Random Fractal Filling of a Line Segment

ABSTRACT: An algorithm which produces complete random fractal filling of a one-dimensional line segment by ever-shorter line segments is described. It can be viewed as a novel form of stochastic process. It is shown that the algorithm does not halt for a substantial range of its parameters.

1. Introduction.

In Fig. 1 the dark regions are line segments that have been randomly placed within a "container" line segment. The bottom bar shows the structure at 1x scale, whence it can be seen that the container segment is mostly full. Other bar graphs show the left end progressively expanded, revealing ever-finer detail there.

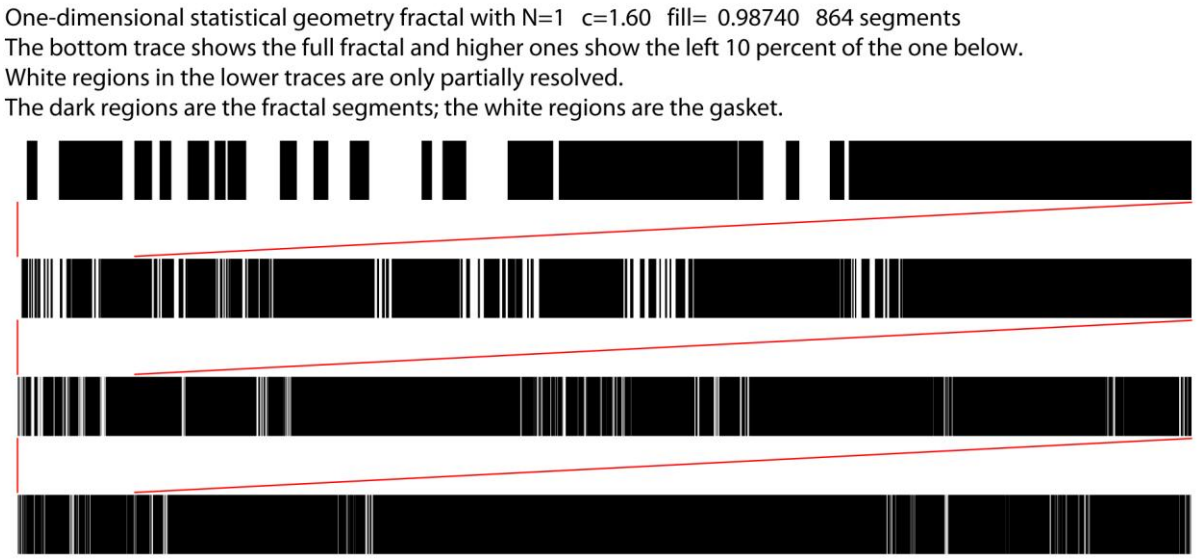

Fig. 1. An illustration of the fractal with $c = 1.6$ and $N = 1$. Several scales of detail are shown so that the reader can better grasp its form.

How is this done? We progressively generate segment lengths labeled by an integer n ($n = 1, 2, \ldots$) which follow a power law in the sequence number, i.e., segment length ~ $1/(n+N-1)^c$ where c is a fixed exponent with values in the approximate range $1 < c < 5$. The first segment is placed at random at a location entirely inside the container segment. The following ever-shorter segments are given random "trial" positions anywhere within the container segment and tested to see if at the trial position the segment overlaps any previously placed segment. If it does, another trial is made, continuing until a nonoverlapping place is found, at which point we have a "placement" and n is incremented. A concise statement of the algorithm follows in the next section.

Extensive numerical work has shown some surprising but quite persistent results. (1) By construction the process is space-filling. (2) Within a substantial range of c values, the random search algorithm never halts, although it slows down. (3) The process works for a variety of shapes. (4) There is an upper limit c_{max} for c. Evidence for these claims is presented below.

2. The Algorithm Stated.

Let L be the length of the container segment. We define a sequence of segment lengths L_n (n = 1, 2, …) given by the rule

$$L_n = \frac{L}{\zeta(c,N)(n+N-1)^c} \qquad (1)$$

Where $\zeta(c,N)$ is the Hurwitz zeta function defined by

$$\zeta(c,N) = \sum_{i=0}^{\infty} \frac{1}{(i+N)^c} \qquad (2)$$

This is known to converge when $c > 1$ and $N > 0$. In view of Eq. (2)

$$\sum_{n=1}^{\infty} L_n = \sum_{n=1}^{\infty} \frac{1}{\zeta(c,N)(n+N-1)^c} = \sum_{i=0}^{\infty} \frac{1}{\zeta(c,N)(i+N)^c} = L \qquad (3)$$

so that the sum of all the lengths L_n is the length L to be filled, that is, *if the algorithm does not halt it is space-filling* by construction. The parameters c and N can be chosen over a substantial range of values. Parameter N need not be an integer, but an integer is used in the examples presented here.

Step 1. Let n = 1. Place a segment with length L_1 at a random position within the container segment to be filled such that it does not overlap the boundary. Place the *position and length* of the segment in the placed-shapes database. Increment n. This is the *initial placement*.

Step 2. (iterative). Choose a random position for a new segment with length L_n, again entirely within the boundary of the container segment. This is a *trial*. Does segment L_n overlap with any previously-placed segment? If it does then repeat Step 2. If not then place the *position and length* of the segment in the placed-shapes database, increment n, and repeat Step 2. This is a *placement*. Stop when a given number n of placed shapes has been achieved or a chosen percentage fill has been reached.

In the random trials all positions x are equally probable.

The result is a random fractal space-filling pattern of line segments within a fixed container segment whose areas follow a power-law sequence, as shown in Fig. 1. All of the computations have been done with double-precision arithmetic.

The parameters c and N can have a variety of values. When N = 1 the parameter c must be in the range $1 < c < c_{max}$ where c_{max} is about 2.7. The failure of the process at the high-c limit is not a sudden, sharp event but occurs gradually as c increases. For higher N values, c_{max} can be > 2.7.

By construction the result is a space-filling random fractal -- *if the process never halts*[1]. It can be demonstrated (see sec. 11) that it does not halt when $1 < c < 2$ if $N = 1$. As the algorithm progresses, the fill factor comes arbitrarily close to 1, while the number of ever-narrower intervals between placed segments grows without limit.

The fractal dimension D is given by

$$D = \frac{1}{c} \qquad (4)$$

This agrees with box-counting determinations of fractal D for numerical examples made by Paul Bourke [7].

3. Unique Features of the Algorithm

- By construction it is space-filling if it does not halt.
- Because of the power law the result is fractal.
- It is random, not deterministic.
- It is not recursive.
- It is not a single fractal but a family of fractals with parameters c and N.
- The placed segments are non-overlapping and non-touching.

