The Negative Collatz Sequence

Abstract

The Collatz sequence is not defined for negative numbers, but we show how it operates with negative numbers, and form both a negative Rank table and a negative structure table (albeit with the negative signs suppressed).

The negative Collatz sequence has two known loops, and the correspondence between starting values and one of the three possible outcomes is shown to be a relatively constant fraction of all the possible outcomes up to and beyond 1E300 ($1/3^{rd}$ of outcomes go to each of the three possibilities!)

Using this same idea that loops and infinite-iteration starting points both generate infinite sets of starting values with those outcomes, we probe the positive Collatz sequence up to 1E300 to suggest that loops and infinite-iteration starting points are above that value (if they exist).

We finally probe beyond **millions** of decimal digits without finding any non-terminating values and therefore conclude, on the basis of the analytic evidence, combined with the experimental data, that the positive Collatz sequence always terminates *for all practical applications*.

Other names for the positive problem / sequence include:

Ulam conjecture, Kakutani's problem, Thwaites conjecture, Hasse's algorithm, Syracuse problem, 3*n* + 1 problem, 3× + 1 problem, Hailstone sequence.

Introduction

Much has been written about the Collatz iteration sequence. We assume that readers fall into two broad camps: (1) those who have barely heard of it, and (2) those who know it quite well.

For those who have barely heard of it, we have provided a very readable introduction, written primarily for school children.[1]

For those who know it quite well, a full historical context and prior work is either redundant, or can readily be found online. It is also redundant to quote work which was not consulted in the course of writing this paper.

Since there are subtly different variants of the Collatz sequence in the literature, we define the variant being used here for the sake of clarity.

- **ALG#P:** Start from any positive integer
- **If** the value is even then divide it by two,
  else multiply it by 3 and add 1.
- Repeat the previous step if the resultant value is greater than one.

Notice that the algorithm starts from a positive integer. Starting from zero is quite boring since zero is even, and the next step is zero as 0/2 = 0, and so on. Here we want to consider what happens if we start from strictly negative values.

- **ALG#N:** Start from any negative integer
- **If** the value is even then divide it by two,
  else multiply it by 3 and add 1.
- Repeat the previous step if the resultant value is lesser than minus one.

It is fairly obvious that using **ALG#N** we start from a negative value, iterate with all negative values, and end up with a negative value. This is typographically boring, so it would be convenient to suppress the negative signs on both the input an output values. Adding one to a negative value reduces the magnitude of the result. Dividing by two or multiplying by three have the same effect, regardless of the sign of the operands. It is then clear that **ALG#N** with suppressed negative signs on all inputs and outputs is equivalent to a new algorithm:

- **ALG#NP:** Start from any positive integer
- **If** the value is even then divide it by two,
  else multiply it by 3 and ***subtract*** 1.
- Repeat the previous step if the resultant value is greater than one.

We may implicitly refer to either **ALG#N** or **ALG#NP** as the negative Collatz algorithm (or ⁻Collatz) where context will point to the correct variant.

The reason to study the Collatz sequence of negative numbers is firstly because it exists, but more importantly because it definitely has some useful properties which we have speculated could exist for positive values,[2] but which have never been found in extensive computer searches.

Possibly the negative sequence helps us to understand the apparent non-existence of loops and stray tables in the positive sequence, given that the negative sequence does have these features.

[1] *Introduction to Convergence of the Collatz Sequence*, 2021, Green, L.O.

[2] "*The Inner Structure of the Collatz Iteration Sequence*", 2021, (v1.72, 2022). Green, L.O.

Negative Sequence Rank Table

We consider all odd numbers in the first column, and all $(3\times - 1)$ ⁻Collatz values generated from the odd numbers. We then divide by 2 in successive columns until an odd number is reached. It is the column of this odd number which defines the Rank of the *initial odd number*.

A ***chain*** starts with an odd value, and is guaranteed to ***terminate*** in a *different* odd value, all on the same row of the table. (This use of "terminate" is distinct from its use in the overall Collatz sequence.)

To be clear, starting with a 9 in the left-most column, the 9 is a Rank 1 (R1) value because the chain terminates at 13 in the R1 column.

Starting from 5, every other odd number is an R1 value, { 5, 9, 13, 17, 21, … }.

We will use ↗ (up) to represent the ⁻Collatz $(3\times - 1)$ step, and ↘ (down) to represent a divide-by-two step.

C10: An R1 value is of the form (4k + 1) and terminates in the odd number (6k + 1), where k has the same value throughout.

$(4k + 1)$ ↗ $3(4k + 1) - 1 = 12k + 3 - 1 =$ $(12k + 2)$ ↘ $(6k + 1)$

The value (6k + 1) is clearly odd, so the (4k + 1) chain had only one possible divide-by-two step, and hence was in R1.

An R1 chain ends at a greater value than its start, since $(6k + 1) > (4k + 1)$

An R2 chain also has a systematic repeating pattern. The starting R2 values are { 7, 15, 23, 31, 39, 47, 55, … }, being of the form (8k – 1).

C11: An R2 value is of the form (8k – 1) and terminates in the odd number (6k – 1), where k has the same value throughout.

$(8k - 1)$ ↗ $3(8k - 1) - 1 =$ $(24k - 4)$ ↘ $(12k - 2)$ ↘ $(6k - 1)$

odd	3x-1	R1	R2	R3	R4	R5	R6	R7
1								
3	8	4	2	1				
5	14	7						
7	20	10	5					
9	26	13						
11	32	16	8	4	2	1		
13	38	19						
15	44	22	11					
17	50	25						
19	56	28	14	7				
21	62	31						
23	68	34	17					
25	74	37						
27	80	40	20	10	5			
29	86	43						
31	92	46	23					
33	98	49						
35	104	52	26	13				
37	110	55						
39	116	58	29					
41	122	61						
43	128	64	32	16	8	4	2	1
45	134	67						
47	140	70	35					
49	146	73						
51	152	76	38	19				
53	158	79						
55	164	82	41					
57	170	85						
59	176	88	44	22	11			
61	182	91						
63	188	94	47					
65	194	97						
67	200	100	50	25				
69	206	103						
71	212	106	53					
73	218	109						
75	224	112	56	28	14	7		
77	230	115						
79	236	118	59					
81	242	121						
83	248	124	62	31				
85	254	127						
87	260	130	65					
89	266	133						
91	272	136	68	34	17			
93	278	139						
95	284	142	71					
97	290	145						
99	296	148	74	37				
101	302	151						
103	308	154	77					
105	314	157						
107	320	160	80	40	20	10	5	
109	326	163						
111	332	166	83					
113	338	169						
115	344	172	86	43				
117	350	175						
119	356	178	89					
121	362	181						
123	368	184	92	46	23			
125	374	187						
127	380	190	95					
129	386	193						
131	392	196	98	49				
133	398	199						
135	404	202	101					

C12: An R3 value is of the form (16k + 3) and terminates in the odd number (6k + 1), where k has the same value throughout.

$(16k + 3) \nearrow 3(16k + 3) - 1 =$

$(48k + 8) \searrow (24k + 4) \searrow (12k + 2) \searrow (6k + 1)$

C13: An R4 value is of the form (32k – 5) and terminates in the odd number (6k – 1), where k has the same value throughout.

$(32k - 5) \nearrow 3(32k - 5) - 1 =$

$(96k - 16) \searrow (48k - 8) \searrow (24k - 4) \searrow (12k - 2) \searrow (6k - 1)$

C14: An even-Ranked number, RE, is of the form $(2^{E+1}k - (2^E - 1)/3)$ and terminates in the value (6k – 1).

$(2^{E+1} \times k - (2^E - 1)/3) \nearrow 3(2^{E+1} \times k - (2^E - 1)/3) - 1 = 3 \times 2^{E+1} \times k - 2^E$

$= 2^E (6k - 1) \searrow \ldots \searrow 2^2 (6k - 1) \searrow 2^1 (6k - 1) \searrow (6k - 1)$

The E-index denotes the number of divide-by-two operations necessary before the odd value (6k – 1) is reached.

Comparing the Rank tables for positive and negative Collatz sequences:

	$^{+}$Collatz		$^{-}$Collatz	
	start	**end**	**start**	**end**
R1	4k – 1	6k – 1	4k + 1	6k + 1
R2	8k + 1	6k + 1	8k – 1	6k – 1
R3	16k – 3	6k – 1	16k + 3	6k + 1
R4	32k + 5	6k + 1	32k – 5	6k – 1
R5	64k – 11	6k – 1	64k + 11	6k + 1
R6	128k + 21	6k + 1	128k – 21	6k – 1

The $^{-}$Collatz Rank table is very similar to the $^{+}$Collatz Rank table. Both have 100% coverage over the odd numbers, and Ranks have regular positions within the tables. The Rank table on its own apparently gives us no clues about loops (since the $^{-}$Collatz sequence is known to have loops).

The $^{-}$Collatz Structure table is shown overleaf. It looks very similar, in an overview sense, to the normal (positive) Structure table. Again there is absolutely no indication of loops.

As with the $^{+}$Collatz structure table, there are dark blue shaded rows and columns. Observe that all these values are multiples of 3. Clearly an integer cannot simultaneously be of the form (3k) and (3k ± 1). From an odd Level (shown in green), an odd integer *n* is transformed to the next lower even Level by use of the form (3*n* ± 1).

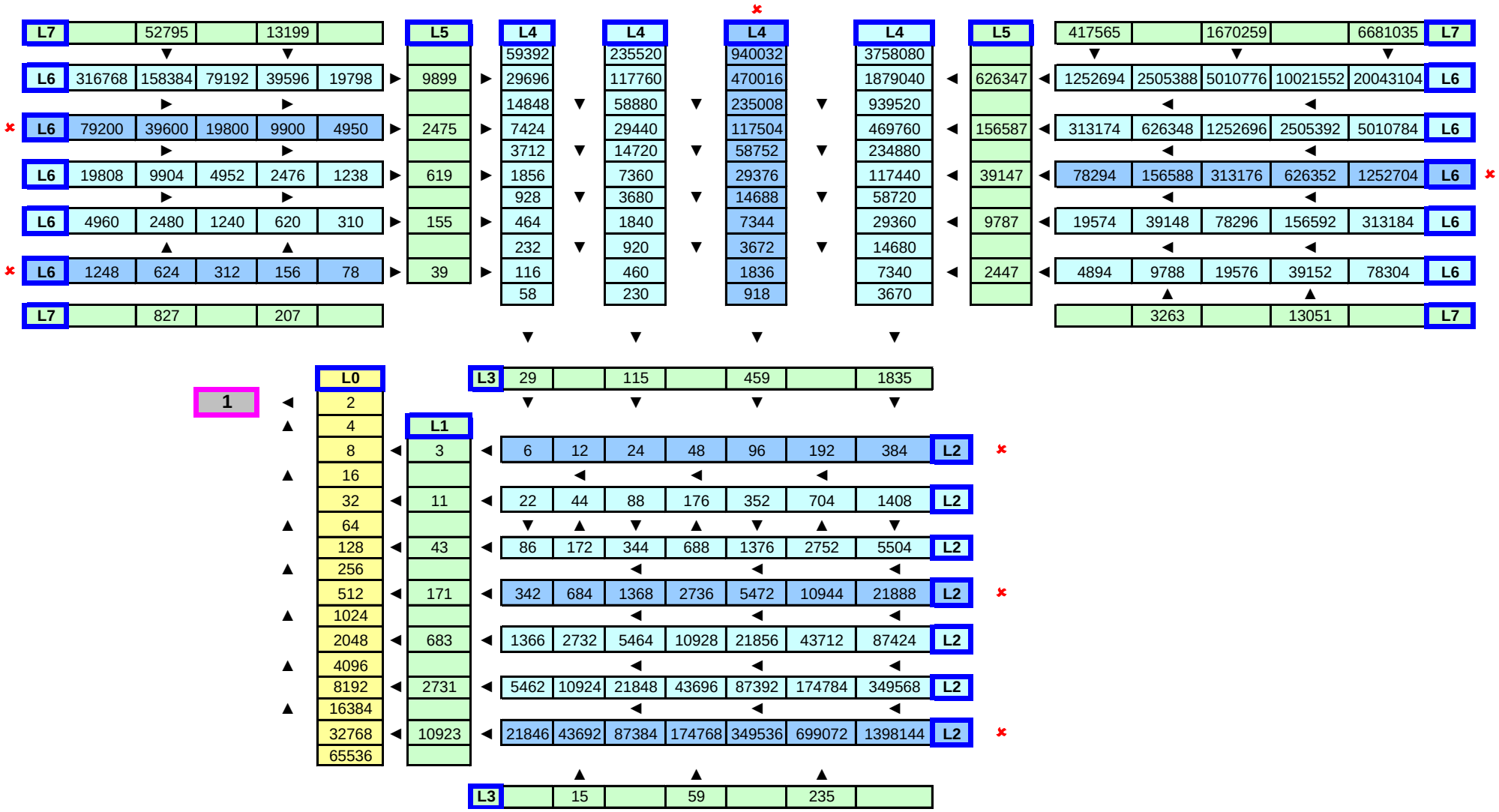
It means the dark blue rows/columns can never be reached from an odd Level.

