + \frac{1}{10}(0)\right)\right)\right)\right) \\ = \frac{1}{10}\left(24 + \frac{1}{10}\left(8 + \frac{1}{10}(20)\right)\right) \\ = \frac{1}{10}\left(24 + \frac{1}{10}\left(10 + \frac{1}{10}(0)\right)\right) \\ = \frac{1}{10}(25) = 2 + \frac{5}{10} = (10.1)_2. \end{aligned}$$

To undo the twice multiplications by 4, multiply by 1/(16) by moving the binary point to the left four spaces to yield $(0.00101)_2$. With this number, combine with the first two digits $(0.10)_2$ computed previously to produce the final result.

$$(0.65625)_{10} = \frac{1}{10}\left(6 + \frac{1}{10}\left(5 + \frac{1}{10}\left(6 + \frac{1}{10}\left(2 + \frac{1}{10}(5)\right)\right)\right)\right) = (0.10101)_2.$$

This may seem like a cumbersome procedure for converting a decimal fraction into a binary fraction, but the process is central to the Spigot algorithm described in the following section when our continued product extends to an infinite number of terms.

Section 4: Computing One Million Digits of π in Binary Using the Spigot Algorithm

Our math club has previously published a paper in SIAM SIURO on novel ways to compute the value for π [3]. This raised an open question concerning the statistical properties of the digits for π. We decided to compute the digits of π in base 2. Statistically, we wonder if the digits can be distinguished as being different from a random set of independent binary digits having a Bernoulli distribution. We have chosen to investigate how the continued fraction notation and radix arithmetic can be used for the computation of π using the mixed radix format called the Spigot algorithm by Rabinowitz [2]. Rabinowitz developed the Spigot algorithm and in his paper shows how to compute digits of π in base 10. We have reprogrammed the Spigot into an Excel spreadsheet as well as into a compact Java program. Both compute digits of π in binary rather than decimal as Rabinowitz did. The reason for binary digits is so that we may investigate the statistical properties of the digits of π as Bernoulli random variables. (Such statistical analysis may be a topic for future investigation but is out of the scope of this paper.) First, consider a series that sums to π/2.[2]

$$\frac{\pi}{2} = \sum_{n=0}^{+\infty} \frac{n!}{(2n+1)!!} = 1 + \frac{1}{3} + \frac{1 \cdot 2}{3 \cdot 5} + \frac{1 \cdot 2 \cdot 3}{3 \cdot 5 \cdot 7} + \dots$$
$$= 1 + \frac{1}{3}\left(1 + \frac{2}{5}\left(1 + \frac{3}{7}\left(1 + \frac{4}{9}\left(1 + \frac{5}{11}(1 + \dots)\right)\right)\right)\right).$$

Multiply both sides by 2:

$$\pi = 2 + \frac{1}{3}\left(2 + \frac{2}{5}\left(2 + \frac{3}{7}\left(2 + \frac{4}{9}\left(2 + \frac{5}{11}(2 + \dots)\right)\right)\right)\right).$$

The denominators 3, 5, 7, … indicate that this can be viewed as a continued product with each digit representing a different base: base 3, base 5, base 7, and so on. See Appendix B for how this continued product calculates π. The Spigot algorithm permits the calculation of π to arbitrary precision independent of the computer processor's calculation precision. All computation is done using integer arithmetic. The reason for this lies at the heart of the calculations described in the introduction. Calculate digits one at a time as we slide them into the units position through multiplication by the base term *b*. After calculation, each digit is subtracted as in equation (1). After a digit is calculated, it is no longer needed for additional calculations. We then use the new continued product terms to calculate the next digit. To

begin, the units digit for π, which is 3, is subtracted, leaving the fractional part of π shown in equation (4). While the digits are decimal representations, they must be converted to the new base one at a time as they slide off. The decimal $(3)_{10} = (11)_2$. Consider

$$\pi - 3 = -1 + \frac{1}{3}\left(2 + \frac{2}{5}\left(2 + \frac{3}{7}\left(2 + \frac{4}{9}\left(2 + \frac{5}{11}(2 + \ldots)\right)\right)\right)\right). \quad (4)$$

The trick in the Spigot is in how the calculations are performed when multiplying by the base b, which in our case is $b = 2$. To pull the next binary digit out of the fraction multiply both sides by 2, and then grab the units digit of that product. It should be either a zero or a one.

