Antplot: Visualising Long Binary Strings Using a Variation of Langton's Ant

Mark Agate
Waterlooville, Hampshire, UK
mark.agate@outlook.com

Abstract

Langton’s ant is an interesting example of cellular automata, displaying both chaotic and emergent behaviour. This paper presents a simple integer-based numeric expression which generates the trajectory of Langton’s ant on an infinite plane, and extends the technique so that binary strings of arbitrary length can be represented graphically. Patterns created by applying this technique to the mathematical constants e and pi are presented, together with those for a selection of long factorial and reciprocal values.

1. Introduction

In its simplest form, Langton’s ant moves across an infinite grid one square at a time, changing direction at every step according to the colour of the square it lands on. If the square is white, the ant changes the colour of the square to black and turns 90° to the right. If the square is black, the ant changes it to white and turns 90° to the left. Wikipedia has an animation which demonstrates the first few steps taken by the ant on an initially white grid, as well as an image of the trail left by the ant after 11000 steps [6].

The algorithm executed by the ant is quite a simple one, and it is not too surprising that it generates several symmetrical patterns for the first 475 hundred steps. However, the ant then traces a largely chaotic path up to the 9977th step, and thereafter it settles into a pattern which repeats every 104 steps. On an unbounded plane, this pattern repeats to infinity.

For the purposes of illustration, it is convenient to plot the ant trail on a non-white background, treating any squares which the ant has not yet visited as if they were white (thus when the ant lands on an unvisited square, it changes it to black and turns right). This allows the full extent of the ant trail to be seen; otherwise the squares left as white by the ant would be indistinguishable from the ones it never visited. Figures 1 to 3 show the trail left by Langton’s ant against a blue-grey background.

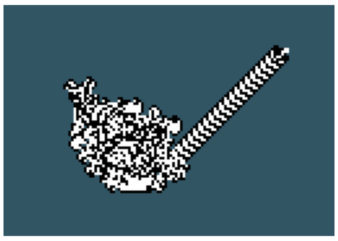
Figure 1: Langton's ant trail after 12000 steps

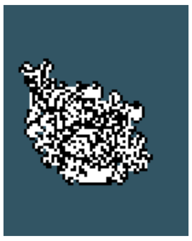
Figure 2: First 9977 steps of Langton's ant

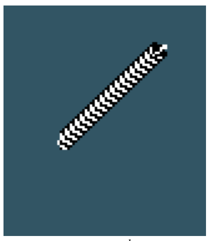
Figure 3: 9978th to 12000th steps of Langton's ant

C.G. Langton originally devised his Ant as a part of his work on artificial life [3]. There have been numerous extensions to the basic algorithm based on multiple colours, bounded grids, multiple ants colliding with each other, and similar automata known as Turmites, but this discussion will be confined to the path taken by a single black-and-white ant.

The behaviour of the individual ant is far from intuitive. It obeys simple rules, yet traces an apparently chaotic path, then suddenly adopts a repetitive pattern which it follows for the remainder of its track as far as the grid allows. Could any mathematical construct other than the algorithm for 2-dimensional movement of Langton’s ant produce identical behaviour?

Researcher Graham Medland has achieved just that by moving the ant on the surface of a torus instead of an infinite plane [4]. The sequential output of the algorithm is mapped to the grid using modulo arithmetic. This is similar to the way in which linear video memory addresses are sometimes mapped to a computer screen, i.e. going off any edge of the screen brings you back on the opposite edge. Interestingly, the expression which generates the ant trail takes the form of a wave equation. A script which implements the algorithm is available on Medland's website [5].

In this paper we explore a simpler mathematical expression for the path of the ant, expressed as a ratio of two integers. In an attempt to make sense of the result, the algorithm for Langton's ant is adapted so the large numbers involved can be visualised. This algorithm is then applied to a variety of other long numeric strings, with some interesting graphical results.

2. An Integer Expression for Langton's Ant

Take the first step of the ant as the most significant bit of a binary number; the second step as the next most significant, and so on. Where a pixel is turned black we set a 1, and where it is turned white we set a zero. The path of the ant thereby generates a string of 9977 bits with no obvious pattern, followed by a repeating string of 104 bits which carries on to infinity. Any infinitely recurring pattern of digits represents a rational number; we should therefore be able to express the path of Langton's ant as a ratio of two integers.