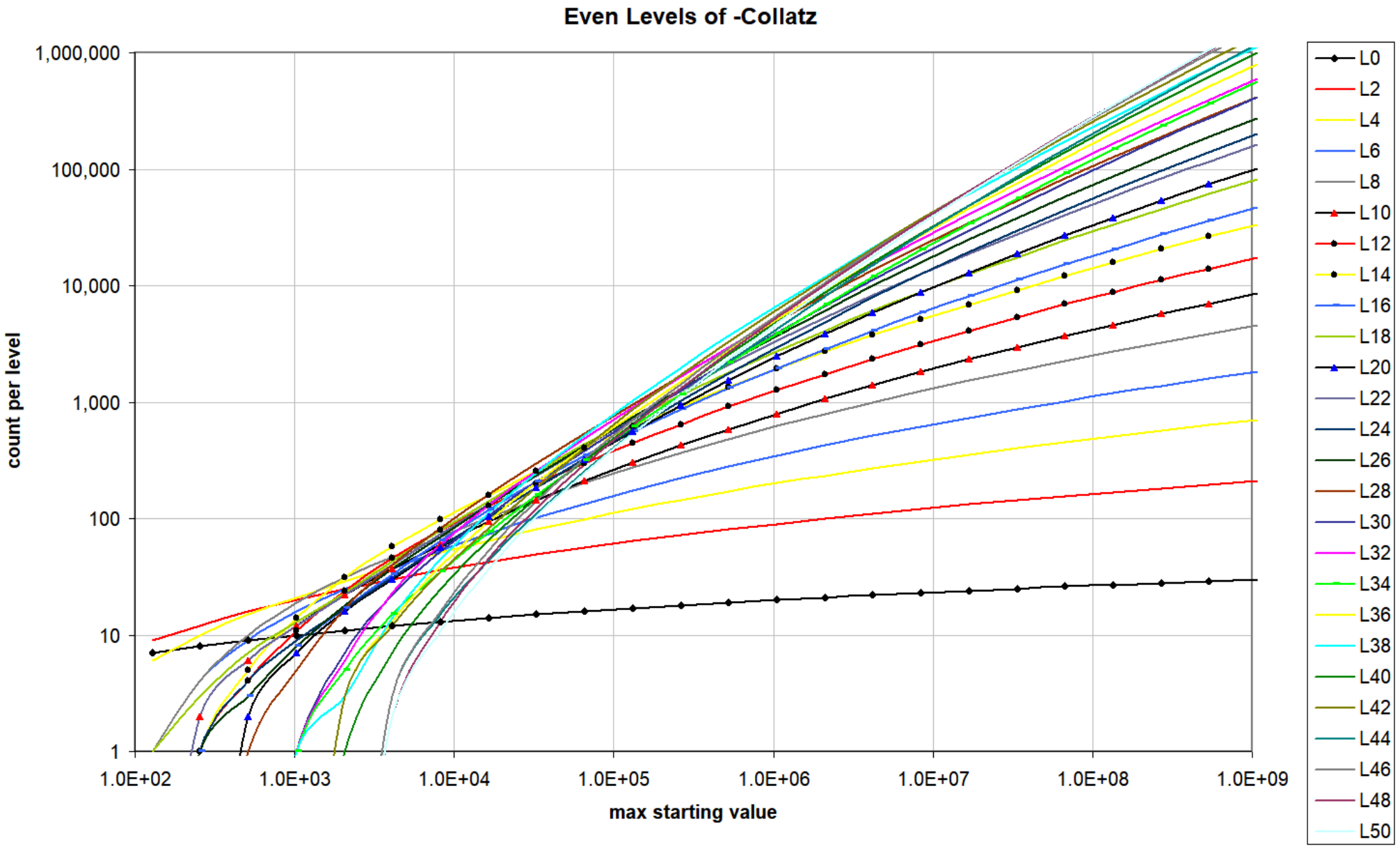
In the earlier paper, the $^{+}$Structure table values in the odd Levels were related by a (4× + 1) relation.[3] In the $^{-}$Structure table the relation is changed to (4× – 1).

The interested reader is invited to apply the method from the earlier paper to confirm these relationships.

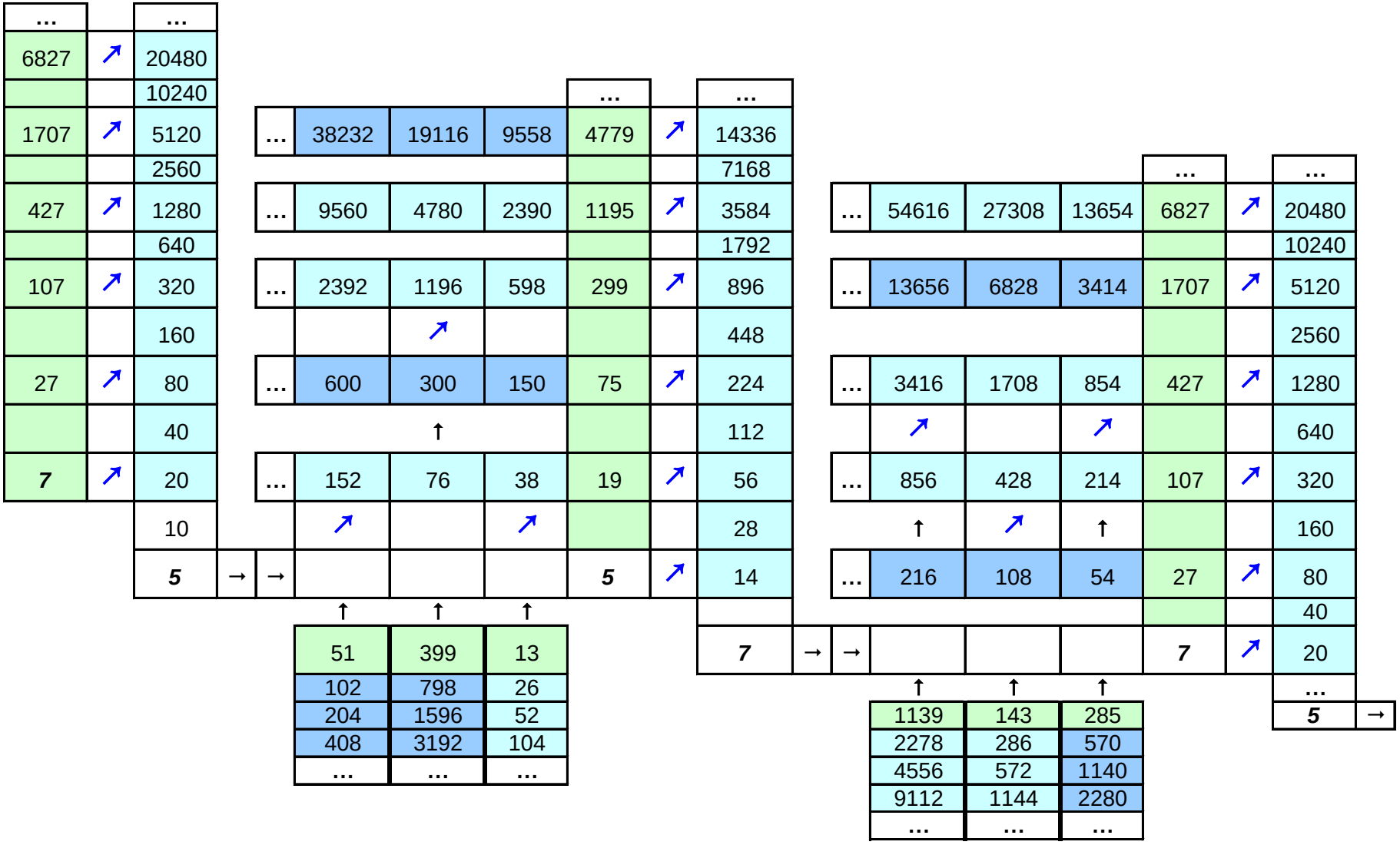
[3] See C65, C66, C67, C68, C69 in the earlier paper.

Collatz Structure Table




Even Levels of -Collatz
count per level
1,000,000
100,000
10,000
1,000
100
10
1
1.0E+02
1.0E+03
1.0E+04
1.0E+05
1.0E+06
1.0E+07
1.0E+08
1.0E+09
max starting value
L0
L2
L4
L6
L8
L10
L12
L14
L16
L18
L20
L22
L24
L26
L28
L30
L32
L34
L36
L38
L40
L42
L44
L46
L48
L50

⁻Collatz Stray Table (small loop)



In the previous paper,[4] (*to which we will frequently refer*), the stray table was an abstract concept, since we could never find a suitable starting value. Here we have a simple genuine example that we can study. Even though this table only cycles through the (odd numbered) loop 5 → 7 → **5**, starting values which follow this loop are nevertheless infinite in extent. The question we would like to answer is: "How does this table grow as we move to larger and larger numbers?"

To start to answer this question we consider perfect squares. As we head up the natural number line we find that perfect squares become increasingly less common. Number theorists fudge the question by saying that *almost no natural numbers are perfect squares.* Recognise that this is a *qualitative* answer to a quantitative requirement.

For the Collatz sequence in general we have exactly 3 possible **outcomes** for a given starting value:

1) normal termination (at 1).
2) gets stuck in a loop and never terminates.
3) heads off to infinity and never terminates.

For the standard Collatz sequence, no numerical result has ever been obtained where a value does anything other than terminate at 1. However, the ⁻Collatz sequence gives us a rare opportunity to look at other outcomes. Whilst outcome (3) has not been observed, it is trivial to observe loops in the ⁻Collatz sequence.

Specifically, listing only the odd values, we have:

(A) 5 → 7 → **5**

(B) 17 → 25 → 37 → 55 → 41 → 61 → 91 → **17**

[4] "*The Inner Structure of the Collatz Iteration Sequence"*, 2021. (v1.72, 2022) Green, L.O

Needless to say, we actually have two disjoint stray tables here!

We therefore have three possible (or at least readily findable) outcomes. We can then use a computer to answer the question of how frequently these outcomes occur.

We would like to consider the size of the stray table as a function of the maximum listed number. It would appear that the size of the cycle (A) table would be smaller than the cycle (B) table because cycle (A) starts from two odd values whereas cycle (B) starts from seven odd values.

If the structure table, stray table (A), and stray table (B) all had the same starting number count (catchment area), if we started from all integer values up to N, we should expect that one third of them would terminate normally, one third would hit cycle (A), and one third would hit cycle (B). *This is not a probabilistic argument*. There is a set of numbers which form the Structure table. There is another set for stray table (A). There is another set for stray table (B). All three sets are necessarily disjoint, and the sum of elements in all of them is equal to N. Testing all the numbers in the given range ***measures*** the size of the sets for that given value of N.