$$2(\pi - 3) = -2 + \frac{1}{3}\left(4 + \frac{2}{5}\left(4 + \frac{3}{7}\left(4 + \frac{4}{9}\left(4 + \frac{5}{11}(4 + \ldots)\right)\right)\right)\right).$$

However, instead of one digit at a time do nine digits at a time. Instead of multiplying each side of equation (4) by 2, multiply by $2^9 = 512$ called SCALE in Figure 1. This will give us the following number that must be converted into binary.

$$512(\pi - 3) = -512 + \frac{1}{3}\left(1024 + \frac{2}{5}\left(1024 + \frac{3}{7}\left(1024 + \frac{4}{9}\left(1024 + \frac{5}{11}(1024 + 888)\right)\right)\right)\right). \quad (5)$$

The term 888 at the right end is the accumulated sum of the unseen terms to the right of the continued product. We have pre-calculated this and it is only an approximation. Figure 1 shows a section of the spreadsheet that does the calculation in order to help track the computations of the Spigot. The computation begins on the far right of the continued product. The terms of the continued product get smaller and smaller moving to the right due to the multiplication by the fractions (1/3), (2/5), …, (5/11). Each term is less than one-half, so the

	A	B	C	D	E	F	G	H	I	J
1			0	1	2	3	4	5	6	7
2	Below is SCALE		1	3	5	7	9	11	13	15
3	512		2	2	2	2	2	2	2	2
4	row3 * A3		1024	1024	1024	1024	1024	1024	1024	1024
5	Carry	000000011	583	726	795	836	865	888	903	912
6	sum		1607	1750	1819	1860	1889	1912	1927	1936
7	remainder		71	1	4	5	8	9	3	1
8	row7*A3		36352	512	2048	2560	4096	4608	1536	512

Figure 1: Section of Excel spreadsheet that does some computation

terms drop off faster than $(1/2)^n$. The best way to understand the operation of the binary Spigot is to review the spreadsheets calculations in detail. The first values to consider above are in column H in Figure 1. Note that rows 1 and 2 contain the numerator and denominator for the fractions 1/3, 2/5, 3/7, …. The repeated terms $1024 = (512 \cdot 2)$ are in row 4. In cell B5 is the binary number `11', which is the whole number three, the integer component of π.

The analysis begins with the numbers in column H. Calculations move from right to left and then down across the spreadsheet. The process can be gleaned by reading the cell formulas in the spreadsheet and carries through the calculations beginning with equation (5). Add 1024 and 888 to get the sum 1912 to be multiplied by (5/11). Don't multiply to get a decimal, but rather:

$$\frac{5}{11}(1912) = 5\left(\frac{1}{11}(1912)\right) = 5\left(173 + \frac{9}{11}\right) = 865 + \frac{5}{11}(9).$$

The term 865 is added to 1024 to give 1889. The remainder 9 stays in place for later calculations. In a similar fashion moving to the left in equation (5):

$$\frac{4}{9}(1889) = 836 + \frac{4}{9}(8)\,.$$

Equation (5) can now be expressed as

$$512(\pi - 3) = -512 + \frac{1}{3}\left(1024 + \frac{2}{5}\left(1024 + \frac{3}{7}\left(1024 + 836 + \frac{4}{9}\left(8 + \frac{5}{11}(9 + \ldots)\right)\right)\right)\right). \tag{6}$$

The process continues doing division with remainders, *i.e.*, Euclidean division, rather than with decimals.

The calculations in the Spigot algorithm proceed in a fashion similar to the mixed radix calculations for time in Section 2. All the numbers in Excel are naturally represented as base 10, but the calculations are nevertheless correct. When the final steps are displayed in column B. The decimal number must be converted to binary by use of the DEC2BIN function in Excel. Finally, to compute the next nine binary digits, the process begins again by using the

row with the remainders in the same way the row of twos was used to begin for the first nine digits.

When the next set of nine binary digits are calculated, the remainders of the first set of calculations are multiplied by SCALE = 512, like the row of 2s in the first set of calculations. Then those remainders are used for the next set of digits and so on. While the spreadsheet does nine binary digits at a time. The Java program in the Appendix does 13 digits at a time. Running the program using native Java on a MacBook, one million binary digits of π are computed overnight.