Let the string of binary digits representing the path of the ant be given the symbol L. If we choose to put the binary point after the first 9977 steps, then the part of the ant trail shown in figure 2 will generate a 9977-bit integer, which we call L_i. The remainder of the ant trail, as shown in figure 3 (but extended to infinity) is a binary recurring fraction which we call L_f. So:

$$L = L_i + L_f \qquad \text{(Equation 1)}$$

Conveniently, the recurring pattern in L_f is 104 bits long, which is exactly 13 bytes. In hexadecimal, the sequence of bytes is:

0x4F279E5E87B785EF0BD30CF349

The value of the fraction which has this recurring pattern can be found by dividing the above number by the following hexadecimal value:

0xFFFFFFFFFFFFFFFFFFFFFFFFFF

Both these integers are less than 128 bits long, so it is easy to use an online calculator to obtain the prime factorization of each. By doing so, we find that the highest common factor of both integers is 5. Since we only need the ratio of the two integers to calculate L_f, we can divide each by 5 to get:

$$L_f = C / B \qquad \text{(Equation 2)}$$

Where:

C = 0x0FD4B9461B24B463025D68FD75

and

B = 0x33333333333333333333333333

Combining equations 1 and 2, we get:

$$L = L_i + (C / B)$$

This can be re-expressed as:

$$L = (B.L_i + C) / B$$

which we can write as:

$$L = A / B \qquad \text{(Equation 3)}$$

where:

$$A = B.L_i + C \qquad \text{(Equation 4)}$$

Equation 3 gives us an expression for the trail of Langton's ant as a ratio of two integers. The values of B and C given above have been used in equation 4 to give the following hexadecimal value for A:

```
631812FCF2C6A4B463025F9D8EC9530D6622F8EECA6FFBEE4D8B9CFDA4198639
A46F4EAD308E488FABD20FC4527738D23960890DFD7CEA96F9FCED173D9C3910
8B9C1241E66817021AA1FDE6167470B569FF3B1ED9BCCFE0CB95CB4E51CFE9C9
339D185834ECC6DB971B0702C6A6C11D4A9B69267E3869CB35B9654202CCCCD3
1A0575D2C7E1F294C457AC6C8C520013E28E958173CFF0CA14F938AF7660DF11
83874F4E564B4728A6EDF82B6FB7ABE9A86BB56783B35FC01DA63B870E48FFC4
3D0BC2F7050B62E70095272D35F1246195F33E82B6AAE223E1BCA6ECA85C6AB1
4D99BD5F9BEA7E0C5219A7EAAACD38798E6FD3DEC91AD272ACAC74C262AB9224
1ED7E8860CBC19A056CABE4A41C334DDBCA65E430CBED2962383E99BA3969034
EE5B1CD1869CB204886CB63A7B1E2C621CF61699CE13D73F7E39E05729E40FAC
F7D39DFFB79A302B43DC17BD8AC75FFFD1E40D0F86EA394892671991E3E85D87
B62AB8F8846D8E125A0872D3C8BEE020C3CA39DAB9610C27C5EB999F5D31F30A
5401191CDEA91821E9456AAE3104D26553E0BBD6AAE285414D7DFFC5D68D1286
E5AFFB9820D57328348B3D5422A02E2F62A02711BD24FB74A9044A111BDED8A6
2082BF2705598A1ECE6B8DF303B42687B6C3EF272F65A983063A16F96FF51669
16ED494668C145D128154C4E33B2501742C80EE2CDCEC57CE2D5EEF8497201DD
60040B17EB4BAC4D114408336F095AA164ABF0623C95CEA23EAA62BC2BF37C08
8437B201BB249BDF2BDE5574FADA2079B6007DA2D40CC1B375694916F7E3AFD6
0700AA240910DCA2E2CF5835BBDC5B1A4A020D8020573F8843B55ECD579A9B3B
8B0104291055BC76365D4BC9B492A2D793A154E7EE079508852DC8E34B0123C5
AC85C958B6DB4E523496886F57D350D76ACF0766BF803B4C770E2FAED1C7B55D
CC744BE33330773D29071ED9489A0602C7688EAFB6606CBCD5646C9EBF15F7F6
9C3FB238EEEA2A8DC9F8BA0D7A0C5999B228A883B53148964B5FD55D8F34E24F
AEA0AD513FFDA94CB9021E2B167C2841B8A0D1823C87C9CA81B638E8469D2C87
D0BA8C427AFCA85BA659DD23983A06D58E67CCF176B5110A28AF88E3A62F066D
13DD7BD990670F54D4415C7784FD6D21F93761CDEC1D42BE725D39CEE8BFC74F
7DDF26669D85B32581568416086D1BDCFD9126A9E21C077B93A9BDD447027D33
292A4642E108555411617E100DDB921BCEA5BB6943A086C5D2B5AF7F7449C094
9665A41679A7E8440E7750215F0BDDBE3B6FD337687959994820E94892789CAA
499ACD73082D7910E4B25ECB181475F0382FA616F34ED7A4047941F43D48CC10
6C950F4FF69F53CA401965F633396406ADF45AE8A6E7CE1FDF02AE145834EFD8
28B05B35FA40F907B8C2C548D7E389F4221F46B86150093E83F38E0CF990C9C3
96AF44B0A15862295DEC18BD91D7FDCC335B14E1C065A7AF1E3AB5A81D37B484
20162A81CE22DA7C1F33396BA3A4FC536BC28196E32D849992EC894314A31659
2D0D95C9B9E1ACCA2662926B98EC0A82BF807A7B6655F26694A729B4B9E0B58C
EE0A77C6DE58368658901 62EF25CE4FACF27706D3F5560544ECAC3FB8EB0A3F5
E62D6EDC5E36DCD2393ACD7581C048837246B0856A1F69858F93A6562777C492
77C9B593E16E0CB9F11A056CF5FFD60A0F99025CC342CC35908A232D85F06714
040AEBA58FC6D23D1AE300D64B8ADB037C01B1CA890E9BFBC6E6FAD77BAE4EA7
10DD52666883B155D962345D
```