The author is not aware of any other fractal which has the same properties. The algorithm also works in two and three spatial dimensions [7-8].

4. Boundaries.

The rules say to choose a container length L. There are two ways to define its boundaries. One way (inclusive boundaries) is to simply require all of the segments to fall completely inside the end points. Another way (periodic boundaries) is to allow segments to cross the boundaries but insist that they be periodic, i.e. if a segment lying between x_1 and x_2 crosses the boundary, another identical segment must be included that is placed at $x \pm L$. Both methods give satisfactory results.

5. Effect of the Parameter *c*.

For large c the segment lengths decrease more rapidly with n. The average distance between segments gets smaller as c increases, i.e., the packing is tighter. The closer segment-to-segment spacing and the smaller

[1] There are two ways to consider the halting question. One can ask whether the *computational algorithm* using floating-point numbers stops. Here the answer is probably yes, since there is a smallest feature size of 1 least-significant bit. Or one can ask whether the algorithm stops for ideal mathematical numbers, which have infinite resolution (one can think of them as floating-point numbers with infinite word length). The computational data suggest that the algorithm does not stop when using ideal numbers, but this is a long way from a rigorous proof.

dimensionless gasket [5] width mean that more trials are needed with high c for the placement of the i-th segment. It is also seen that c correlates with order and disorder; the arrangement is quite random for low c, and becomes progressively more ordered for higher c.

6. Run-Time Behavior.

What happens as the algorithm is executed? In particular, how many trials does it take to place some number of segments? How is this behavior affected by c? To this end one can plot $log_{10}(n_{cum})$ versus $log_{10}(n)$ where $n_{cum}(n)$ is the total number of trials needed to place n segments. Such a record will be different for each run.

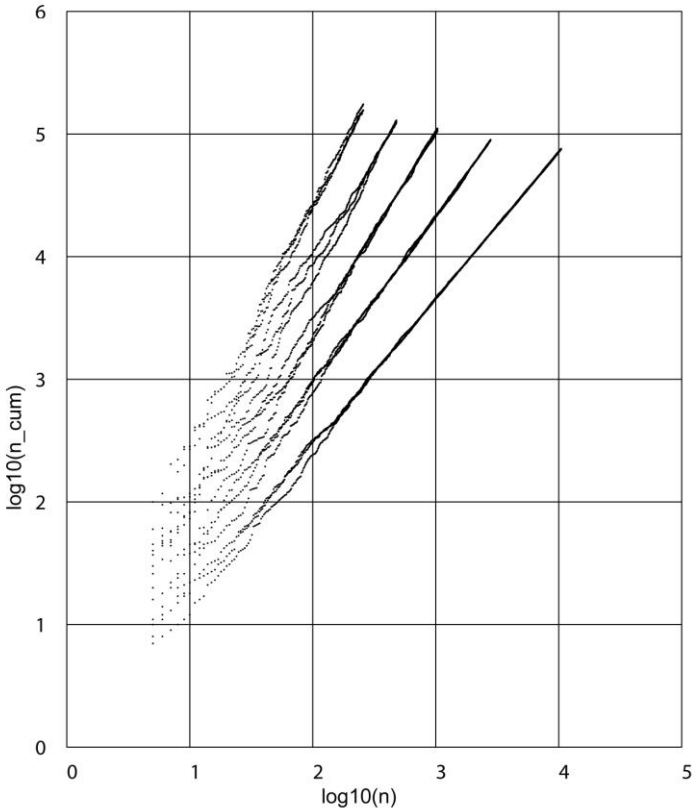

Fig. 2. Run-time records. $N = 1$ in all cases. The vertical coordinate is $log_{10}(n_{cum})$ where n_{cum} is the cumulative number of trials needed to place n segments. The horizontal coordinate is $log_{10}(n)$. From bottom to top, c = 1.2, 1.4, 1.6, 1.8, and 2.0. Three runs are plotted for each c value.

For large n, the data follows a straight line, showing that $n_{cum}(n)$ follows a power law in n, i.e., $n_{cum}(n) = Kn^{\phi}$. The parameters K and ϕ can be estimated from the data. They will have an uncertainty associated with the randomness of the process. One can calculate an expected number k of trials that will be needed for any number n of segments to be placed. This is one basis for saying that the process does not halt. Although k may be huge, it is finite.

For all of the c values on the graph, $\phi \approx c$ within statistical error.

The data becomes quite noisy for large c. It is thought that this noise sets the upper limit for usable c values.

7. Order and Disorder, a Digression into 2D.

Figure 3 shows fractalized triangles for several c values. Here $c_{max} \cong 1.25$. Near c_{max} you see a quite orderly arrangement of the triangles, especially the small ones, with the vast majority of small triangles having three near neighbors. As c falls, the arrangement becomes more noisy and random.