Section 5: Continued Product for Periodic Decimals

Numberphile is a website that illustrates interesting properties in number theory. The video referenced demonstrates interesting properties of the decimal expansions for $(1/9)^2$, $(1/99)^2$, $(1/999)^2$ and so on. The repeating decimal for $(1/9)^2$ is

$$\left(\frac{1}{9}\right)^2 = .\overline{012345679},$$

and

$$\left(\frac{1}{99}\right)^2 = 0.\overline{000102030405060708090001 02...9799}$$

Note that these repeating decimals include all digits sequentially except for 8 in the first case and in the second case 8 and 98. We will show that a similar pattern appears for number base systems other than decimal numbers. For example, For base *b* numbers:

$$\frac{1}{(b-1)^2} = .0123\ldots(b-3)(b-1)\,012\ldots,$$

where the digits go from '0', '1',… up to (*b* - 3). The digit $(b-2)$ is skipped and the next digit is (*b* - 3) in the radix expansion.

First, consider the decimal expansion for $(1/9)^2$ derived as a recursion:

$$\left(\frac{1}{9}\right)^2 = \left(\frac{1}{10^2}\right)\left(\frac{10}{9}\right)\left(\frac{10}{9}\right),$$

where $$\frac{10}{9} = 1+\frac{1}{9} = 1+\frac{1}{10}\frac{10}{9} = 1+\frac{1}{10}\left(1+\frac{1}{9}\right).$$

Continuing the pattern:

$$\frac{10}{9} = 1+\frac{1}{10}\left(1+\frac{1}{10}\left(1+\frac{1}{10}\left(1+\frac{1}{10}\left(\ldots\right)\right)\right)\right).$$

$$= 1.11111...$$

This means $(1/9)^2 = (0.1111...)^2$ as a decimal.

The normal long format multiplication of these two numbers is shown in Figure 2. The multiplication gives larger and larger stacks of ones as it moves to the right. The key elements are the carried terms as we sum from right to left. The carried terms are added to the column sums as happens with normal addition. As the columns and carries are added what remains at the bottom of each added column are the units digits found in a row at the very bottom of the figure. As noted by Numberphile, the units digits contain every digit from 0 to 9 excluding 8. This is as a result of the carries in the column additions.