This is an enormous integer (2520 hex digits) which makes for tedious reading. In decimal, its value is approximately $9.336064 \times 10^{3033}$. However, any patterns within its string of digits would be of interest. In the following sections, a graphical method is developed for verification of the value of A, and this same technique is then used to look for patterns within its value.

3. Verification of the Integer Expression for Langton's Ant

In order to verify that we have the correct values for A and B, we can use their values in equation 3, and see if the value we compute for L corresponds to the trail of Langton's ant. Doing this graphically requires a subtle but important modification to the plotting algorithm. Instead of controlling the ant direction and shading according to the previous colour of each square, we need to use the individual binary digits of an input value. If the input bit is 1, the ant shades the square black and turns 90° to the right. If the input bit is 0, the ant shades the square white and turns 90° to the left.

If we compute the value L = A / B, and feed the binary digits of L into the above algorithm, the output shown in figure 4 is obtained. The trail of Langton's ant is faithfully reproduced, with the length of the repeating pattern limited only by the arithmetic resolution to which the computation is performed.

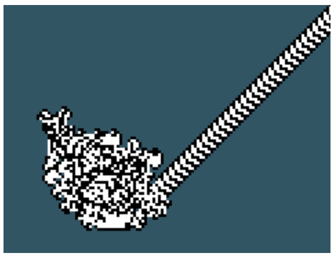
Figure 4: L = A / B

4. Visualisation of Binary Strings

The algorithm developed in section 3 takes a 1-dimensional binary string as input, and generates 2-dimensional graphical output. We call this output the antplot of the binary string. So far we have applied the algorithm to our calculated value of L, but what happens when we apply it to other values?

The obvious next choice of candidate for an antplot is the value of A. So far we have only viewed this as the long hexadecimal string in section 2. Using the antplot algorithm on the value gives the output shown in figure 5.

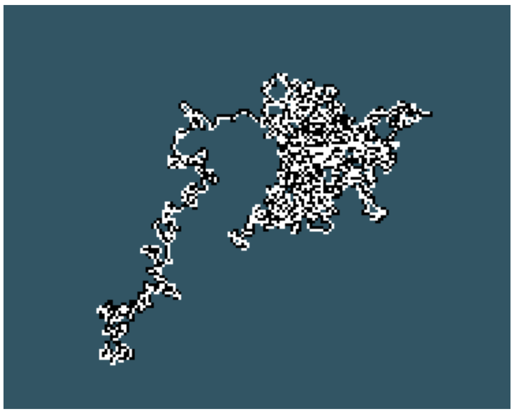
Figure 5: Antplot of A

What does this tell us about A? It is disappointing that there are no obvious patterns in the antplot; neither does the trail for A resemble the trail of Langton's ant. It is perhaps notable that there is a long narrow part of the trail leading to a patch which resembles a butterfly or flying squirrel, but antplots of numerous random binary strings have yielded similar forms. The tendency for observers to perceive objects in random patterns is known as pareidolia, as used to good effect in Rorschach inkblot tests. Further examples of antplots resembling familiar objects are presented in later sections.

For completeness, the antplot of of the value B is presented in figure 6. The value consists of the hex digit 3 (or the binary digits 0011) repeated 26 times, so the ant makes pairs of right and left turns, and shades the squares appropriately leaving the trail shown.

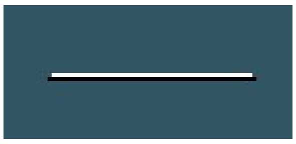
Figure 6: Antplot of B

5. Antplots of Factorials

In an attempt to apply the antplot algorithm to other finite-length numeric strings, software has been written to generate factorials of integers up to about 20000. Only a small fraction of these factorials have been plotted, and almost all of those that have appear as fairly amorphous blobs, similar to antplots of random strings. However, a few have given notable results.