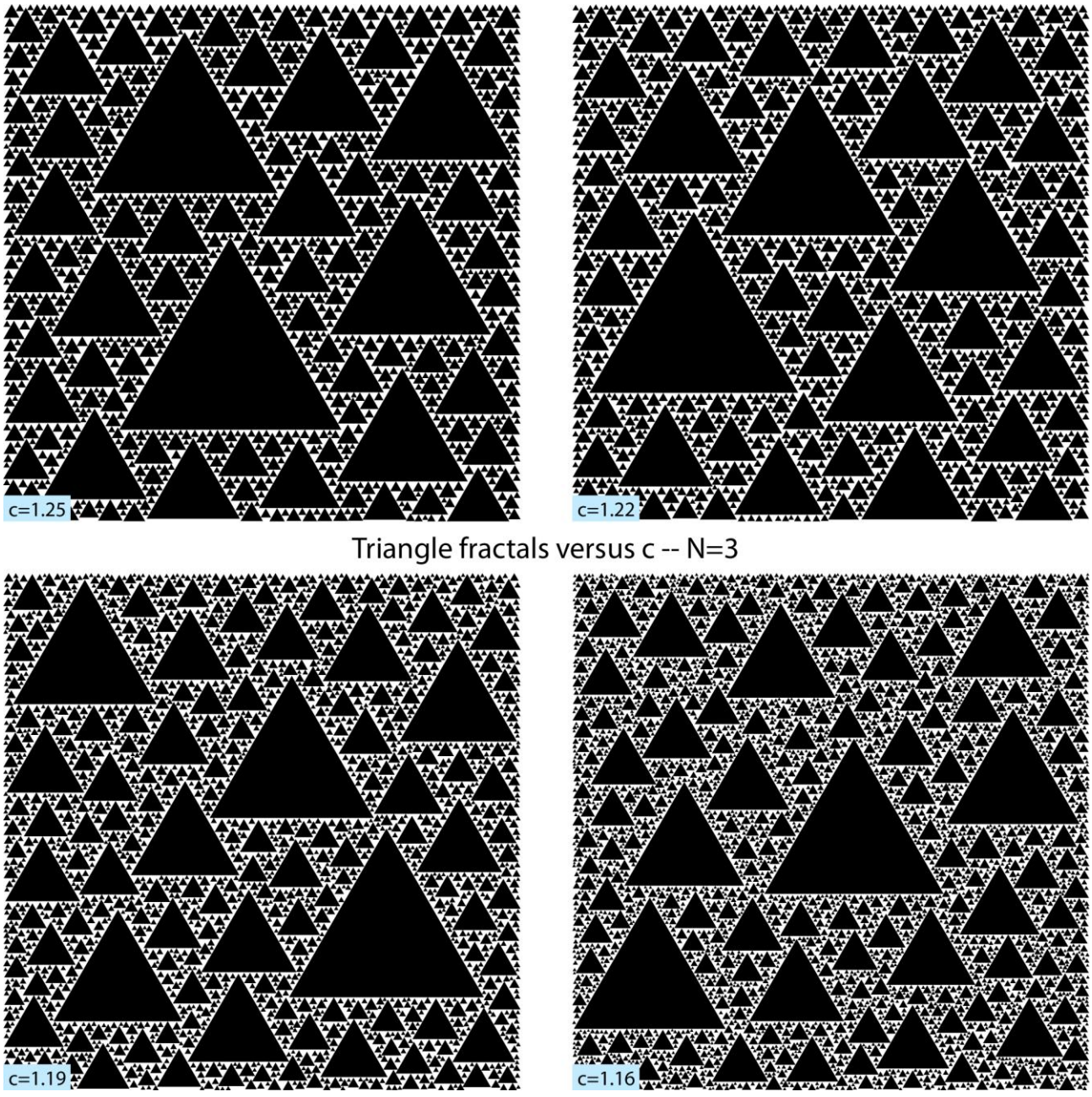

Fig. 3. Triangle fractals in 2D for various c values.

The increase of c is accompanied by a huge increase in the number of trials needed for a placement. What then limits c from above? The claim is that *c cannot go past the limit of perfect order*. And what is that?

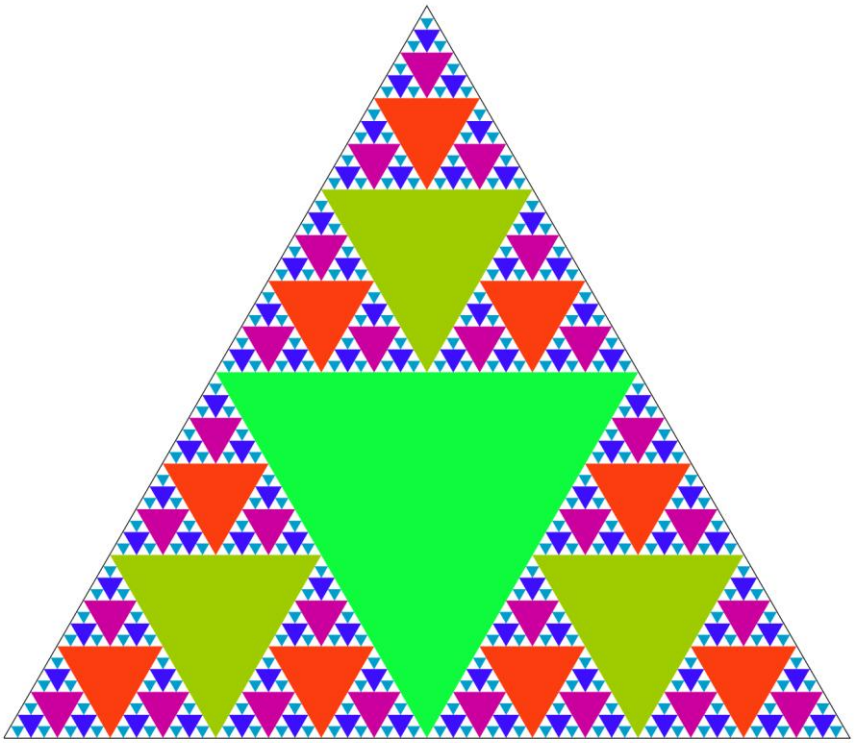

Fig. 4. A Sierpinski construction made by recursively adding triangles to an empty triangle. It is an example of a 2D fractal with perfect order (no randomness). The local arrangement of the smaller triangles is quite similar to the case c = 1.25 in Fig. 3. Each recursive generation has its own color.

The fractal of Fig. 4 represents the perfect-order case for triangles. It is well known to have the fractal D value $D = log(3)/log(2) = 1.585$. The observed high-$c$ limit of the random triangle fractal gives $D = 2/1.25 = 1.600$. If we view c as a measure of order and disorder we see c going up (and D falling) until "just above" the perfect-order value $D = 1.585$ (which would correspond to $c = 1.262$). This is a long way from a proof, but an interesting line of reasoning and a plausible numerical near-coincidence. It should not be taken from this that the algorithm will, if pushed hard enough, yield the Sierpinski construction shown. A random process of the kind that interests us cannot "bridge the gap" to perfect order. It can only approach it.

8. The c_{max} Values in 1D.

The same order-disorder variation with c described above also occurs in 1D but is harder to visualize. For simple segments we find from computational experiments to date that $c_{max} \approx 5$. This suggests that there is a limiting fully-ordered fractal with D a little below $D = 1/5 = 0.2$.

The recursive fractal picture for this case is complicated. Figure 5(c) shows a fairly obvious way of creating a recursive fractal with single segments. One can define the procedure as one where a segment of length g times the gap is inserted at each recursive generation. If this is done the length change from one generation to

another is (1–*g*)/2, so that $D = log(2)/log(2/(1- g))$. A problem with this is that it creates a family of fractals with parameter *g*, which have fractal *D* values all the way down to zero. Which one really sets the "complete-order" limit? On the positive side, these fractals have an extremely sharp drop in the element size which is in fact what we see with the random fractals. If the computationally observed c_{max} is ≈ 5, the corresponding *g* value would be > 32, and the length ratio *g* per recursive generation would be ≈ 0.94.