0.1 1 1 1 1 1

x 0.1 1 1 1 1 1

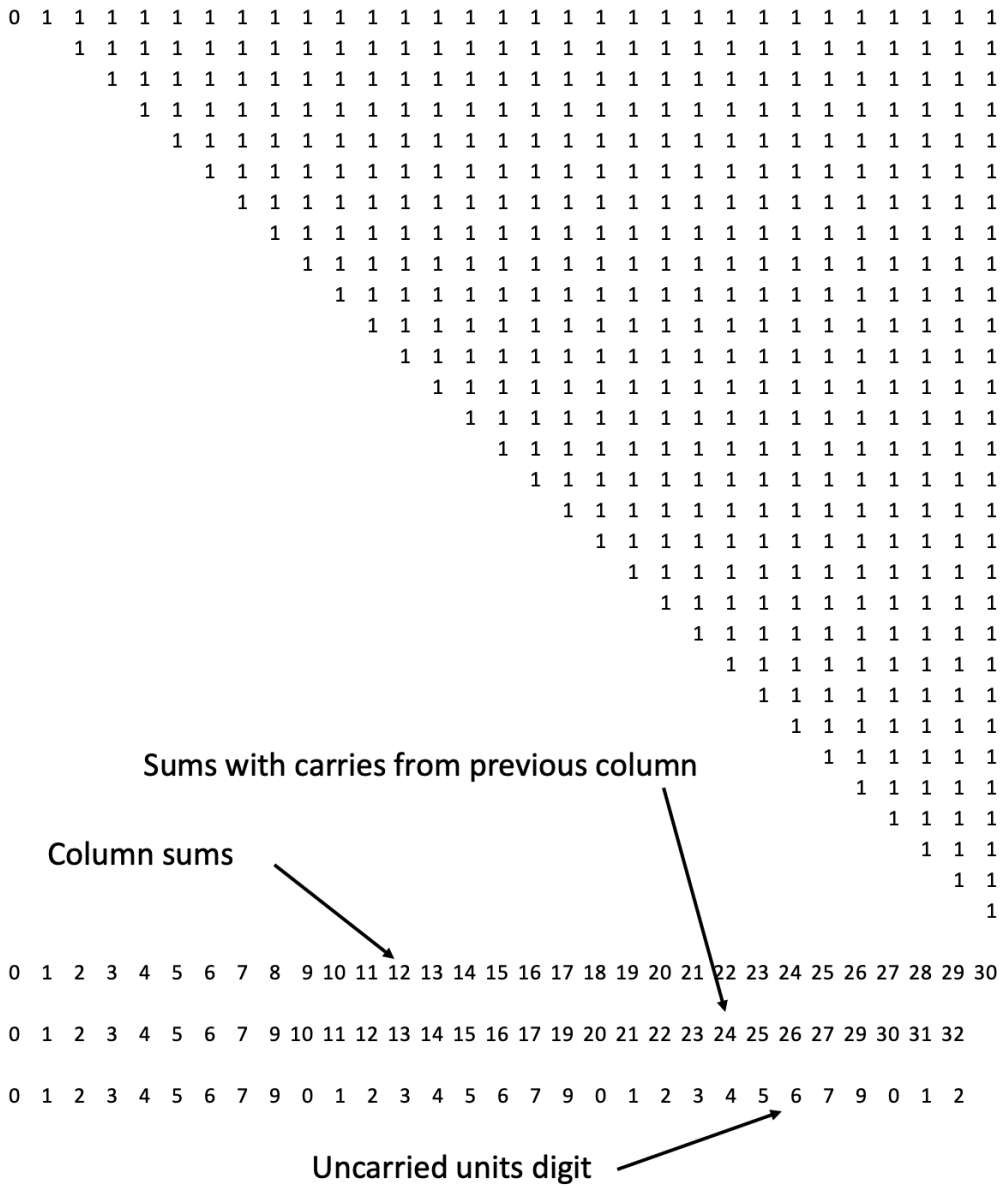

Figure 2: Long Form Multiplication for (0.11111…)²

Figure 2 shows what is happening, but is not a formal proof. A better proof comes from doing long division.

$$
\begin{array}{l}
\qquad\qquad 1+\frac{2}{10}+\frac{3}{10^2}+\frac{4}{10^3}+\frac{5}{10^4}+\frac{6}{10^5}+\frac{7}{10^6}+\frac{8}{10^7}+\frac{9}{10^8}+\frac{1}{10^8} \\
\hline
(10^2-20+1)\,)\,10^2+\ 0+\ 0+\ 0+\ 0+\ 0+\ 0+\ 0+\ 0+\ 0... \\
\quad -(10^2-20+1) \\
\hline
\quad 20-1 \\
\quad -(20-4+\frac{2}{10}) \\
\hline
\quad 3-\frac{2}{10} \\
\quad -(3-\frac{6}{10}+\frac{3}{10^2}) \\
\hline
\quad \frac{4}{10}-\frac{3}{10^2} \\
\quad -(\frac{4}{10}-\frac{8}{10^2}+\frac{4}{10^3}) \\
\hline
\quad \frac{5}{10^2}-\frac{4}{10^3} \\
\quad -(\frac{5}{10^2}-\frac{10}{10^3}+\frac{5}{10^4}) \\
\hline
\quad \frac{6}{10^3}-\frac{5}{10^4} \\
\quad -(\frac{6}{10^3}-\frac{12}{10^4}+\frac{6}{10^5}) \\
\hline
\quad \frac{7}{10^4}-\frac{6}{10^5} \\
\quad -(\frac{7}{10^4}-\frac{14}{10^5}+\frac{7}{10^6}) \\
\hline
\quad \frac{8}{10^5}-\frac{7}{10^6} \\
\quad -(\frac{8}{10^5}-\frac{16}{10^6}+\frac{8}{10^7}) \\
\hline
\quad \frac{9}{10^6}-\frac{8}{10^7} \\
\quad -(\frac{9}{10^6}-\frac{18}{10^7}+\frac{9}{10^8}) \\
\hline
\quad \frac{10}{10^7}-\frac{9}{10^8} \\
\quad -(\frac{10}{10^7}-\frac{20}{10^8}+\frac{10}{10^9}) \\
\hline
\quad \frac{11}{10^8}-\frac{10}{10^9}
\end{array}
$$