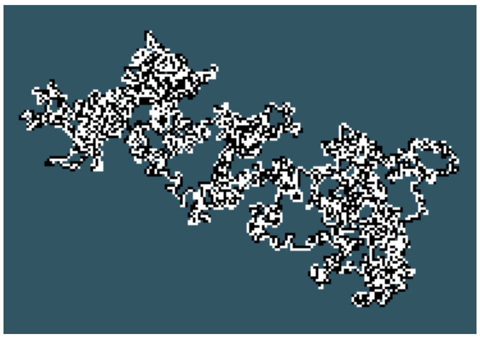
Figure 7: 2047! or 2048!

The software increments an integer value and produces the corresponding factorial plot each time a key is pressed. Whenever the value is one less than a power of 2, it is observed that the antplot of the next factorial is identical. For example, the antplot shown in figure 7 is obtained for the factorials of 2047 and 2048.

The reason is that 2048! is equal to the value 2047! multiplied by 2048. Multiplying any binary value by a power of 2 is equivalent to shifting its digits to the left. This demonstrates that the antplot of any number depends only on the pattern of digits within it, and is not related to the magnitude of the number.

There is no obvious symmetry in any of the factorial antplots viewed so far for values above 5. Very few factorial antplots resemble any recognisable objects from the real world, probably due to this lack of symmetry. The only examples of pareidolia in antplots of factorials encountered by the author are shown in figures 8 (a dinosaur?) and 9 (a person kicking their leg?).

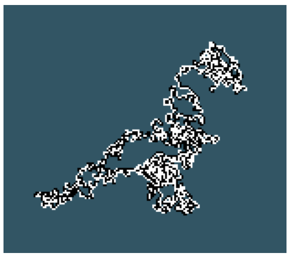
Figure 8: 941!

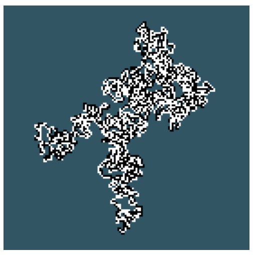
Figure 9: 1316!

It is worth noting that the direction taken by the ant at the start of the plot is arbitrary (up, down, left or right). Changing the initial direction could, for example, cause each of these figures to be rendered upside-down. Looking at the shapes in any other orientation tends to reduce or eliminate the pareidolia, demonstrating how arbitrary and subjective the phenomenon can be.

6. Antplots of e and Pi

So far the results presented have been for finite-length values, except for the recurring fraction calculated to prove the expression for Langton's ant as a ratio of two integers. In this section we turn our attention to two infinitely long irrational numbers.

Other graphical attempts have been made to find recognisable patterns in mathematical constants such as e (Euler's number) and pi. One method uses each digit of the number to control the angle through which a "turtle" turns at each step [2]. The turtle is a plotting device equivalent to our ant. The technique can be applied using any number base from 3 upwards. This is because the angle of turn is always a multiple of 360° divided by the number base. Using binary, the turtle would only turn forwards and backwards along a straight line.

In contrast, the antplot algorithm only works on a binary input string. Values for e and pi have been calculated to a length of over half a million bits, or decimal accuracy of nearly 158000 digits. The corresponding antplots are shown in figures 10 and 11.

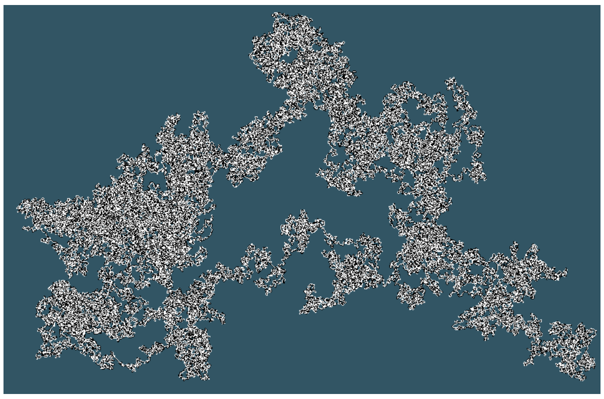
Figure 10: Antplot of Pi

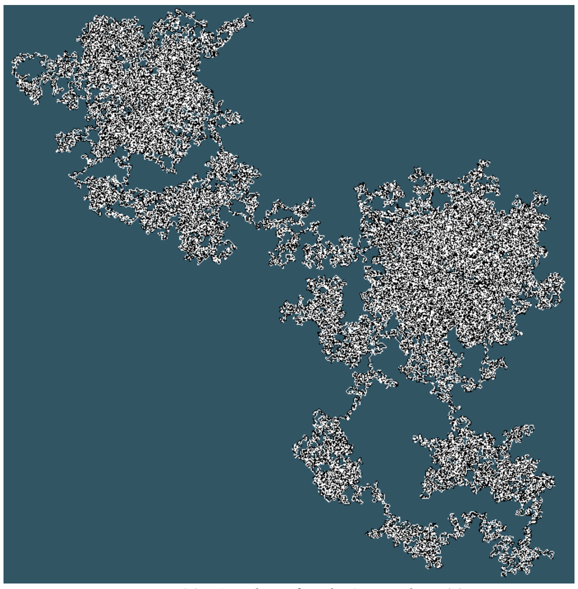
Figure 11: Antplot of Euler's number (e)