On direct test, with N increasing in decade steps, the fractional number in each set is substantially the same for each decade increment of N up to 1E9. We therefore state (*due to symmetry*) that this constancy of fractional size is a feature of any stray stable. Stray table (B) is no larger than stray table (A), despite having a large cycle length.

It should be noted that starting at 2672464025 (2.6E9) the intermediate value iterates up to the point where it exceeds INT64_MAX/3, so the next UP step cannot be allowed.

Although we have previously mentioned that a stray table is of infinite extent, since all iteration branches can descend from an indefinite value above odd elements, we have not applied this same idea to a growing non-terminating sequence. For example if S is some odd natural number,

which is the smallest available value from which the iteration heads off to infinity, there is a whole (infinite) upwards chain of $S \times 2^k$ which would iterate down to S. Likewise, having iterated up to $(3S - 1)$ there is another infinite chain of $(3S - 1) \times 2^k$ and so on. This is no different to the spread of a stray table, so we should deduce that if we get well above S (numerically), we will have ample opportunity to be 'captured' by (or fall into) the infinite-iteration path.

Randomness in the ⁻Collatz Sequence Outcomes

We know that the ±Collatz sequences are entirely deterministic, following the ±Rank tables, and ±Structure tables. But there is a probabilistic element in the sense that if we pick an arbitrary starting value at random we could assign a probability to the Rank of the number found (1/2 will be in R1, 1/4 in R2, 1/8 in R3, 1/16 in R4 etc.)

For the ⁻Collatz sequence in particular we know (by computational evidence) that we have approximately equal chances of randomly picking one of the three iteration outcomes. This sounds as though the outcomes might be randomly distributed throughout the tested range, but it could easily be the case that there is bunching of outcomes.

We can examine this idea with a coloured map, where each starting value is coloured according to its iteration outcome. We have made this map 60 squares down and 130 squares across, landscape format being useful for viewing on a computer screen.

We start from 1 in the top left corner, with 60 in the bottom left corner, and 61 again being at the top but one column in. The total is 7800 squares. The top left is always the starting value, even if we start from 1E9.

The colour maps are on successive pages so a bit of scrolling is called for. The first map is just using three colours picked randomly (rand() % 3). The result looks suitably random.

The second map is coloured according to the iteration outcome, and again has one of three colours. By eye they seem a bit more clumped together than the pseudo-random number generator version.

The third map starts from 1E9 and shows considerable green dominance. The fourth map, starting from 1.1E9, shows considerable red dominance.

We are now in a position to outline our plan of attack, and of course the goal of this attack.

GOAL: Answer the question: Does the ⁺Collatz sequence always terminate?

Plan of Attack:

Using the ⁻Collatz sequence, find a test method which works accurately to find out if the ⁻Collatz sequence has loops or indefinitely increasing responses. This seems like a redundant step in the sense that we already know it has at least two loops! Indeed we test for a loop by seeing if the iterated value ever hits 5 or 17. However, for an unknown loop starting point, we obviously cannot hard-code fixed values. We therefore need to create a test which is independent of the actual loop limits. Storing all previous values up to some ill defined limit also seems computationally infeasible. We therefore only look for points which take an 'unreasonably long time' to iterate. More on this later.

If we 'sample' starting numbers for the ⁻Collatz sequence, we should find that around 67% of the time we find values which are impossible to iterate to termination. For a test, we could assume that only 1% of starting numbers were 'resistant' to termination. If we randomly tested 100 starting numbers, and not one failed to terminate, we would say that the probability of this happening was $(1 - 0.01)^{100} = 0.366$. Too big.

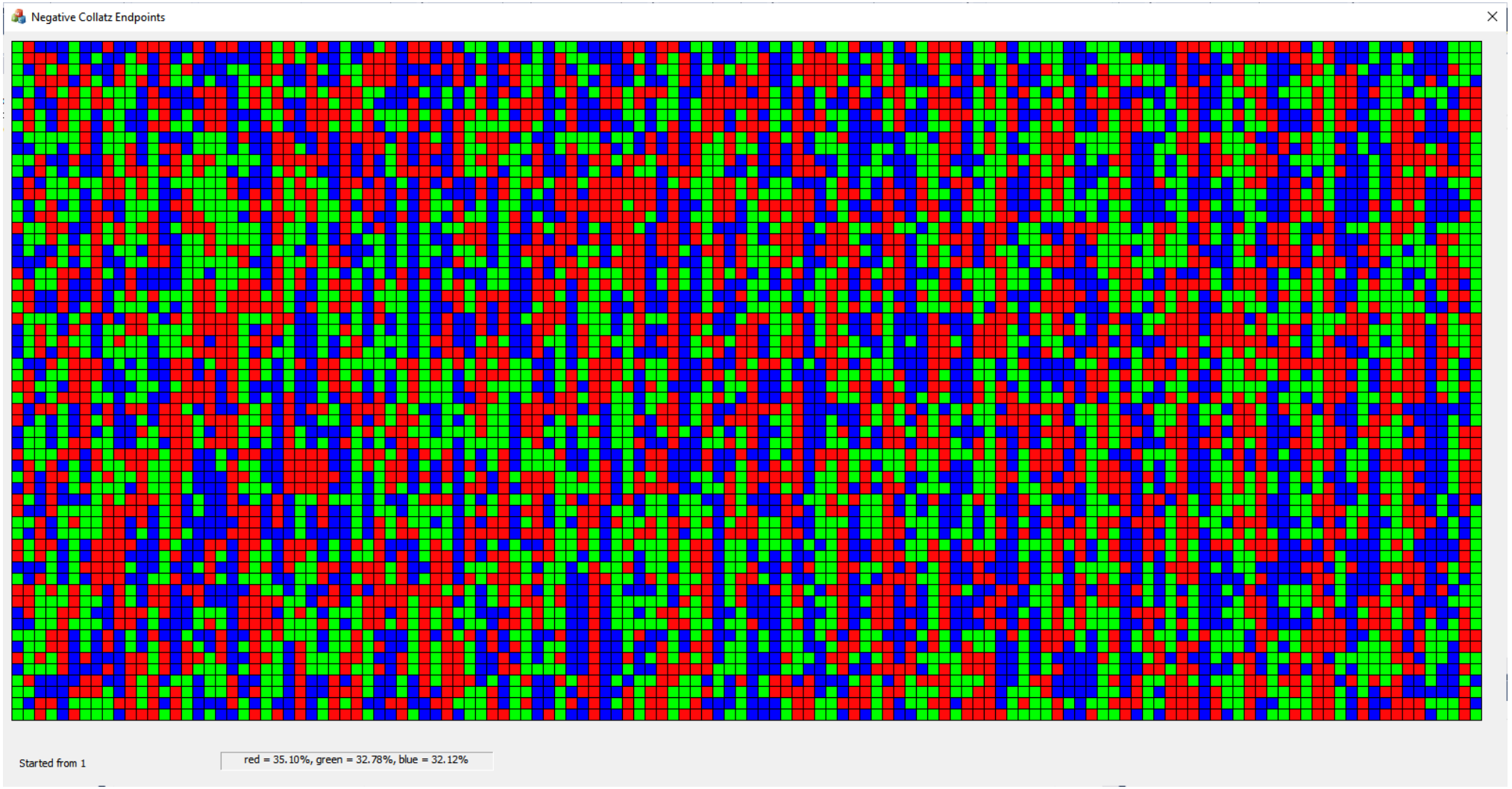
$$(1 - 0.01)^{1000} = 4E\text{-}5 \quad (0.004\%)$$

Not finding a resistant point is potentially strong evidence.

Random 3-Colour Map

Collatz Outcome Map: Starting from 1

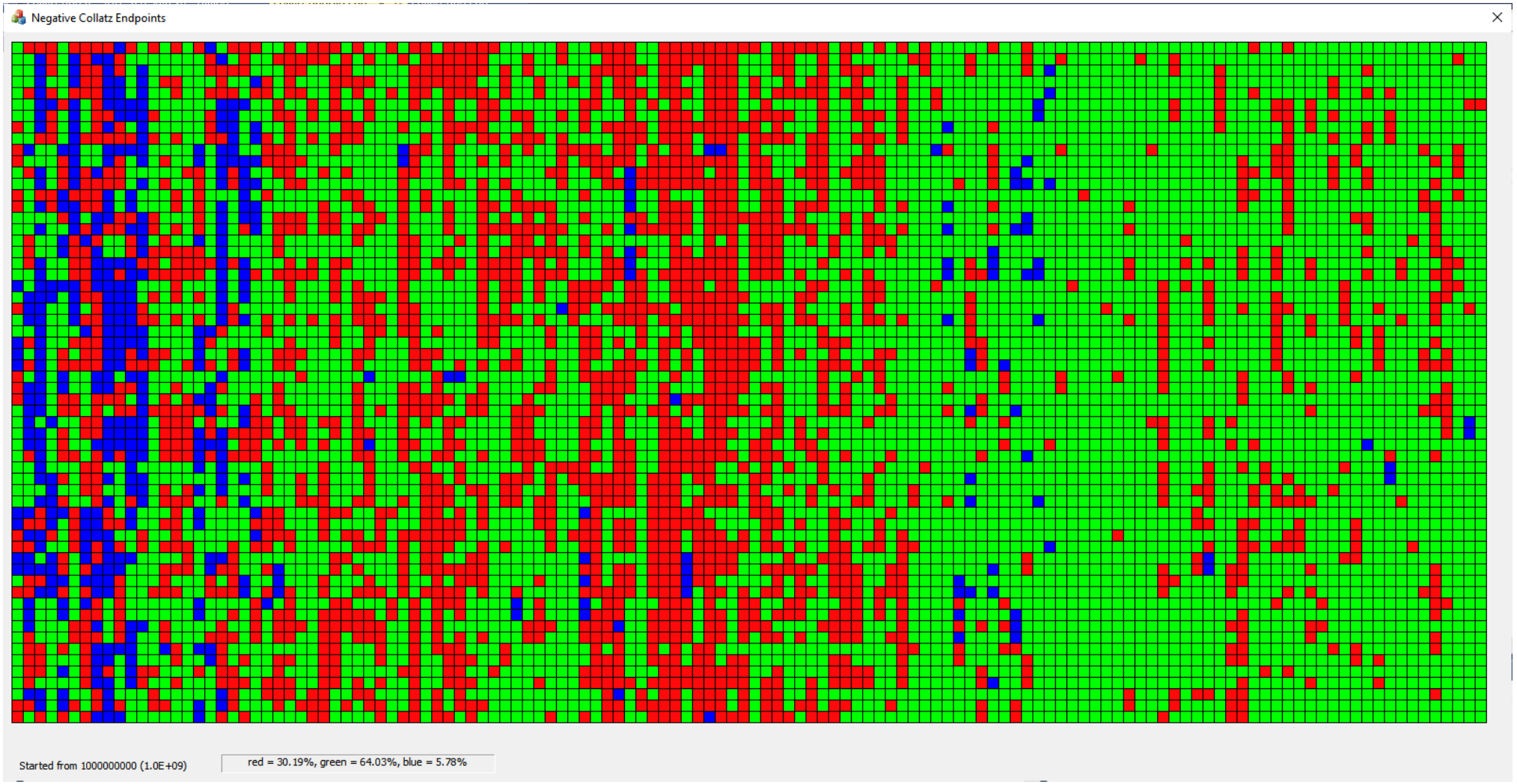
Green = terminates at 1; Blue = loops to 5; Red = loops to 17



This pattern does not look ‘as random’ as the 16-bit rand() value used for the previous plot.