The bottom term is a repeat of the top

Figure 3: Long division until the numerals repeat the terms in the dividend;
This last term $= \frac{11}{10^8} - \frac{10}{10^9} = \frac{1}{10^9}(10^2 + 0)$*, begins repeating the terms in the original dividend indicating that the quotient will also repeat.*

Consider

$$\frac{1}{81} = \frac{1}{(10-1)(10-1)} = \frac{1}{10^2 - 20 + 1} = \frac{1}{10^2}\left(\frac{10^2}{10^2 - 20 + 1}\right).$$

Write this out in a fashion similar to polynomial division as seen in Figure 3. In the quotient term in Figure 3, the '1' is a carry and turns the '9' into a ten which then carries to the '8' making a '9'. The numeral '8' disappears from the quotient. Because the terms in the dividend repeat as the long division progresses, then the quotient should repeat too. The term with the numeral 8 always disappears. This is more of a proof. At the bottom in Figure 3 the last term is equal to $\frac{11}{10^8} - \frac{10}{10^9} = \frac{1}{10^9}(10^2 + 0)$, which is the same as the original dividend except for the multiplying factor of $\frac{1}{10^9}$. This means that the terms in the quotient will repeat except for a factor of $\frac{1}{10^9}$. The numerators in the quotient will repeat. Consider the last three terms in the quotient:

$$\frac{8}{10^7} + \frac{9}{10^8} + \frac{1}{10^8} = \frac{9}{10^7}.$$

This division method can be used to prove that for any base, *b,* the general result in equation (7). For a base b number, the largest digit is $b_1 = b - 1$. Suppose we have a number where all the digits are equal to b_1 as shown below. Then using division it can be shown

$$\frac{1}{\underbrace{(b_1 b_1 \ldots b_1)^2}_{p \text{ digits}}} = \sum_{n=1}^{\infty} \frac{n}{b^{p(n+1)}}. \tag{7}$$

The first step is to realize the p-digit number with every digit equal to b_1 satisfies $(b_1 b_1 \ldots b_1) = (b^p - 1)$. Consequently,

$$\frac{1}{\underbrace{(b_1 b_1 \ldots b_1)^2}_{p \text{ digits}}} = \frac{1}{(b^p-1)^2} = \left(\frac{1}{b^{2p}}\right)\frac{b^{2p}}{b^{2p}-2b^p+1}. \tag{8}$$

Long division gives

$$(b^{2p}-2b^p+1)\overline{)\,b^{2p}+0+0+0+0+0+0+0+0+0...}$$
$$1+\frac{2}{b^p}+\frac{3}{b^{2p}}+\frac{4}{b^{3p}}+\ldots$$

Combine this with the $\left(\frac{1}{b^{2p}}\right)$ from equation (8) yields equation (7):

$$\frac{1}{\underbrace{(b_1 b_1 \ldots b_1)^2}_{p \text{ digits}}} = \left(\frac{1}{b^{2p}}\right)\frac{b^{2p}}{b^{2p}-2b^p+1} = \left(\frac{1}{b^{2p}}+\frac{2}{b^{3p}}+\frac{3}{b^{4p}}+\frac{4}{b^{5p}}+\ldots\right).$$

Now some examples to illustrate the final result.

Example 1: Let the base $b = 7$ and $p = 1$:

$$\begin{aligned}\frac{1}{6^2} &= \ldots\frac{4}{7^5}+\frac{5}{7^6}+\frac{6}{7^7}+\frac{7}{7^8}+\ldots = \ldots\frac{4}{7^5}+\frac{5}{7^6}+\frac{6}{7^7}+\frac{1}{7^7}+\ldots\\ &= \ldots\frac{4}{7^5}+\frac{5}{7^6}+\frac{7}{7^7}\ldots\\ &= \ldots\frac{4}{7^5}+\frac{6}{7^6}+\ldots\end{aligned}$$

So, $= (0.012346012346\ldots)_7$

and the base 7 representation skips the digit 5.