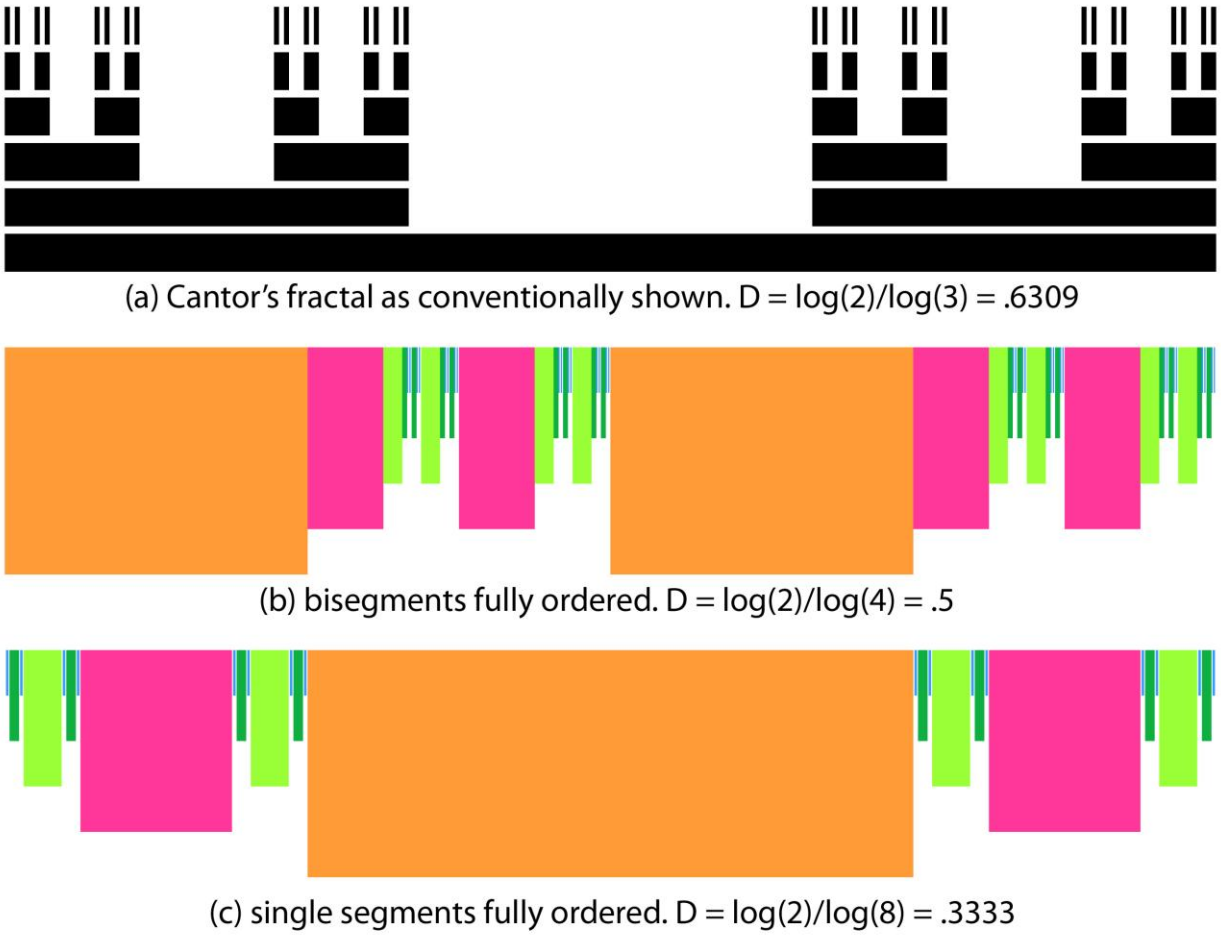

Fig. 5. Some recursive one-dimensional fractals. For (b) and (c) the recursive generations are given different colors. In (b) and (c) the convention is followed that new elements are added to an empty line segment, unlike the Cantor case.

9. Bisegments.

One might think that one can only have one shape in one dimension. Since the algorithm does not require that the shapes be contiguous, it is in fact possible to have shapes with 1, 2, or more distinct segments. A study was done with a "bisegment", where one has two segments of width *a*, separated by gap of width *a*.

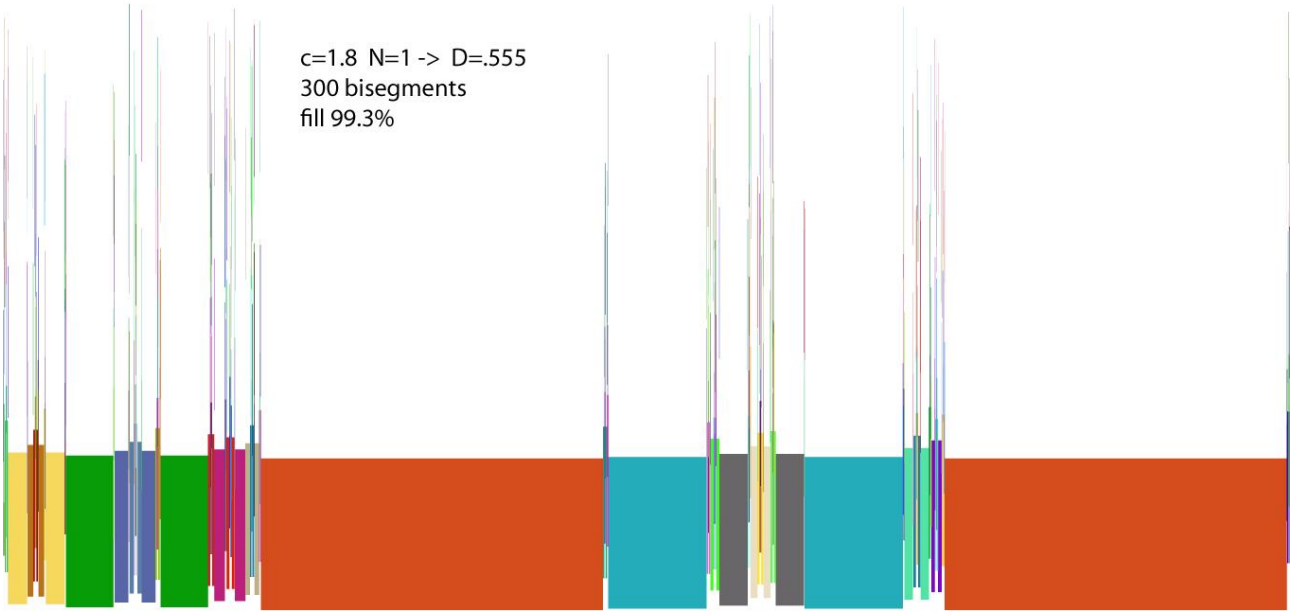

Fig. 6. Fractalized bisegments. The use of color allows one to see the distinct parts of the same bisegment. Each added bisegment is drawn a little higher up so that one can see traces of even the ultra-thin ones. Random colors, inclusive boundary.