Collatz Outcome Map: Starting from 1.0E9

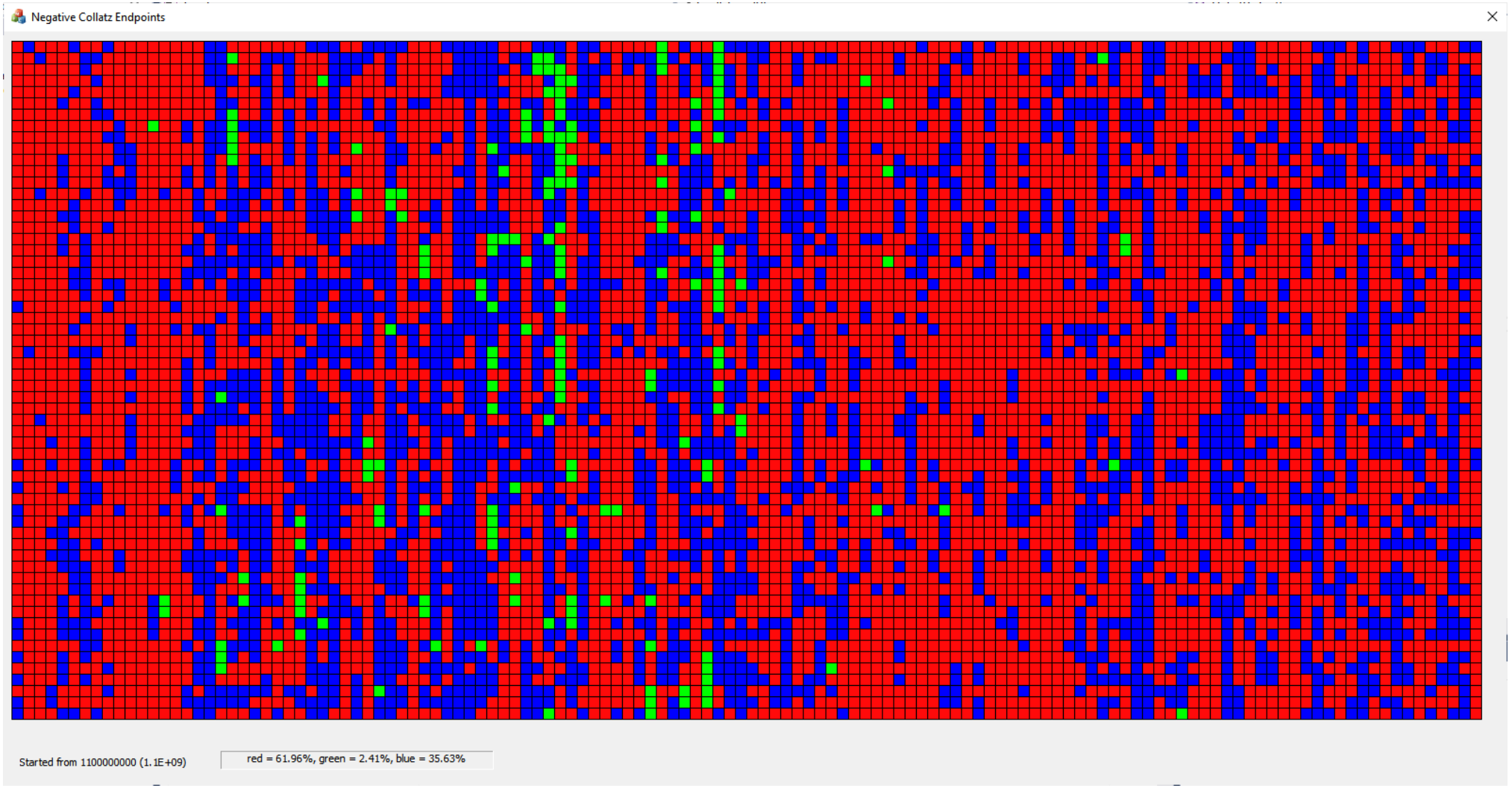
Green = terminates at 1; Blue = loops to 5; Red = loops to 17



Clearly green (normal termination) dominates in this region.

Collatz Outcome Map: Starting from 1.1E9

Green = terminates at 1; Blue = loops to 5; Red = loops to 17



Red (loops at 17) dominates in this region.

Big Starting Points

The statement "Not finding a resistant point is potentially strong evidence." needs immediate clarification. The fact that all starting values up to 10^{20} have been tested and found to converge for the $^{+}$Collatz sequence is no evidence at all for large numbers.

Suppose however we test 1000 or more starting values above say 10^{22}. If we find no unreasonably difficult numbers, we could conclude that any loops or starting point for heading off to infinity was either non-existent or *above* the point we tested. This allows the possibility of leap-frogging ahead of the distributed computing project which is testing all possible values.

Such a test is totally successful if it finds a counter-example. It would be a Collatz killer result! Any single numerical counter-example kills the possibility that the conjecture is correct. The new test gives a higher probability of finding such a counter-example in a given amount of time since we can search up to ridiculously large values relatively quickly.

It is important to state the analytic basis for such testing:

(1) A loop (or iterate-to-infinity starting point) always has an infinite (fractional) 'capture area' when sampled above that value.

We have numerically tested this idea over seven decades of the $^{-}$Collatz sequence, and we get around 33% of values looping at 5, and 33% of values looping at 17. We use the symmetry between the stray table and the structure table for the larger claim that this is the *characteristic behaviour* of loops.

(2) Our sampling is adequate to overcome any local clumping of iteration outcomes.

We have seen such clumping on a small scale (up to 1E9), but numbers above 1E20 may suffer from increased clumping.

(3) $^{+}$Collatz behaves similarly to $^{-}$Collatz in broad terms.

We are actually dealing with the *same* iteration sequence, and just a different input set, so some similarity in the responses should reasonably be expected. We have also constructed a $^{-}$Rank table and a $^{-}$Structure table, which are similar to their positive counterparts.

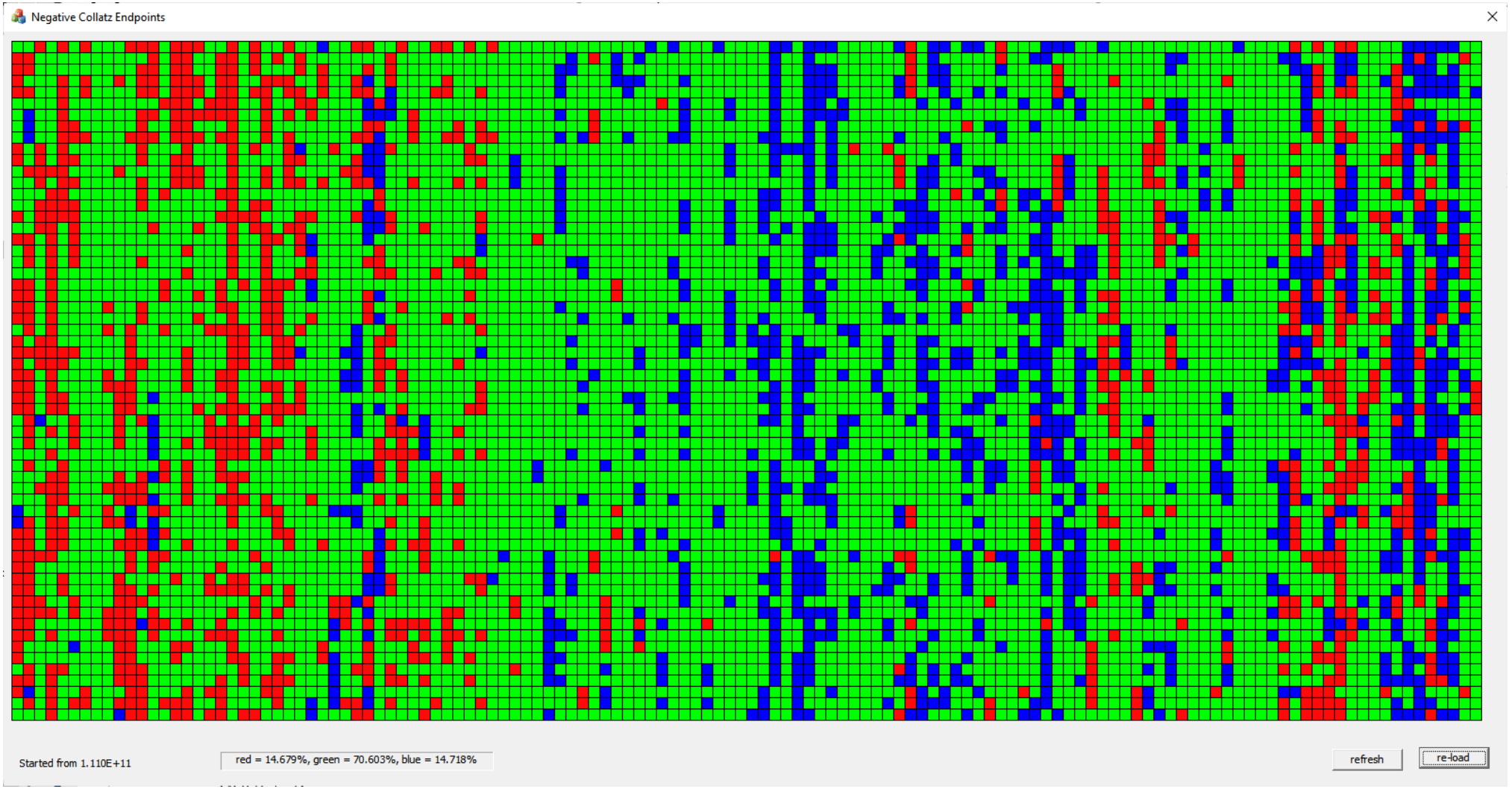
It is these statements which are the limiting conditions for the test to work. The confidence intervals required for sampling are relatively unimportant compared to these larger considerations.

Sampling adequacy is something we can investigate numerically. We can change our iteration loop to use MPIR [5] integers, and then push the loop starting point up to ridiculous values without much (computational) effort.

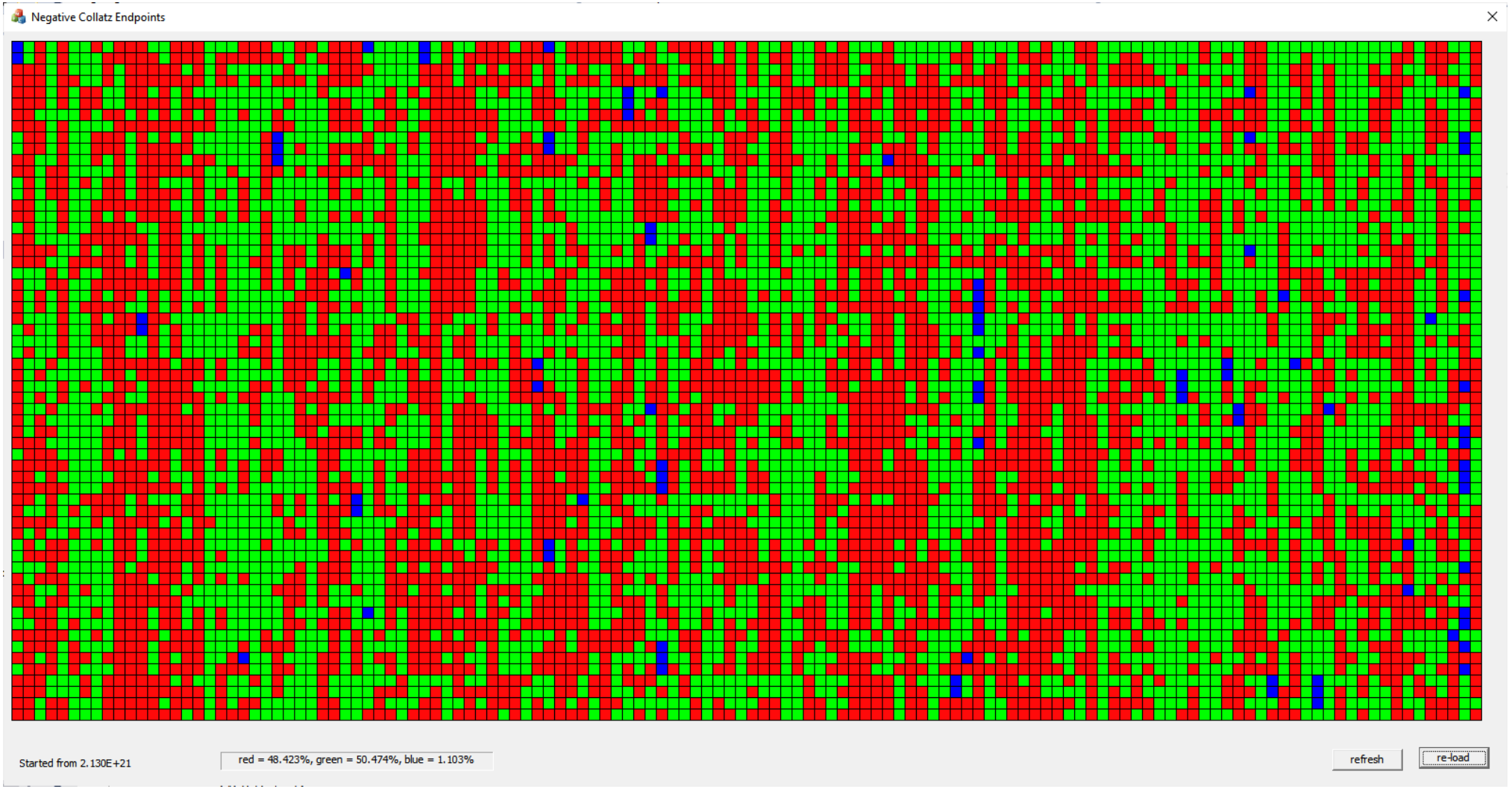
It is convenient to separate the computing into two parts. The display part is a 32-bit windows application. The generation part is a 64-bit windows console application, which generates a small text file of the termination outcomes.