Example2: Return to $b = 10$, and let $p = 2$:

$$\frac{1}{99^2} = \sum_{n=1}^{\infty}\frac{n}{10^{2n+2}}.$$

Consider some terms in the middle of the series:

$$\ldots\frac{97}{10^{196}}+\frac{98}{10^{198}}+\frac{99}{10^{200}}+\frac{100}{10^{202}}+\frac{101}{10^{204}}+\frac{102}{10^{206}}+\frac{103}{10^{208}}\ldots \qquad \text{(A)}$$

$$\frac{100}{10^{202}} = \frac{1}{10^{200}},$$

and

$$\frac{101}{10^{204}} = \frac{1}{10^{202}} + \frac{1}{10^{204}}$$

and

$$\frac{102}{10^{206}} = \frac{1}{10^{204}} + \frac{2}{10^{206}}.$$

Substitute these values back into equation (A) and consolidate like terms:

$$\ldots + \frac{97}{10^{196}} + \frac{99}{10^{198}} + \frac{0}{10^{199}} + \frac{0}{10^{200}} + \frac{1}{10^{202}} + \frac{2}{10^{204}} + \ldots$$

There are additional terms on the right that will carry down into these terms

In the general case of base-b, p-digits, and when $n \approx b^p$, the carries work as in these examples. Consider a collection of terms in the series located around where the first carries occurs:

$$\ldots \frac{b^p - 3}{b^{p(b^p-2)}} + \frac{b^p - 2}{b^{p(b^p-1)}} + \frac{b^p - 1}{b^{p(b^p)}} + \frac{b^p}{b^{p(b^p+1)}} + \frac{b^p + 1}{b^{p(b^p+2)}} + \frac{b^p + 2}{b^{p(b^p+3)}} \ldots$$

The numerator b^p - 2 is changed by a carry to b^p - 1 eliminating the digit b^p - 2 in the radix expansion for the full sum found in equation (7).

The two terms on the right can be summed and simplified using carries:

$$\frac{b^p + 1}{b^{p(b^p+2)}} + \frac{b^p + 2}{b^{p(b^p+3)}} = \frac{1}{b^{p(b^p+1)}} + \frac{2}{b^{p(b^p+2)}} + \frac{2}{b^{p(b^p+3)}}.$$

Also using carries, the three terms in the middle combine:

$$\frac{b^p - 2}{b^{p(b^p-1)}} + \frac{b^p - 1}{b^{p(b^p)}} + \frac{b^p}{b^{p(b^p+1)}} = \frac{b^p - 1}{b^{p(b^p-1)}} + \frac{0}{b^{p(b^p)}}.$$

The combination of all the terms produces

$$\ldots \frac{b^p - 3}{b^{p(b^p-2)}} + \frac{b^p - 2}{b^{p(b^p-1)}} + \frac{b^p - 1}{b^{p(b^p)}} + \frac{b^p}{b^{p(b^p+1)}} + \frac{b^p + 1}{b^{p(b^p+2)}} + \frac{b^p + 2}{b^{p(b^p+3)}} \ldots$$

$$= \ldots \frac{b^p - 3}{b^{p(b^p-2)}} + \frac{b^p - 1}{b^{p(b^p-1)}} + \frac{0}{b^{p(b^p)}} + \frac{1}{b^{p(b^p+1)}} + \frac{2}{b^{p(b^p+2)}} + \frac{2}{b^{p(b^p+3)}} \ldots$$

The final term here on the right does not include the carries from further down the sum. By plugging in the correct values for p and b, the results in the previous two examples also derive directly from this expression.

Section 6: Summary

To begin, we investigate the conversation of a fraction from one number base to another by use of a continued product notation and/or long division. It is possible that a radix expansion does not terminate in one base system, or the other, or both. This means the process of base conversion may need to take into account carries in the multiplication process. Our investigation was motivated by a video on the Numberphile website about repeating decimal expansions of certain fractions that repeat all digits sequentially except for the digit 8 , leading back to an investigation of the number π. We have developed a generalization, with examples, of the Numberphile result from base 10 to arbitrary base *b*. We believe this is original.

The Spigot algorithm of Rabinowitz and Wagon uses such a continued product in the computation of a seemingly arbitrary number of digits for π. They use an infinite mixed base continued product to help them in their computation. A key element of their algorithm is to use Euclidean division with remainders in the mixed base system rather than decimal division that converts everything to base 10. Instead of computing the digits of π in base 10 as do Rabinowitz and Wagon, we implement base conversion to generate π in base 2. This links our discussion of base conversion to the Spigot algorithm. We believe our binary π computation as found in the Excel spreadsheet and Java code are original.