The algorithm ran smoothly for bisegments. According to the computational studies the case shown is close to c_{max}. Close study of Fig. 6 shows a high degree of order, and one readily sees "nesting" of bisegments within the "holes" in previous bisegments. This nesting indicates a *highly-ordered* arrangement.

What limits c to fall below c_{max}? Figure 5(b) shows how bisegments can be fractalized recursively to create a bisegment fractal with perfect order which would limit c and D. One can observe that in Fig. 5(b) the nesting is complete and ordered. We see that the D value (D = .555) which follows from c_{max} = 1.8 is just above the fractal D of the perfect-order case.

Other bisegments are possible with different ratios of the two segment lengths and the segment-to-segment spacing. They will have different c_{max} values.

10. The Statistics and Dynamics of the Process.

The process begins by placing segment 1 at random. This also creates two vacant intervals within the container segment, which can be filled during further placements. (If periodic boundaries are used there is one vacant interval after the initial placement.) Since each new placement eliminates one interval and creates two new ones, there are (n + 1) vacant intervals after placement of segment n. The database at any point in the process contains the locations of all placed segments, and their lengths. It is thus possible to compute a complete set of vacant interval lengths after placement n. When this is done the intervals can be further sorted into those too narrow to accommodate the next-to-be-placed segment length (L_{n+1}), and those sufficiently wide. We will call the sufficiently wide ones "gaps". Thus at any point in the process there is a number n_{gap} of available places to put the next-to-be-placed segment. This provides a very sharp criterion for halting of the algorithm. *The algorithm halts if* $n_{gap} = 0$. If $n_{gap} > 0$ for all n the algorithm will continue to run and will eventually succeed for all n, even if an astronomical number of trials is needed.

The calculations reported here were done with inclusive boundaries.

(a) <u>The Distribution of Intervals</u>.

The histogram in Fig. 7 was constructed by running the algorithm repeatedly with a fixed c, N, and n until data for 100 cases had accumulated. This gives good resolution. If continued "to infinity" this would define a continuous function which is the probability distribution function (p.d.f.) for interval length. It can be seen that this is a falling exponential-like function with a large tail beyond the length of the next-to-be-placed segment. For these parameters the process is unlikely to halt (in fact all 100 examples ran smoothly).

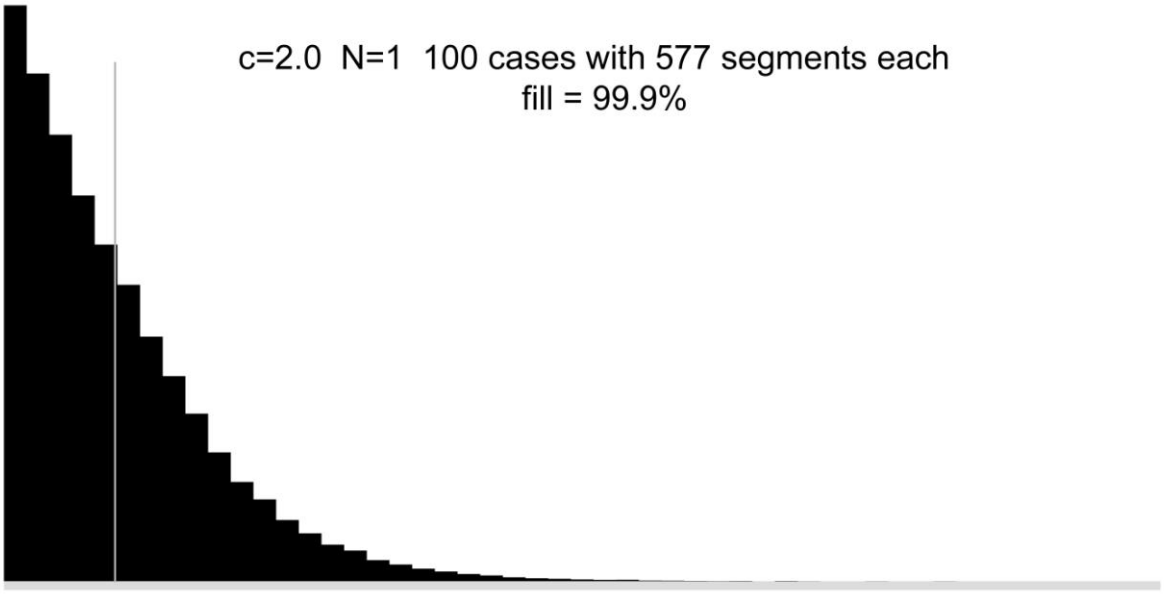

Fig. 7. A histogram of the interval widths. The vertical coordinate is the number of instances. The gray line underneath shows the extent of the data. The largest gap lies in the farthest-right bin. The vertical gray line shows the width of the last-placed segment on the same x scale.

The "same at all scales" principle of fractals suggests that this distribution function, *when the x-axis is scaled to the value of the last-placed segment*, may be universal, i.e. the same at all n (or the same at all n when n is large).

There is a sparse population of quite large gaps, which are improbably numerous if this is truly an exponential p.d.f.

(b) Trials and Placements.

We denote by $n_{cum}(n)$ the cumulative number of trials needed to place n segments. Because of the randomness, it is different for every run, but it shows distinct statistical trends. We define $n_{tri}(n)$ to be the number of trials needed to make placement n. It has a very noisy behavior. Figure 8 plots these quantities on a log-log scale.