[5] https://mpir.org/ Whilst ordinary integers are declared as int or _int64, in MPIR the large integers are of type mpz_t and have essentially unlimited size.

Collatz Outcome Map: Starting from 1.11E11

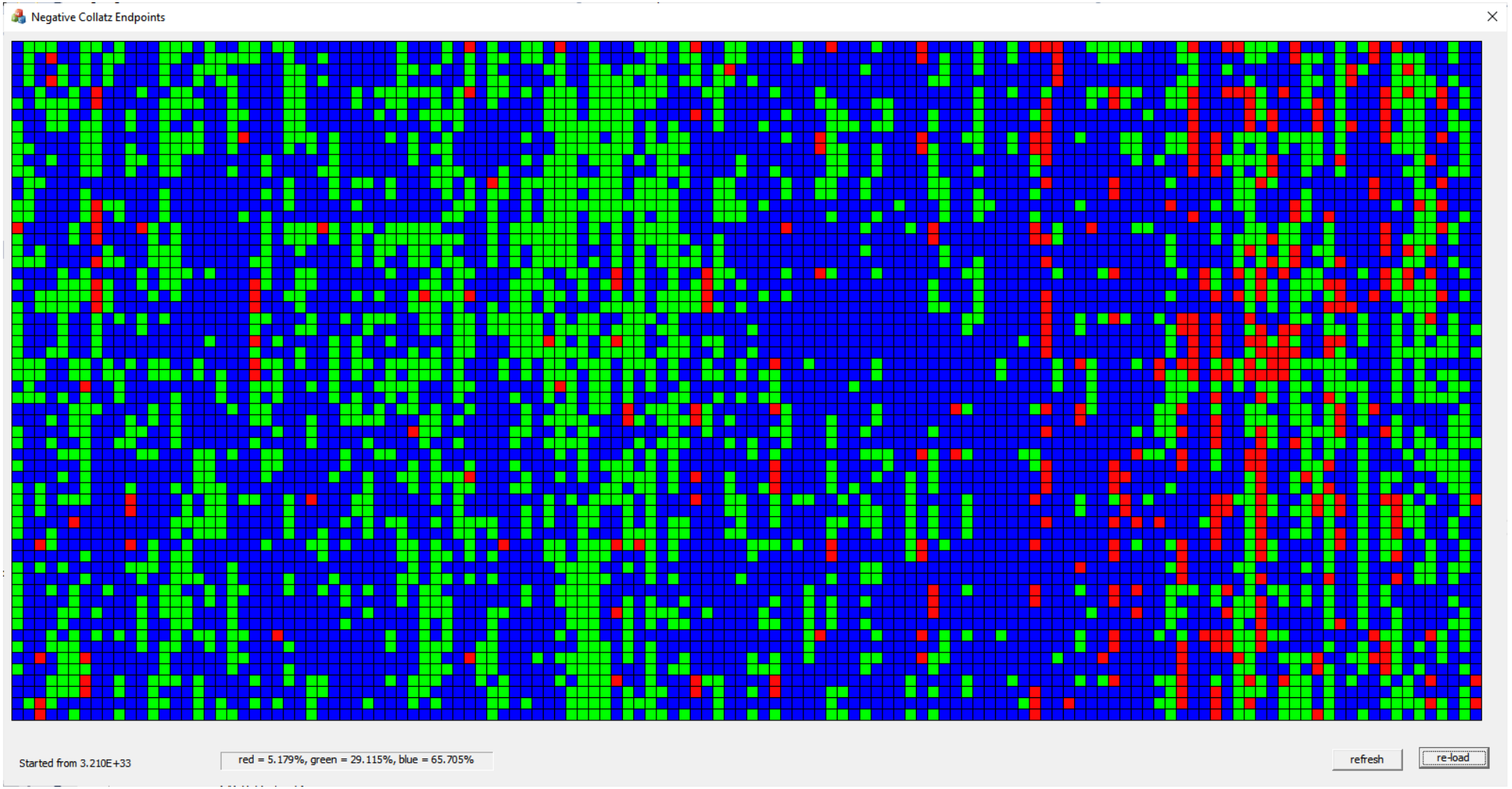


Collatz Outcome Map: Starting from 2.13E21



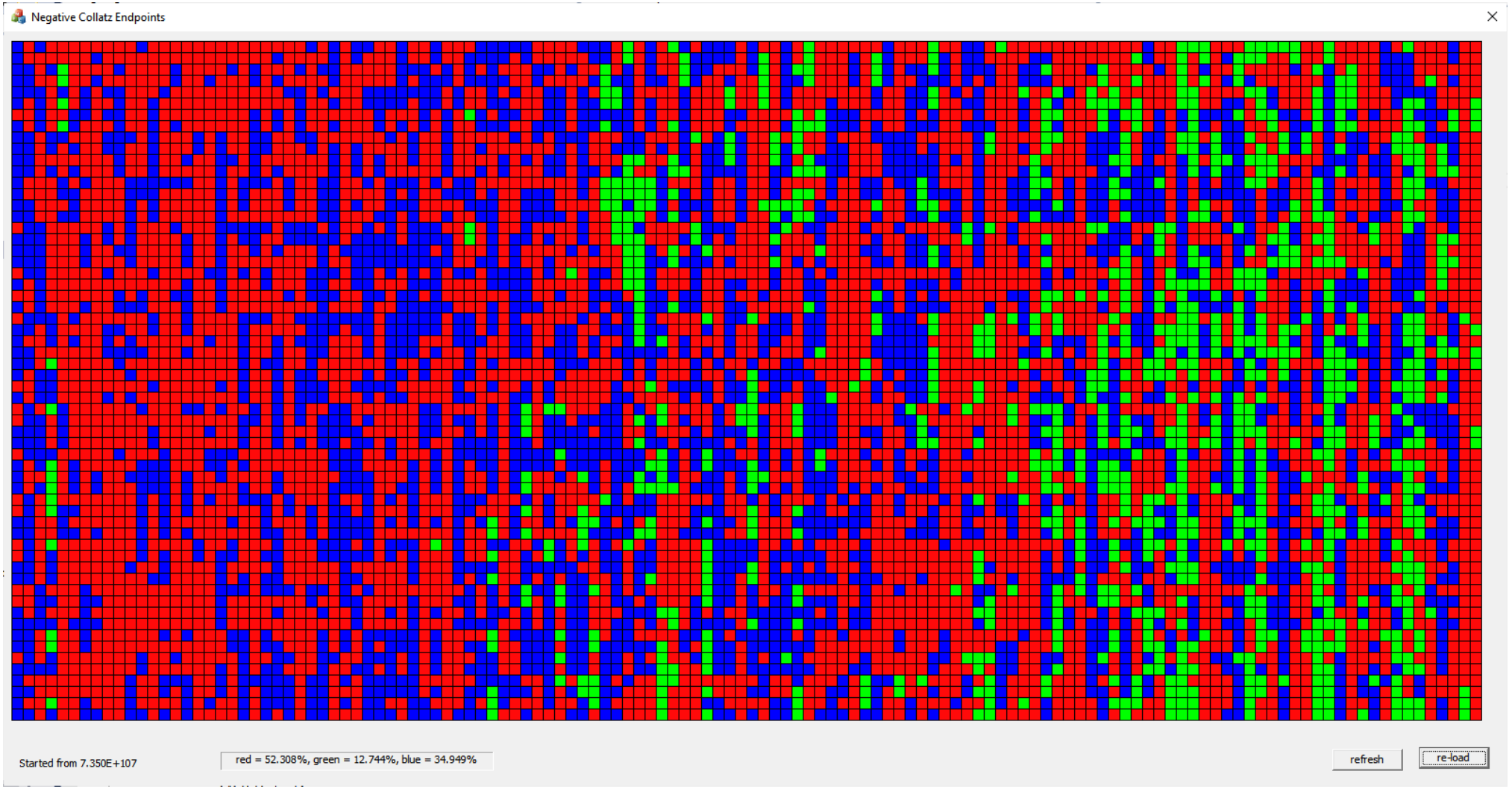
Collatz Outcome Map: Starting from 3.21E33

(run time 2 seconds)