Like the antplots for most factorials, there are no instantly recognisable patterns. The antplot trails for e and pi are not easily distinguishable from those for random bit sequences of similar length.

It must be remembered that the antplots in figures 10 and 11 are only for the first half-million or so binary digits of e and pi. Extending the plots using more and more digits of precision could result in the ant trail looping back and overwriting what is already there. In such cases, the antplots obtained might not resemble the ones shown here. Similarly it is possible (though unlikely) that patterns have already arisen early on in the antplots, but these have been overwritten by later movements of the ant, so they are hidden in figures 10 and 11. There is no right answer for how long the plots should be to try to find patterns in irrational numbers.

7. Antplots of Reciprocals

Rational numbers are far more likely than irrationals to yield patterns in their antplots. The software which steps through factorials has been modified to step through reciprocals instead, resulting in antplots with much more structure and symmetry.

Antplots of reciprocals fall broadly into two types:- those which step-and-repeat to form lines which go off to infinity; and those which loop around to cover a finite area. Whether the antplot for the reciprocal of an integer forms a line or a loop depends wholly on the length and bit values of the recurrent pattern in the reciprocal value.

7.1 Line Antplots

A selection of line antplots is shown in figure 12. The first two cases are very simple patterns. The third case looks like a spiral; in fact it is interesting how the antplot algorithm seems to shade the pattern to suggest a 3D shape. Numerous other values also give spiral patterns in their reciprocal antplots. In the fourth case (1/39893), the ant steps and repeats a complex shape due to the relatively long recurring pattern of the fraction. The repeating motif resembles a bird of prey. The final example in figure 12 resembles a line of motorbike riders.

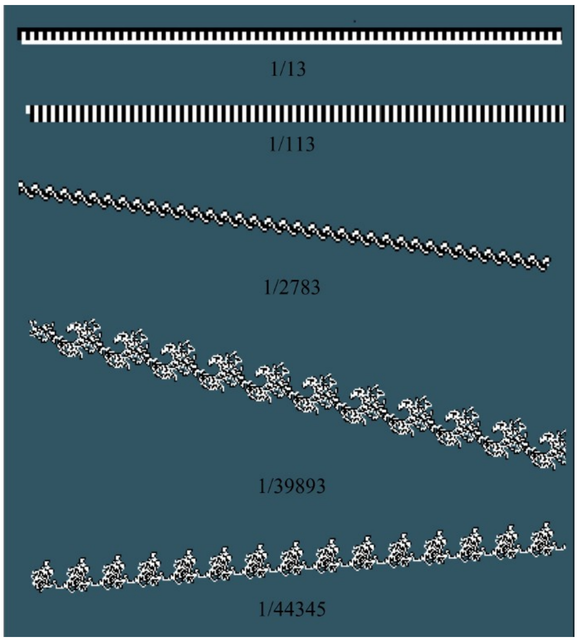

Figure 12: Line antplots

The separators between sections in this document were created from antplots of various reciprocals made against a white background. This leaves only the black squares visible.

In some cases the path of the ant is such that a repeated motif overwrites previous iterations of itself. This gives rise to the kind of effect shown in figure 13. Note the reflective symmetry of the repeated motif, as well as the fact that the black and white pixels are swapped in each half.

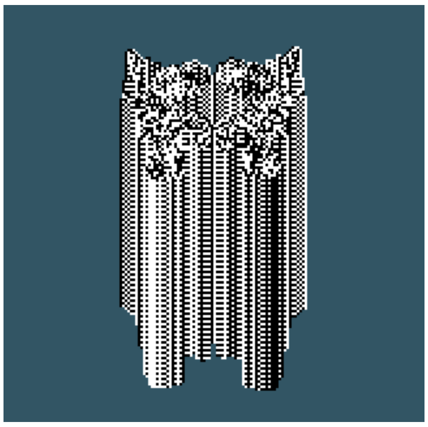
Figure 13: 1/84097 (Angels ascending?)