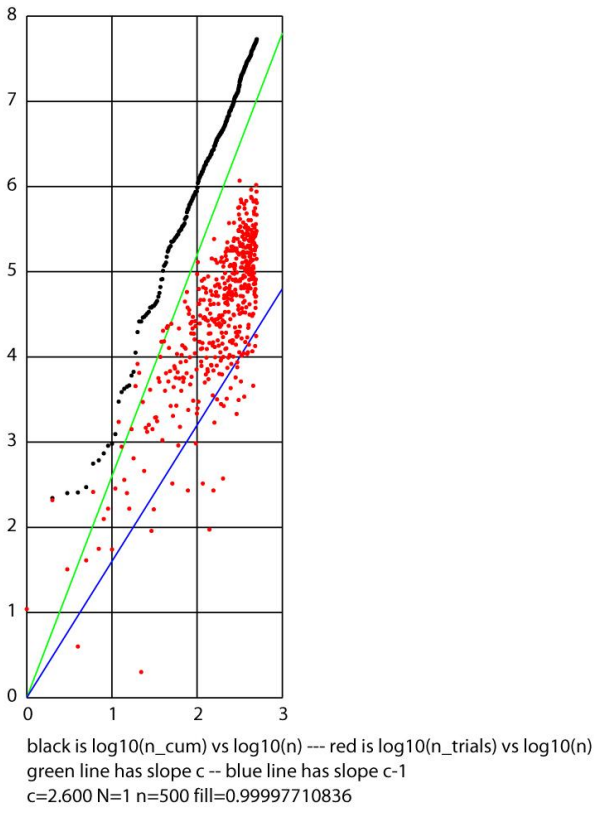

Fig. 8. Trials. The black points are $log_{10}(n_{cum})$ versus $log_{10}(n)$. The red points are $log_{10}(n_{tri})$ versus $log_{10}(n)$. The green line is a reference line with slope c, while the blue one is a reference line with slope $(c-1)$. The numbers labeling the grid lines are powers of 10, e.g., 3 corresponds to 1000.

The process is quite noisy for small n, but as n increases the cumulative trials n_{cum} settle down to a power law which is reasonably well approximated by an exponent c (i.e., the black points make a line parallel to the green line). After the process has settled into a steady state the log slope always agrees reasonably well with

c, for all *c* and *N* values studied. There are placements at about $log_{10}(n) = 1.3$ and $log_{10}(n) = 1.6$ where the black points go up very steeply, indicating that an unusually large number of trials were needed there. Such behavior is commonly seen in such records, especially for large *c* values where many trials are needed for each placement.

The n_{tri} data is extremely noisy. Looking at the red points toward the right of the graph is it seen that there is a point where $log_{10}(n_{tri}) > 6$, i.e., it took more than a million trials to make a placement. Since the rules of statistics say that if the cumulative distribution follows a given functional form the differential distribution is the derivative *d/dn* we would expect the averaged n_{tri} curve to have a slope $(c-1)$ (blue line) and this does not contradict the very scattered data.

(c) Gaps.

As stated above, by a gap we mean an interval between placed segments which is large enough to hold the next-to-be-placed segment. The number of gaps (n_{gap}) will vary as the process continues.

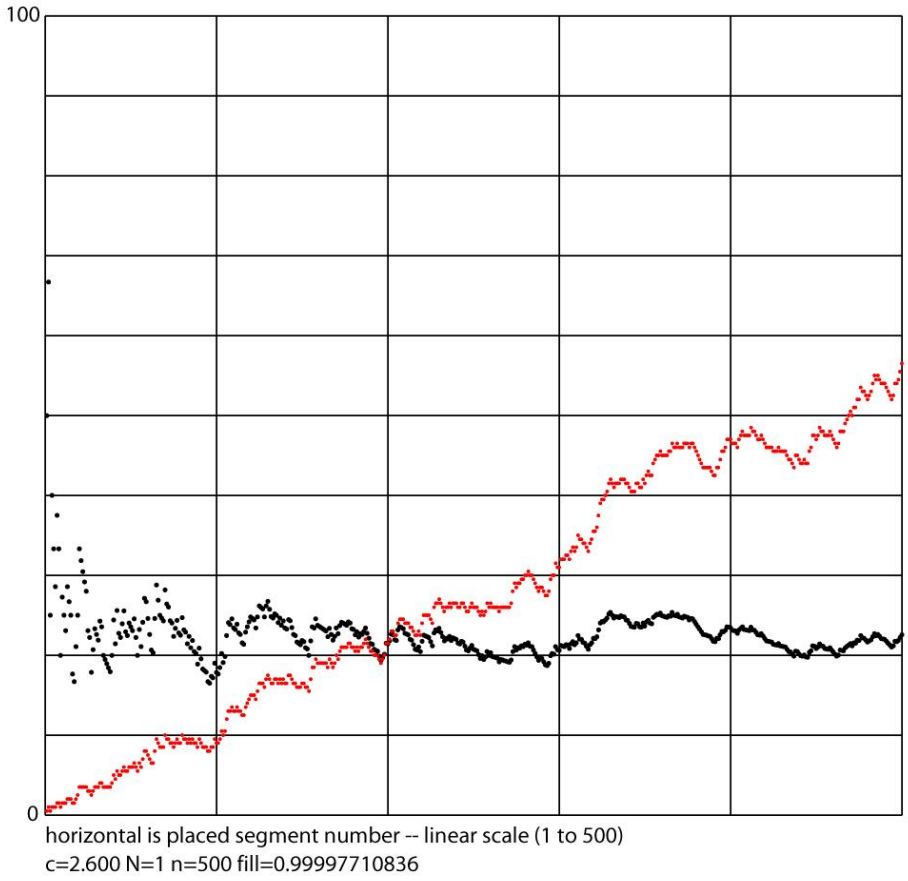

Fig. 9. Gaps. The red points are n_{gap} values (integers). The black points are $100 \cdot n_{gap}/(n+1)$, i.e. the percentage of gaps relative to all intervals. The horizontal scale is placement number *n*, and is linear.

It can be seen that n_{gap} climbs steadily. It thus is quite unlikely that the red curve will ever dip down to zero (i.e., failure or halting), which is an indication that for these parameters the process does not halt. There is mild oscillatory behavior, but it appears to be damped and not associated with instability.

The fraction of intervals which are gaps settles down to a steady 22-24%, which says that a p.d.f., such as Fig. 7 for the probability p_{int} of finding an interval of width d should obey a scaling rule

$$p_{\text{int}}(d) = f(d / L_{n+1}) \qquad (5)$$

It is possible that f may be a universal function (or independent of n for large n).

With $N = 1$, $c_{max} \cong 2.7$ based on computational experiments. If N is increased it is found that c_{max} also gets larger, and to date the largest value found is about 5 (corresponding to a fractal D of 0.2). Since the recursive ordered fractal of Fig. 5(c) can have any fractal D down to zero, there may be no solid upper limit for c for single segments.

(d) The Placement Probability.