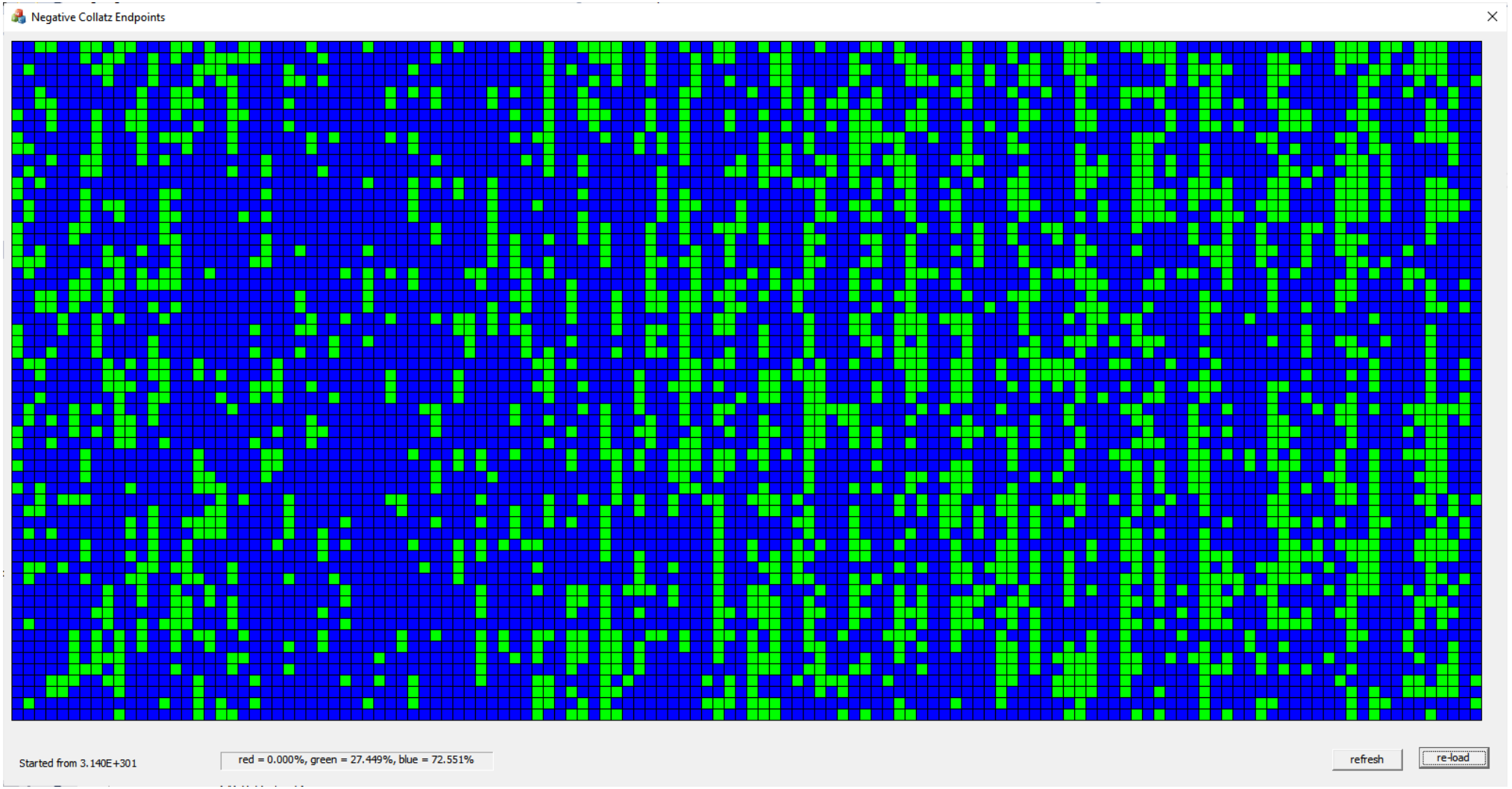
Collatz Outcome Map: Starting from 7.35E107

(run time 6 seconds)

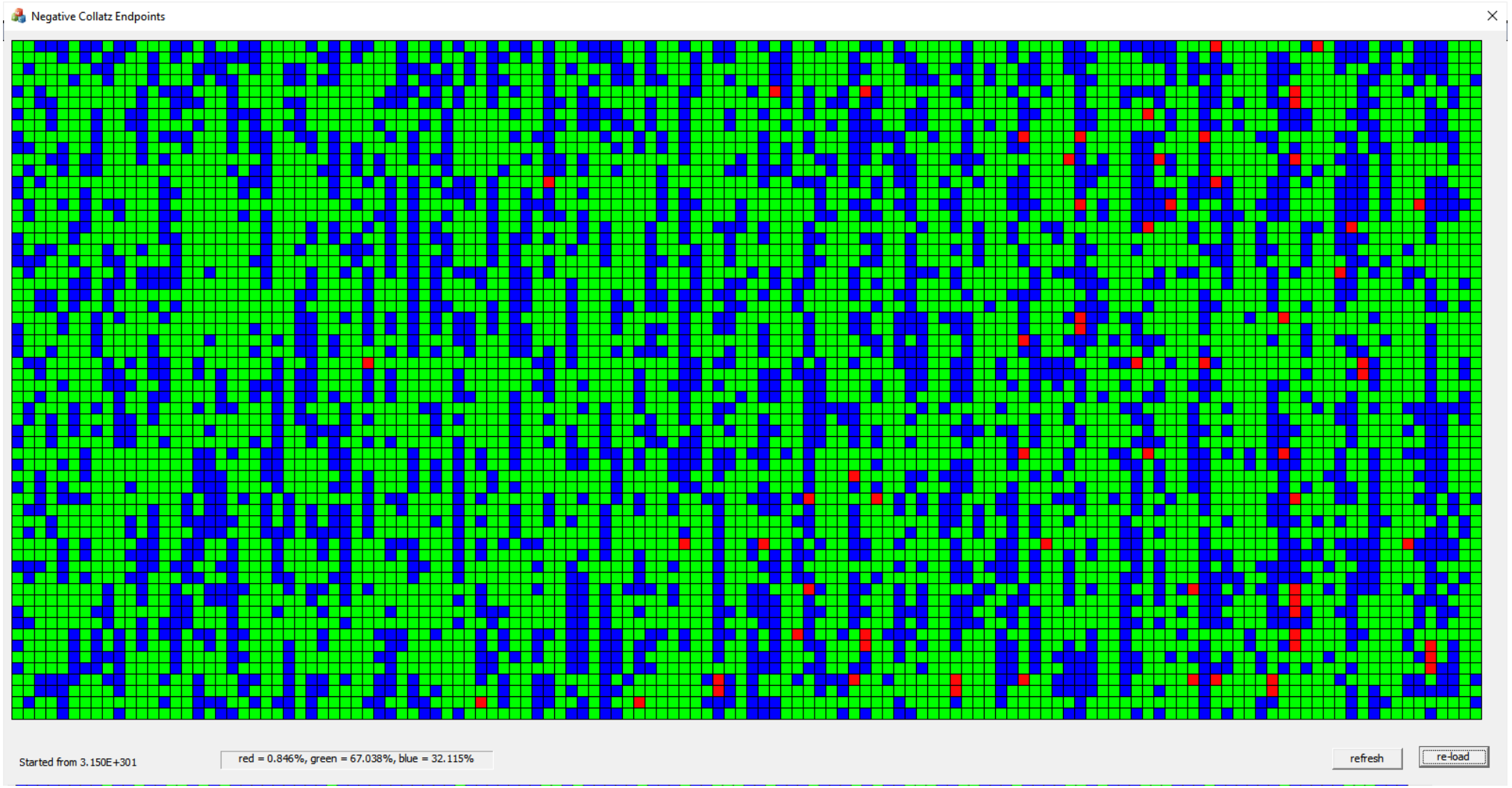


Collatz Outcome Map: Starting from 3.14E301

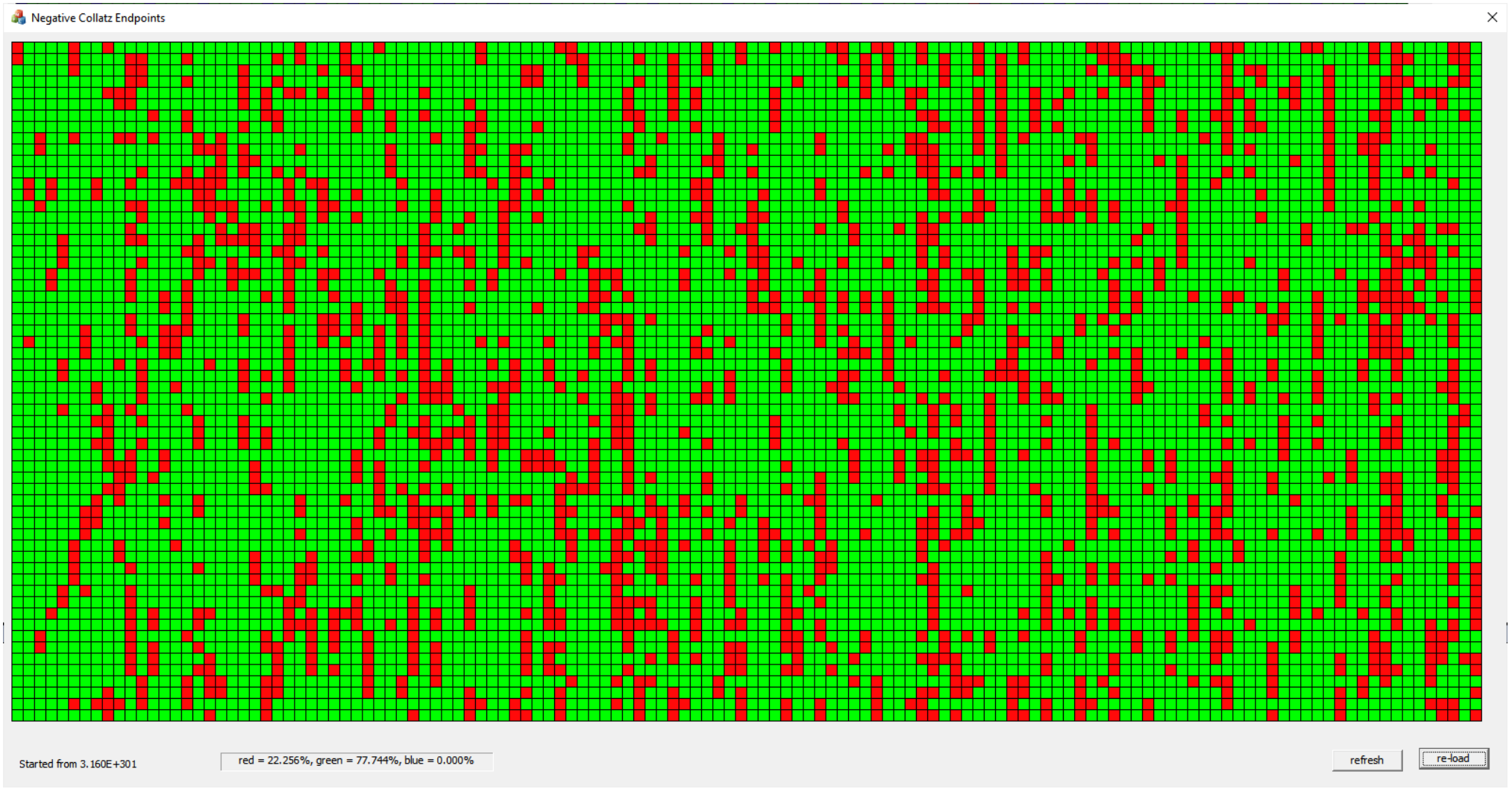
(run time 19 seconds)



Collatz Outcome Map: Starting from 3.15E301



Collatz Outcome Map: Starting from 3.16E301



We could keep going, but the type double runs out of range much above 1E308. The point is we can see definite evidence of clumping as we go up the number line.