References

1. Numberphile website https://youtu.be/daro6K6mym8, accessed January 20, 2019.
2. S. RABINOWITZ AND S. WAGON, A Spigot Algorithm for the Digits of Pi, Amer. Math. Monthly 102(1995), pp. 195-203.
3. J.C. FAIR, et. al, Insights into the Computation for π, SIAM SIURO, 8(2015), pp. 269-285, also available online from http://evoq-eval.siam.org/Portals/0/Publications/SIURO/Vol8/Insights_into_the_Computation%20for_pi.pdf?ver=2018-04-06-152007-240.

Appendix A

Matlab Script to Convert from Decimal to base B.

```
clc
clear
format long

% This script converts a base 10 number to base B. This is done in two steps,
% for the integer part and one for the fraction. The two steps are
% implemented as function subroutines. To convert the integer part we keep
% dividing the number by B to shift the digits to the units place. The
% remainder function pulls off the integer digits after each divide.
% To convert the fraction part, we keep multiplying the fraction part by B
% and use the floor function to pull of the units digit (from '0' to
% 'B-1'). A radix point is added to the beginning of the fraction part and
% the two strings, integer and fraction, are concatenated into a long
% string to produce the final output.
%
%
% It is easy to modify this script to convert to any number base. Changing to
% something like hexadecimal presents an added complication because we need to
% define the characters for the integers from 10 to 15. The script does not
% convert to a base B>16 but could be easily modified.
B = 7; % Base to which to convert
dec = 4095.999755859375 % Decimal number
dec = 0.65625
dec = 0.027777777777
Limit = 16; % limits the number of base b fraction digits
decfloor = floor(dec) % integer part of number
dec_fr = dec - decfloor% fraction part
binfrac = fraction(Limit,dec_fr,B);% computes base b fraction
binint = integer(decfloor,B);% computes base b integer
```

```
binnum = strcat(binint,binfrac) % concatonates the strings binint and binfrac

function [binfrac] = fraction(Limit,dec_fr,B)% fraction part converted
bin = blanks(Limit);% dimensions and fills the array bin() with blank spaces
i = 1;
q = floor(dec_fr*B);% multiply by base=B to slide radix point to the right
                % using floor to pull off the units digit into q
      Hq = hexadecimal(q); % If needed converts digits to hexadecimal
                        % for bases between 10 and 16.
   bin(i) = num2str(Hq);% puts ascii code for the character q in array bin()
   while i <= Limit-1  % computes the remaining digits one at a time
   dec_fr = B * dec_fr - q;% pulls off the integer part leaving the fractional part
         i = i + 1;
   q = floor(dec_fr*B);%multiplies by 2 creating a new units digit
      Hq = hexadecimal(q);

   bin(i) = num2str(Hq);% puts ascii code for character q units digit
                  % into the the array bin()
   end
binfrac = strcat('.',bin); % adds radix point in front of string
end

function [binint] = integer(dec_nr,B)% integer part converted
i = 1;
q = floor(dec_nr/B);% divide the integer by 2 and drop the fraction part
r = rem(dec_nr, B);% pull off the units digit by dividing by 2 and getting
                % the remainder
Hr = hexadecimal(r);% convert r to hexadecimal if needed
bin(i) = num2str(Hr(i));% convert the units digit to a string and put it into
                  % the array b()
while B <= q % now we pull off the digits one at a time
   dec_nr = q;% new reduced integer
   i = i + 1;
```

```
   q = floor(dec_nr/B);% divide by 2 and pull of units digit
   r = rem(dec_nr, B); % into r
      Hr = hexadecimal(r);
   bin(i) = num2str(Hr); % convert r into a character and put into array bin()
end
      Hq = hexadecimal(q);
bin(i + 1) = num2str(Hq); % Hq now contains the final remaining digit
binint = fliplr(bin); % the digits need to be put into reverse order
end

function [Hex] = hexadecimal(x)
% As long as the digit value x < 10, the output Hex remains this digit.
% Otherwise it is converted to the hexideicmal letter representing the
% number
Hex = x;% places digit into output variable Hex and then converts if needed
if x == 10;Hex='A';elseif x == 11; Hex='B';elseif x == 12; Hex='C';
      elseif x == 13; Hex='D';elseif x == 14; Hex='E';elseif x == 15; Hex='F';
end
end
```