A similar phenomenon gives rise to the effect seen in figure 14, where each half of the line is entirely black or white, except for the edges. It may seem strange that this is generated by an ant which changes direction at every pixel, and colours it black or white according to the direction of turn. However, examination of just the first few hundred steps of the ant (figure 15) shows that the line is formed by a symmetrical shape which is replicated one pixel left and one pixel up at each step. This leaves the two halves of the line shaded as shown in figure 14.

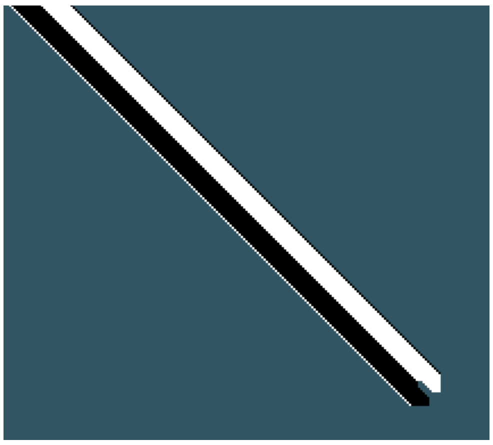
Figure 14: 1/16291

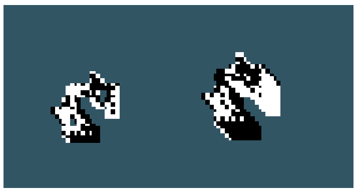
Figure 15: 1/16291 after 627 steps and 1918 steps

7.2 Loop Antplots

Many integers have a pattern in their reciprocal value which causes the ant trail to loop back on itself. Figure 16 shows a selection; these examples and many others show a rotational symmetry of order 4, which we might expect from the rules for movement of the ant on a square grid. There is a surprising degree of intricacy and texture in the figures though.

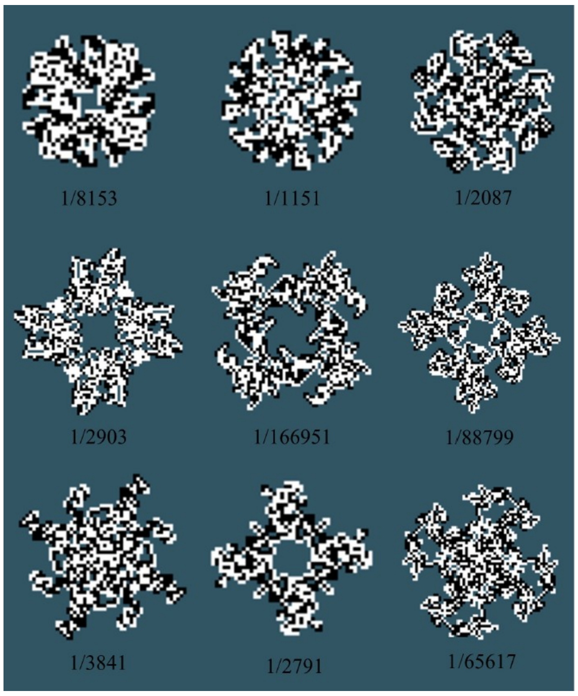

Figure 16: Loop antplots

All the labels shown on antplots in this document are the smallest integer values which will produce each pattern. Any power of two multiplied by the integer value will produce the same bit pattern in the reciprocal, and therefore the identical antplot, as explained in section 5.

In some cases the antplot trail displays symmetry of order 2. A selection of these is shown in figure 17. It is a curious coincidence that the reciprocals of 7135 and 20999 both seem to generate an italic letter ‘f’, or an ‘S’ with an angled ring around its centre.

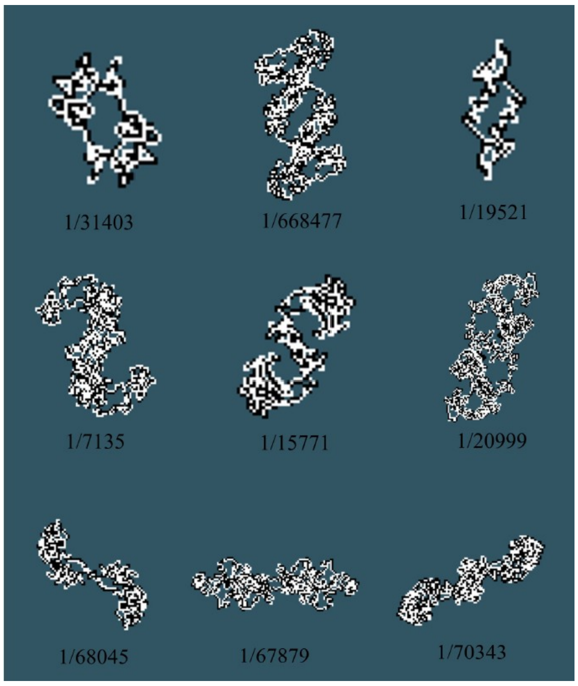

Figure 17: Antplots with symmetry order 2

As yet no obvious relationship has been observed between integer values and whether the antplots of their reciprocals will produce lines or loops. Similarly it is not obvious from integers whose reciprocals produce loop antplots whether the plot will have order 2 or order 4 symmetry. It so happens that all the integer values used in figure 17 (order 2 symmetry) are non-prime, but those in figures 12 and 16 are mixtures of primes and non-primes.