Since we have a complete list of the gaps we can find the total amount of "free" length in all the gaps. If we divide this by the total length L of the container segment we find a placement probability p_{place}. This probability falls with increasing n (corresponding to more trials) and varies somewhat depending on the random numbers.

$$p_{place} = \frac{1}{L} \sum_{L_{gap} > L_{n+1}} (L_{gap} - L_{n+1}) \qquad (6)$$

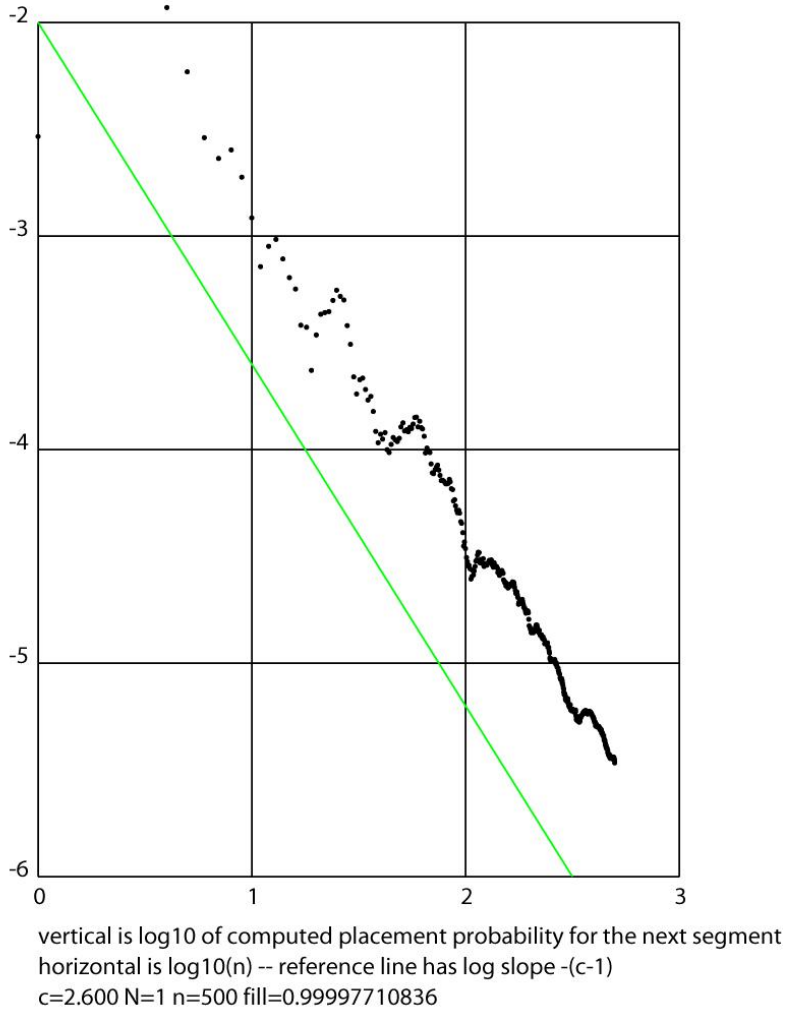

Fig. 10. The placement probability. The black points show $\log_{10}(p_{place})$ versus $\log_{10}(n)$. The green line has a reference slope of $-(c-1)$.

We would expect that the average number of trials for placement n would be proportional to $1/p_{place}$. By the reasoning given earlier, this plot of p_{place} should have a slope $-(c-1)$, the negative of that for the n_{tri} data. It is seen that this is obeyed reasonably well, and is far less noisy than the n_{tri} data. Because the process is

affected by the particular random numbers used, there are dips and excursions in the data, but the general trend is a straight line. The dips etc. show that this is a random process having memory. The very low values of p_{place} for large n agree with the huge numbers of trials needed for large n.

Since there are several gaps available to be filled at each stage, there is a sort of feedback mechanism involved. If by random chance the largest gap is filled, the algorithm will have a harder time and take more iterations at the next placement. On the other hand if the smallest gap is filled the algorithm will have an easier time at the next placement. Study of the run records shows behavior of this kind -- a placement which takes an unusually large number of trials is followed by ones which take fewer. This may account for the dips and excursions of the data in Fig. 10.

11. The Dimensionless Average Gasket Width and the Halting Problem.

For the one-dimensional case (assuming inclusive boundaries) the dimensionless average gasket width $b(c,N,n)$ is defined by

$$b(c,N,n)=\frac{L-\sum_{i=1}^{n}L_i}{(n+1)L_{n+1}}=\frac{L-\sum_{i=1}^{n}\dfrac{L}{\zeta(c,N)(i+N-1)^c}}{(n+1)\dfrac{L}{\zeta(c,N)(n+N)^c}} \qquad (7a)$$

$$b(c,N,n)=\frac{\zeta(c,N)-\sum_{i=0}^{n}\dfrac{1}{(i+N-1)^c}}{\dfrac{(n+1)}{(n+N)^c}} \qquad (7b)$$

$$b(c,N,n)=\frac{\sum_{i=1}^{\infty}\dfrac{L}{\zeta(c,N)(i+N-1)^c}-\sum_{i=1}^{n}\dfrac{L}{\zeta(c,N)(i+N-1)^c}}{(n+1)\dfrac{L}{\zeta(c,N)(n+N)^c}}=\frac{\sum_{i=n+1}^{\infty}\dfrac{1}{(i+N-1)^c}}{\dfrac{(n+1)}{(n+N)^c}} \qquad (7c)$$

$$\begin{aligned} b(c,N,n)&=\frac{(n+N)^c}{(n+1)}\sum_{i=n+1}^{\infty}\frac{1}{(i+N-1)^c}=\frac{(n+N)^c}{(n+1)}\sum_{j=0}^{\infty}\frac{1}{(j+N+n)^c} \\ &=\frac{(n+N)^c}{(n+1)}\zeta(c,n+N) \end{aligned} \qquad (7d)$$

after n placements. Here L_i is defined by Eq. (1). It is thus seen that the dimensionless average gasket width for this one-dimensional case is the ratio of the mean interval value (gasket width/(n+1)) to the next-to-be placed segment length L_{n+1}.

For periodic boundaries the result of Eq. (7d) would be

$$b(c,N,n) = \frac{(n+N+1)^c}{n}\,\zeta(c, N+n+1) \qquad (7e)$$

When $b(c,N,n) > 1$ the algorithm cannot halt[2]; there is always at least one gap $> L_{n+1}$ which a sufficiently large number of trials will find. If $b(c,N,n) < 1$ halting is possible, but may not happen in an individual case, depending on the random numbers which are used. When $b(c,N,n) = 1$ halting occurs[3] with zero probability.