Appendix B

Computation Using Continued Product Beginning with Least Significant Terms

Begin with the innermost term and work out:

$$\pi = 2 + \frac{1}{3}\left(2 + \frac{2}{5}\left(2 + \frac{3}{7}\left(2 + \frac{4}{9}\left(2 + \frac{5}{11}(2 + \ldots)\right)\right)\right)\right).$$

$$\pi = 2 + \frac{1}{3}\left(2 + \frac{2}{5}\left(2 + \frac{3}{7}\left(2 + \frac{4}{9 \cdot 11}\left(2 \cdot 11 + 5 \cdot (2 + \ldots)\right)\right)\right)\right).$$

Drop the (…) term as an approximation to give $5 \cdot 2$. So,

$$\pi \approx 2 + \frac{1}{3}\left(2 + \frac{2}{5}\left(2 + \frac{3}{7}\left(2 + \frac{4}{9 \cdot 11}\left(32\right)\right)\right)\right).$$

$$\pi \approx 2 + \frac{1}{3}\left(2 + \frac{2}{5}\left(2 + \frac{3}{7 \cdot 9 \cdot 11}\left(2 \cdot 9 \cdot 11 + 4 \cdot 32\right)\right)\right).$$

$$\pi \approx 2 + \frac{1}{3}\left(2 + \frac{2}{5}\left(2 + \frac{3}{7 \cdot 9 \cdot 11}(326)\right)\right).$$

$$\pi \approx 2 + \frac{1}{3}\left(2 + \frac{2}{5 \cdot 7 \cdot 9 \cdot 11}\left(2 \cdot 7 \cdot 9 \cdot 11 + 3 \cdot 326\right)\right).$$

$$\pi \approx 2 + \frac{1}{3}\left(2 + \frac{2}{5 \cdot 7 \cdot 9 \cdot 11}(2364)\right).$$

$$\pi \approx + \frac{1}{3 \cdot 5 \cdot 7 \cdot 9 \cdot 11}\left(2 \cdot 5 \cdot 7 \cdot 9 \cdot 11 + 2 \cdot 2364\right).$$

$$\approx 3.1215$$

Appendix C

Spigot Algorithm for Computing Digits of π in base 2

Running it as is, this program listing computes 200 digits of π. Near the bottom of the listing the integer n is set equal to 200. Simply change this value to 1,000,000 to compute a million binary digits of π.

```
public class PiBinaryDigits {
   private static final int SCALE = 8192;
   private static final int ARRINIT = 4096;

   public static String pi_digits(int digits){
                System.out.println("digits = "+digits+"\n");
      StringBuffer pi = new StringBuffer();
      int[] arr = new int[digits + 1];
      int carry = 0;
//************************************************************
      for (int i = 0; i <= digits; ++i)  // Initialize arr[i]
         arr[i] = ARRINIT;
//************************************************************
      for (int i = digits; i > 0; i-= 14) {

         int sum = 0;
         for (int j = i; j > 0; --j) { // note the value j runs backwards
            sum = sum * j + SCALE * arr[j];
```

```
            arr[j] = sum % (j * 2 - 1);
            sum /= j * 2 - 1;
        }
// These are alternate alternate ways to execute the buffer storage of binary digits
// as well as to Print data if desired.
    //     pi.append(String.format("%9s",
    //          Integer.toBinaryString(carry + sum / SCALE)).replace(" ","0"));
           pi.append(String.format("%13s",
                Integer.toString((carry + sum / SCALE),2)).replace(" ","0"));
    //     System.out.println( String.format("%13s",
    //          Integer.toBinaryString(carry + sum / SCALE)).replace(" ","0"));
    //     System.out.println( String.format("%9s",
    //           Integer.toString((carry + sum / SCALE),2)).replace(" ","0"));

 // System.out.println(pi);
          carry = sum % SCALE;
      }
      return pi.toString();
   }

   public static void main(String[] args) {
      int n = args.length > 0 ? Integer.parseInt(args[0]) : 200;
      System.out.println(" n = "+n);
      System.out.println(PiBinaryDigits.pi_digits(n));
   }
}
```