When an antplot forms a loop with symmetry of order 4, it is often the case that the outline is close to being a square. Several such cases are shown in figure 18.

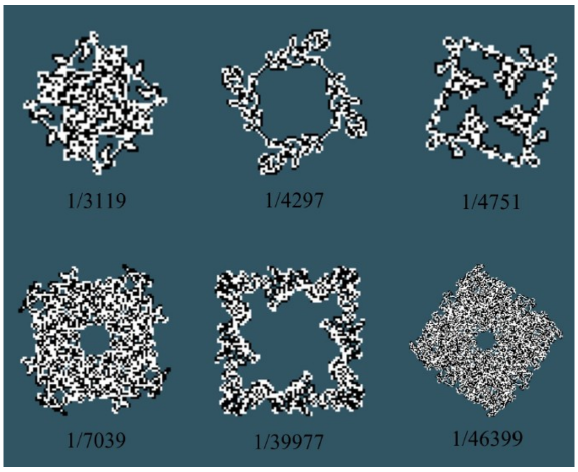

Figure 18: Square antplots

The other shape most commonly formed by loop antplots with symmetry of order 4 is a cross. A number of variations are shown in figures 19 and 20.

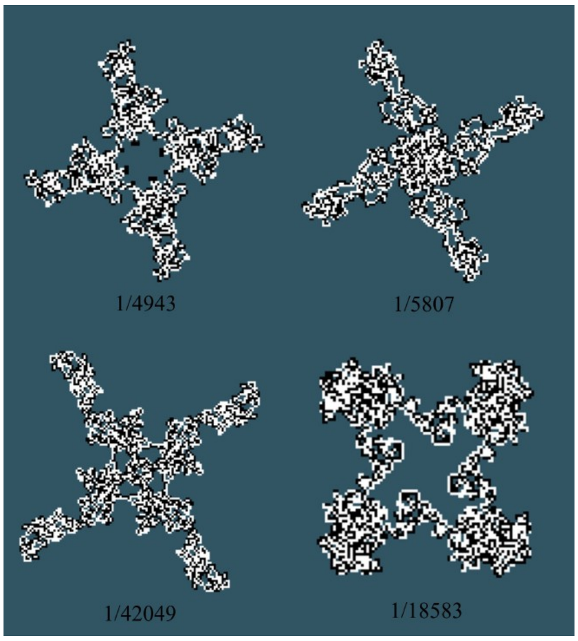

Figure 19: Crosses

Figure 20 shows antplots which have been selected for their resemblance to ancient stonework. The close-packed black and white pixels give an impression of a rough granite-like texture, and an overall grey colour. Several of the shapes echo the form of Celtic or Anglo-Saxon gravestones; in particular the plot of 1/18439 resembles a Celtic cross.

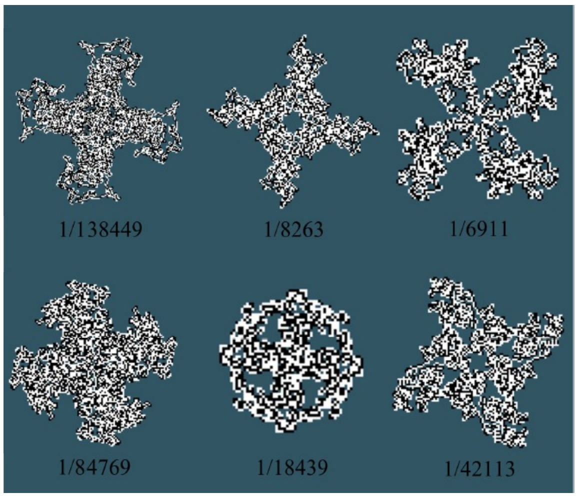

Figure 20: Stone cross antplots

Another symbol which arises in stonework from ancient cultures is the swastika. It was mainly a symbol of good fortune or divinity long before its 20th-century adoption for political extremism. The antplot for the reciprocal of 271 (and power-of-two multiples of it) replicates the swastika very closely, though it is not reproduced here. In other cases (such as 32687, also not shown), the hole left in the middle of the shape is a swastika.

A similar shape is St Brigid's Cross; a cross with 4 off-centre arms, with Celtic or pagan origins. The shapes in figure 21 have been selected for their similarity to St Brigid's Cross. These antplots are also notable for a high degree of intricacy.