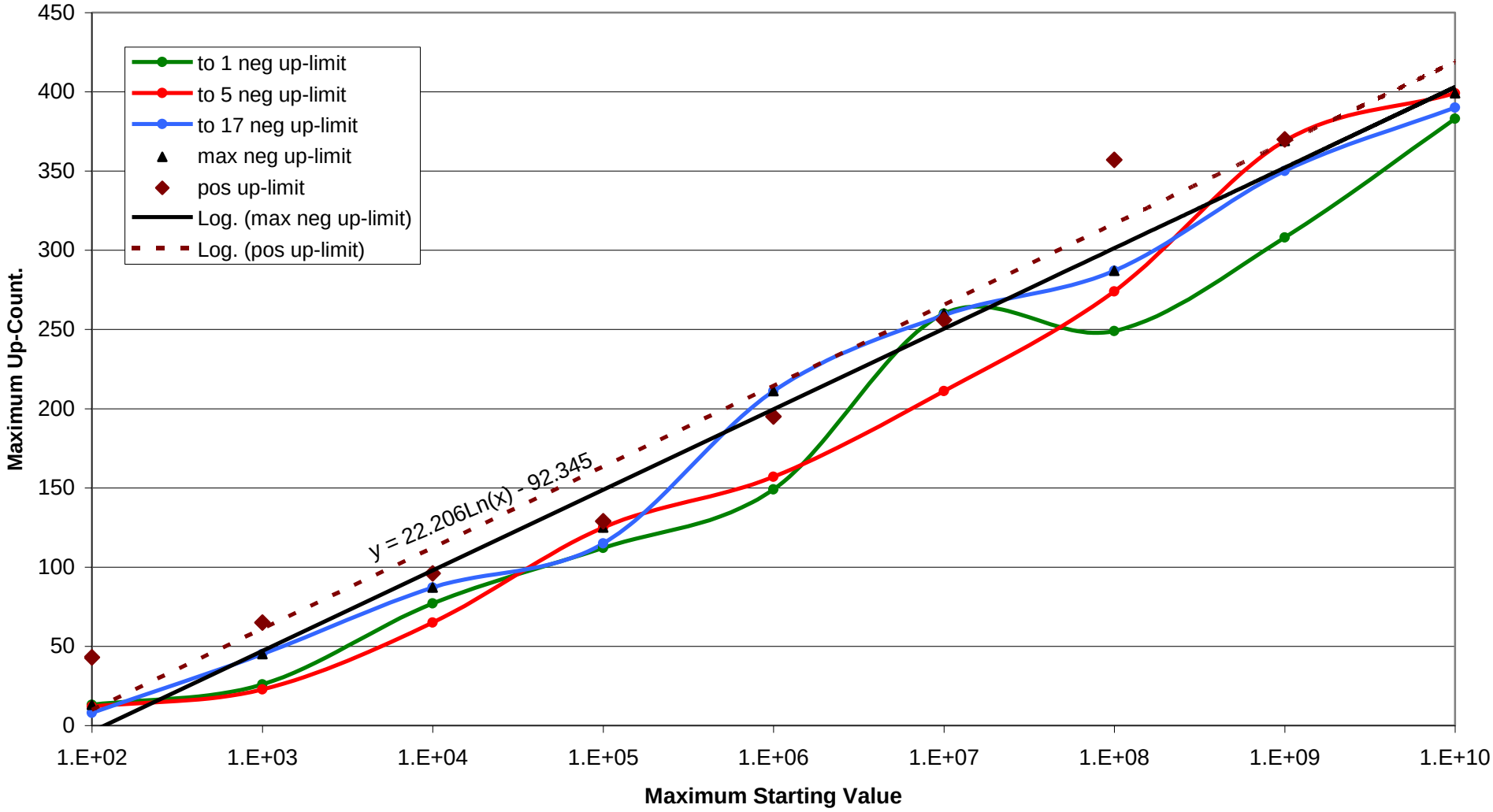
But we have to admit to 'cheating'. We have hard-coded terminations when the values 1, 5, and 17 are reached. These are known for ⁻Collatz, but unknown for ⁺Collatz. We need a reliable method of finding loops or infinite-iteration starting-points without using known values. This is easily achieved by using an iteration up-count limit, the exact value of which is relatively unimportant. The point is we can tweak the value upwards if it causes incorrect reporting.

We show some iteration up-counts on the next page. Testing above starting values of 1E9 is boring as the run-time exceeds 1 day. We use the simple empirical approximation formula for a typical limiting up-count:

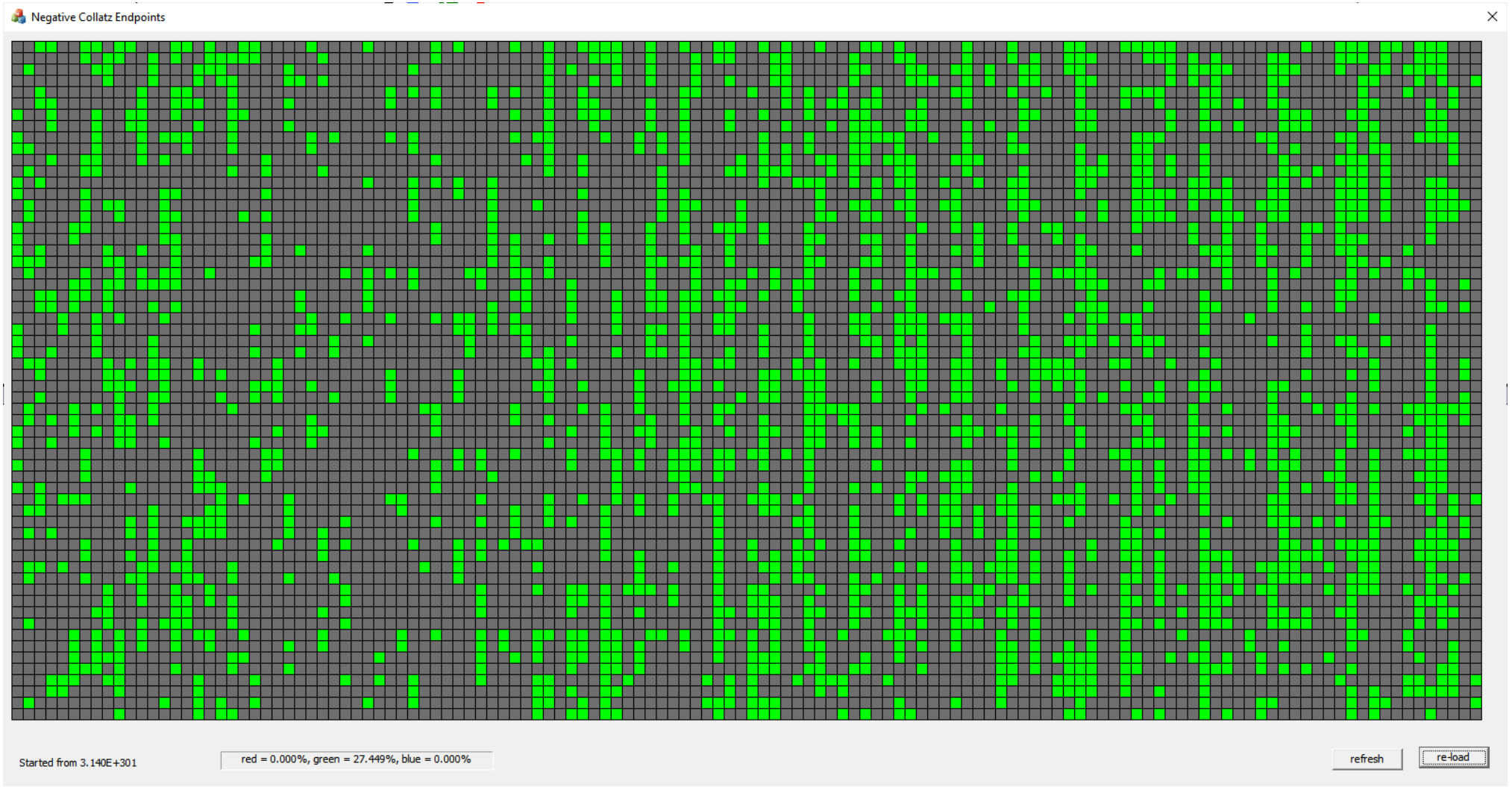
$$\text{upCount} = 50 \times \log_{10}(\text{value}) - 100$$

Then we arbitrarily add a 1000 to that value as a margin. This only affects the run-time on the ⁺Collatz tests, as it takes longer to reject "infinite loops" when we don't know it is actually a (known) loop.

±Collatz Iteration Limits



Collatz Outcome Map: Starting from 3.14E301



Here we make no prejudgements about the existence of loops (or their exact values). Only 27% of starting values terminate.

⁺Collatz Outcomes

Displaying outcome maps for ⁺Collatz sequences is redundant until or unless we find counter-examples to the always terminating sequence. We therefore can try a bit of manual testing, just changing the -1 part to a +1 part (one line of code) in the generator.

Upsteps is the maximum number of upsteps used for any particular number within the tested range, that is from the start value upwards by 7800.

3.14E11 (210 upsteps), 3.14E21 (370 upsteps), 4.32E31 (380 upsteps)
2.12E42 (430 upsteps), 7.13E53 (610 upsteps), 4.17E71 (630 upsteps)
1.11E307 (2600 upsteps), 1.12E307 (2800 upsteps),
1.13E307 (2200 upsteps), 1.14 (2900 upsteps)

These tests take only a second (for the lower values) so we can easily do loads more to improve our sampling process, and improve confidence. It is also easier to loop the tests, giving an importable table, rather than manually typing in the values.

To improve confidence, instead of 7800 points per start value, the amount has been raised to 100,000 points per start value. Obviously this has a knock-on effect for the run time, but even so the 100 randomly selected start points were dealt with in around 2 hours.

The random mantissa and exponent were picked separately in a loop with **p** ranging from 1 to 100:

```
double m = 1.0 + (p % 10) + double( rand() % 1000 )/ 1000.0;
double e = 30.0 + (rand() % 275);

double dStart = m * pow(10.0, e);    // exponent limit is ±308
```

Needless to say, no non-terminating start values were found.

start value	upsteps	time		start value	upsteps	time
2.04E+72	849	35.0 s		2.70E+191	1699	93.0 s
3.33E+130	1067	57.0 s		3.32E+113	1056	60.0 s
4.17E+79	840	39.0 s		4.67E+294	2682	2.6 mins
5.48E+238	2075	1.9 mins		5.14E+41	503	22.0 s
6.96E+294	2594	2.4 mins		6.25E+298	2553	2.6 mins
7.71E+125	1283	60.0 s		7.55E+174	1687	90.0 s
8.28E+82	674	37.0 s		8.66E+62	637	32.0 s
9.96E+246	2269	1.9 mins		9.04E+239	2184	2.2 mins
1.10E+148	1366	74.0 s		1.07E+147	1481	75.0 s
1.83E+241	2119	2.1 mins		1.53E+258	2125	2.1 mins
2.39E+59	589	29.0 s		2.32E+40	543	21.0 s
3.90E+183	1602	89.0 s		3.19E+222	1825	1.9 mins
4.29E+37	509	20.0 s		4.29E+161	1475	75.0 s
5.42E+46	512	21.0 s		5.04E+172	1530	92.0 s
6.72E+125	1104	64.0 s		6.26E+128	1270	58.0 s
7.45E+31	545	16.0 s		7.45E+185	1633	93.0 s
8.77E+293	2558	2.4 mins		8.89E+159	1440	73.0 s
9.87E+142	1275	67.0 s		9.37E+255	2150	2.1 mins
1.07E+205	1620	95.0 s		1.00E+57	623	30.0 s
1.04E+299	2524	2.5 mins		1.39E+278	2323	2.3 mins
2.63E+278	2400	2.3 mins		1.01E+188	1633	89.0 s
3.08E+184	1564	93.0 s		1.93E+71	682	33.0 s
4.76E+45	583	23.0 s		2.83E+70	745	36.0 s
5.97E+256	2234	2.1 mins		3.64E+263	2503	1.9 mins
6.93E+213	2107	1.7 mins		4.70E+60	749	32.0 s
7.94E+294	2580	2.4 mins		5.98E+136	1216	70.0 s
8.63E+78	842	41.0 s		6.67E+141	1150	69.0 s
9.54E+118	1174	57.0 s		7.02E+175	1456	88.0 s