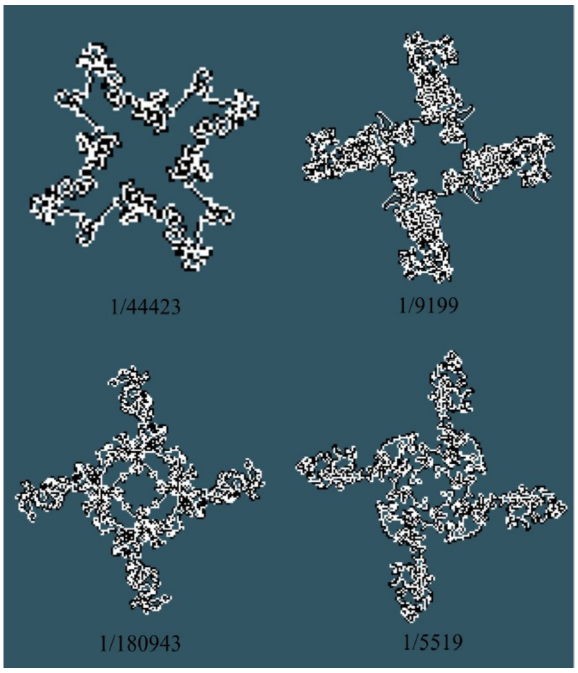

Figure 21: Variations of St Brigid's Cross

As in the case of the swastika, it is sometimes observed that the outline of one reciprocal antplot corresponds closely to the shape of the hole in another. An example of this can be seen where the outline of the antplot for 1/84769 (figure 20) is the same as the hole in the middle of the antplot for 1/5519 (figure 21).

The apparent similarity between antplots of reciprocals of integers and symbols from ancient history seems like a bizarre coincidence. However, there are very many more symbols from ancient cultures based on triangles and other shapes which antplots do not seem to generate.

Cross-shaped antplots probably resemble ancient artefacts due to similarities in the processes by which both are created. It is likely that ancient stonemasons used a template to carve each quadrant of a cross, moving the template round 90° at a time to ensure rotational symmetry. The template is analogous to the recurring pattern of digits in a binary reciprocal. Suppose the recurring bit pattern is such that the ant traces it out and finishes pointing at 90° to its initial direction. The next 3 recurrences will cause the ant to trace the same shape arm in each quadrant, and each further recurrence will then overwrite what is already there with the identical pattern.

8. Software

The antplots presented in this document are a small fraction of those which can be generated. The author has spent many happy hours stepping through antplots in ascending sequences from integers chosen at random, and readers are invited to do the same, but be warned that the process can be addictive.

The source code of the software which generated all the antplots in this document is available to download [1]. It is written in Microsoft Visual C++, and makes use of certain intrinsic 64-bit functions, so porting to platforms other than x64 will likely require some modifications.

9. Conclusion

The antplot algorithm allows us to see patterns, if any exist, within long binary strings. It has been used to verify an integer expression for the trail of Langton's ant, but failed to show any interesting patterns in the values used in the expression.

Antplots of factorial integers, and of the irrational numbers e and pi, also failed to show any meaningful patterns. However, applying the antplot algorithm to rational numbers has provided a surprisingly rich set of graphical outputs.

Simple reciprocals of integers have been found to generate antplots of extraordinary complexity. Further investigation may establish a relationship between integer values and whether the antplots of their reciprocals form lines or loops. It may also be possible to relate values to the degree of symmetry in their reciprocal antplots.

Rational numbers other than reciprocals of integers can be expected to produce an equally rich set of antplots. The only one generated so far has been the trail of Langton's ant from the ratio of two large integers. No single observer will ever be able to see all the reciprocal antplots, let alone those for all rational numbers. A vast space of visual patterns awaits those who wish to explore it.

10. References

[1] M. Agate. Antplot source code software. 2022. https://github.com/DoctorMark/Antplot/

[2] M. Henderson. Plotting Pi and Searching for Mona Lisa. Numberphile, 2 Feb 2022. https://www.youtube.com/watch?v=tkC1HHuuk7c

[3] C. G. Langton. Studying artificial life with cellular automata. Physica D, 22 (1-3): 120–149, 1986

[4] G. Medland. Langtons Ant Wave Equation. SCTPLS Newsletter, 26 (2). Society for Chaos Theory in Psychology & Life Sciences, February 2019

[5] G. Medland. The Langtons ant wave equation. http://www.buzwordsalad.com/

[6] Wikipedia. Langton's ant. https://en.wikipedia.org/wiki/Langton%27s_ant

Biographical Note

Mark Agate has a PhD in 3D computer graphics from the University of Sussex. His interests include DSP and radio control.