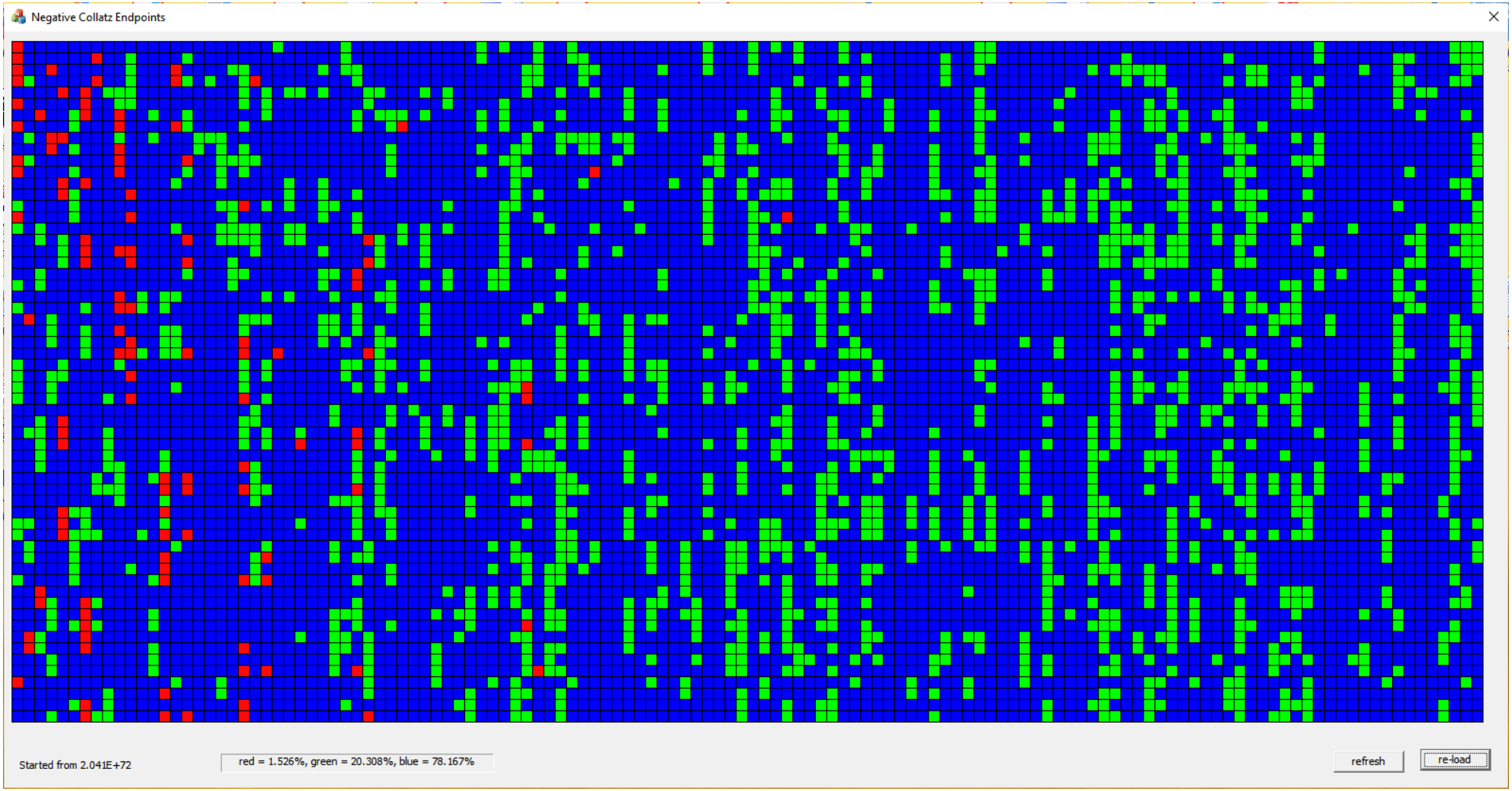
start value	upsteps	time		start value	upsteps	time
8.92E+127	1114	59.0 s		8.51E+135	1204	67.0 s
9.27E+84	866	36.0 s		9.41E+48	583	24.0 s
1.08E+204	1935	1.7 mins		1.09E+172	1633	79.0 s
1.10E+42	549	20.0 s		1.76E+35	485	19.0 s
2.99E+120	967	56.0 s		2.41E+64	687	32.0 s
3.16E+241	2037	1.9 mins		3.62E+217	1966	1.9 mins
4.36E+47	524	23.0 s		4.55E+188	1590	95.0 s
5.66E+204	1815	98.0 s		5.60E+221	1973	1.8 mins
6.03E+257	2247	2.3 mins		6.60E+180	1496	90.0 s
7.35E+80	867	39.0 s		7.29E+66	726	26.0 s
8.94E+304	2504	2.5 mins		8.37E+50	524	21.0 s
9.97E+160	1605	84.0 s		9.60E+126	1236	62.0 s
1.01E+247	2069	2.1 mins		1.03E+130	1128	68.0 s
1.01E+92	913	47.0 s		1.67E+39	459	20.0 s
2.46E+217	1909	1.8 mins		2.28E+89	781	40.0 s
3.75E+238	1874	1.9 mins		3.05E+104	878	46.0 s
4.95E+139	1350	69.0 s		4.42E+193	1921	89.0 s
5.21E+163	1706	76.0 s		5.90E+243	2036	2.0 mins
6.22E+193	1730	90.0 s		6.13E+222	1832	1.7 mins
7.42E+226	1896	1.9 mins		7.73E+73	746	37.0 s
8.65E+238	2048	2.0 mins		1.03E+48	522	21.0 s
9.81E+251	2263	2.1 mins		1.81E+194	1647	1.7 mins

We can't convert directly from double to mpz_t much above 1E308 (which is the double limit), but we can just set a high order bit (in addition to the double value) to boost the range. For example 2^{7000} = 1.621E2107. In order to keep the iteration time to a more reasonable value, the maximum number of values above the starting value has been reduced back down to 7800.

set bit	start value	upsteps	time
2^{7001} +	2.04E+72	17035	90.0 s
2^{7002} +	3.33E+130	16240	90.0 s
2^{7003} +	4.17E+79	17035	91.0 s
2^{7004} +	5.48E+238	17035	88.0 s
2^{7005} +	6.96E+294	17035	90.0 s
2^{7006} +	7.71E+125	17035	90.0 s
2^{7007} +	8.28E+82	16240	91.0 s
2^{7008} +	9.96E+246	17035	90.0 s
2^{7009} +	1.10E+148	16240	92.0 s
2^{7010} +	1.83E+241	17035	95.0 s
2^{7011} +	2.39E+59	16240	91.0 s
2^{7012} +	3.90E+183	17035	92.0 s
2^{7013} +	4.29E+37	17035	91.0 s
2^{7014} +	5.42E+46	17035	92.0 s
2^{7015} +	6.72E+125	17035	94.0 s
2^{7016} +	7.45E+31	17151	93.0 s
2^{7017} +	8.77E+293	17035	93.0 s
2^{7018} +	9.87E+142	17035	92.0 s
2^{7019} +	1.07E+205	16240	91.0 s
2^{7020} +	1.04E+299	17035	91.0 s
2^{7021} +	2.70E+191	17035	92.0 s
2^{7022} +	3.32E+113	17035	95.0 s
2^{7023} +	4.67E+294	17035	95.0 s
2^{7024} +	5.14E+41	17151	94.0 s
2^{7025} +	6.25E+298	17035	96.0 s
2^{7026} +	7.55E+174	17151	93.0 s
2^{7027} +	8.66E+62	17035	92.0 s
2^{7028} +	9.04E+239	17151	95.0 s
2^{7029} +	1.07E+147	17035	94.0 s
2^{7030} +	1.53E+258	17035	94.0 s

⁻Collatz Outcome Map: Starting from 2^{7001} + 2.040E72

For completeness we also show a ⁻Collatz outcome map up at the ridiculously high value of 3.24E2107. All three known outcomes are visible. (The display software does not know about the leading power of two.)



Conclusion

A careless reader, just skimming the surface, may conclude that all we have done here is to say that having sampled the ⁺Collatz sequence at a relatively few points, we have concluded that since all the points tested did terminate, then all other points are also likely to terminate. Such a conclusion totally misses the point.

We have known since the earlier paper that any loop must necessarily create an infinite amount of starting points above that value which also loop. We have further seen here that a single point which iterates off to infinity must also create an infinite amount of starting points above that value which eventually iterate off to infinity. The difficulty is quantifying the relative sizes of these infinities, given that the Structure table is also an infinite construct.

Looking at the earlier paper, the analytic summation of the size of the stray table looked intractable, and was not seriously attempted, although the computer plot of the relative sizes of the Levels shows that increasing even Levels hold a much greater proportion of starting values as the starting values are increased.

Here we have seen experimentally from the ⁻Collatz sequence that the loops generate infinite capture areas above themselves which are a fixed fraction of the total starting values up to 8 orders of magnitude above the lowest loop limit, and there is evidence that these infinite sets do not 'dwindle out', even up to 1E2000. For the ⁻Collatz sequence in particular we have 3 disjoint infinite sets of termination outcomes from particular starting values which completely fill the available number space. We find this result fascinating, and would hope to be able to approach this finding analytically in the future with something other than the argument concerning the symmetry between the structure table and the stray table.

In the earlier paper it was frustrating that the Structure table and the Rank table could not be fully integrated to show that the Structure table contained all values in the Rank table, and therefore had full coverage over the natural numbers. Having seen a similar Structure table in ⁻Collatz and a similar Rank table in ⁻Collatz, a sequence which does have loops, it is clear that the Structure and Rank tables alone cannot make the termination of the Collatz sequence decidable.

Whilst analytically counting the starting values in any Level is hard, computationally it Is not difficult. The new plot of values per Level shows that whilst there are few start values in L0, as you increase the starting value, the proportion of starting values in the higher Levels increases significantly. The shape of the values per Level plot is remarkably similar to that for the ⁺Collatz sequence.

It is the conclusion of this paper that sampling the ⁺Collatz sequence below the lowest value of any loop or infinite-iteration start point has no chance of finding any such loop. (This is self-evident). However, and this is the key point, sampling the ⁺Collatz sequence above any loop or infinite-iteration start point has a **significant** chance of finding such a point.

Having found no evidence of loops up to 1E2020, 2000 orders of magnitude beyond the current exhaustive testing limit, we should reasonably conclude that any loops must either be beyond this value, or not exist.

There is then this niggling doubt that maybe the loop does not start until 1E2030. There is not much we can do about that other than to just go mental and set bit 123456 in the mpz_t starting value. 2^{123456} is a number which has 37,164 decimal digits according to the (awesome) Alpertron calculator. 1E37163 is surely large enough for all except the most serious number crunchers! The run-time was 76.4 minutes for 10,000 points, and gave an up-count maximum of 297,145.

The probability of randomly finding 1, 10, 100, or even 1000 looping points is negligibly small when considering a range up to 1E20. But we are looking at a fractional (infinite) number of looping points amongst another infinite set of starting values. It is this fractional density aspect which makes the argument compelling, and this fractional aspect is backed up by symmetry between the structure table and any stray tables, as exemplified by the negative Collatz sequence. The statistical absence of evidence in this case is in fact evidence of absence.

If loops or infinite-iteration start points existed, we should have found them using this method. We therefore conclude that the $^{+}$Collatz sequence always terminates for values with practical significance.

v1.10: An additional test was done using $2^{1234567}$ on top of the 2.041×10^{72} starting value. This gives starting values above **4E371641** according to the Keisan calculator,[6] the Alpertron calculator having run out of digits at that level. 5 instances of the test program were run, each taking its own block of 2000 points, in order to get the full 10,000 point run 5× faster. Using a 6-core Xeon E5-1550 v4 @ 3.6GHz the run-time (per instance) was 24.6 hours. The maximum up-count was **2,969,190**.

v1.20: A new verification collection including test code, test results, and paper with analysis of results has been created as a zip file, and this will be updated as new data becomes available. Currently, with high confidence, we claim that no Collatz counter-examples exist for starting values with up to 3.7 million digits. Future updates to this limit will be included in "*Experimental Evidence for the Claim of Collatz termination*".[7]

[6] https://keisan.casio.com/calculator
[7] http://lesliegreen.byethost3.com/articles/evidence.zip

Acknowledgement

This paper would not have been possible without the tremendous work of the MPIR team, and before them the GMP development team. Their collective work is much appreciated.

Version History

v1.25: 14 Aug 2022. Minor textual changes to emphasise the importance of the symmetry between the structure table and the stray table in the positive Collatz testing.

v1.20: 4 Aug 2022. Add the plot "Even Levels of –Collatz." New wording in the Conclusion, and strengthened analysis to include important aspect of symmetry between the structure table and the stray table(s). Delete section, ***Further Work***.

v1.10: 31 Jan 2022. Included updated upper test limit data so instead of just 37,163 digits, we also now have 371,641 digits! New data on page 28, and in the abstract. Added missing Acknowledgement to MPIR/GMP teams.

v1.00: 28 Jan 2022. First publication on

http://lesliegreen.byethost3.com