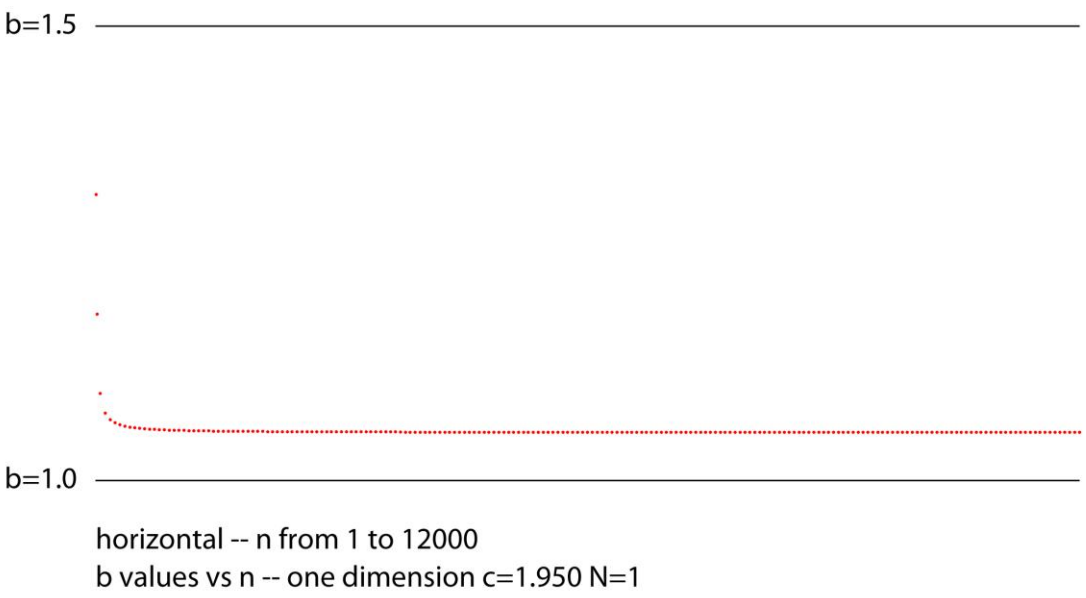

Fig. 11. Computed values of $b(c,N,n)$ for $c = 1.95$ and $N = 1$. The computed b value for $n = 12000$ is 1.05248. If $b > 1$ for all n, the process does not halt. Tests indicate at least 6 significant decimals accuracy.

It will be seen that $b(c,N,n)$ is independent of the random numbers used by the algorithm, and it can thus be computed by the usual methods of calculus. When $b(c,N,n)$ is calculated numerically the results of Fig. 11 were found for the specific case $c = 1.95$, $N = 1$. It can be seen that $b(c,N,n) > 1$ for this range of n values. This also holds for smaller c values. The nearly-flat behavior of this function is similar to what was seen for b in the 2D case. When the behavior of the algorithm for large c and $N = 1$ was studied it was found that one could make successful runs up to about $c = 2.7$, at which point one began to observe halting in some runs.

An asymptotic form or limit for $b(c,N,n)$ when $n \to \infty$ is a serious challenge. A lower bound for $\zeta(c,N)$ can be found by comparison with the corresponding integral valid for $c > 1$, namely

[2] The reasoning here is based upon the idea that if we have values $x_1, x_2, \ldots x_n$ with mean value m, it must necessarily be true that at least one of the values x_j is $\geq m$.

[3] The details here will depend on whether the placed segments are defined as open sets, closed sets, or semi-closed sets.

$$\zeta(c,N) > \int_N^\infty x^{-c}dx = \frac{1}{(c-1)N^{c-1}} \qquad (8)$$

thus $$b(c,N,n) = \frac{(n+N)^c}{(n+1)}\zeta(c,n+N) > \frac{(n+N)^c}{(n+1)}\frac{1}{(c-1)(n+N)^{c-1}} \qquad (9)$$

whence $$b(c,N,n) > \frac{(n+N)}{(n+1)}\frac{1}{(c-1)} \qquad (10)$$

$$\lim_{n\to\infty}\left\{\frac{(n+N)}{(n+1)}\frac{1}{(c-1)}\right\} = \frac{1}{(c-1)} \qquad (11)$$

It can be seen from Eq. (11) that when $c < 2$, $b(c,N,\infty) > 1$. Thus for $1 < c < 2$ the algorithm does not halt. A better lower bound can probably be found. This result is in general agreement with the detailed numerical calculations of Fig. 11. *If the calculations are accurate, and the reasoning sound, it has been shown that the algorithm does not halt for a substantial range of c values:* $1 < c < 2$.

12. Discussion and Conclusions.

Fractals are of great interest today in a number of scientific disciplines [1-3].

The data presented here give a computational account of the process. Many of the parameters and statistical distributions defined here will likely be involved in further studies of the subject.

Evidence has been presented in favor of the claims made above. The construction rule (Eq. (1)) ensures that the process is space-filling. The slope *c* seen for the cumulative trials data in Figs. 2 and 8 supports the idea that the process never halts. The flawless setup for the bisegment case supports the idea that any shape works. The idea of order and disorder and its association with *c*, along with the existence of perfect-order recursive fractals with the right *D* values provide a plausible explanation for why there is a maximum *c* value.

Are these claims true? Can formal proofs be found? Section 11, Eq. (11) shows unconditionally that for the special case $1< c < 2$ the algorithm never halts.

The algorithm described does not fit comfortably into any of the established areas of mathematics. Shape is important and that is the traditional province of geometry. The power law and zeta function belong to calculus and infinite processes. The use of random numbers necessarily involves statistics. The best label for this algorithm is perhaps "stochastic process".

Figure 1 shows how line segments obeying a power law can be randomly placed within a container interval, completely filling it in the limit $n \to \infty$. One can also think of the same sequence of segments being placed non-randomly, placing the biggest one (say) at the left edge of the container interval and working rightward, placing each new segment as a semiclosed interval whose left boundary is the right boundary of the previous segment. It is quite curious and interesting that both procedures fill the container segment. Nonrandom

filling will work for any sequence of intervals which sum to L (such as $L/2$, $L/4$, …). Random filling apparently only works for a power law.

The relationship between c and "degree of order" is one of the more interesting features.

Thus far the largest c value which has been found to work for single segments is a bit more than 5. There is no obvious reason why even higher values cannot be found. Recursive 1D fractals can be constructed with D values approaching 0, which would correspond to c becoming arbitrarily large. The c_{max} value for simple segments remains an open question.

Because we have constructed additive fractals rather than subtractive, we have followed the rules defined by physicists for determining the fractal dimension D in random fractal packing studies [4-6]. Our work differs from these packing studies in that our fractals are non-Apollonian. Additional information can be found at the author's web site [8], and earlier less-complete publications [9-10].

It is noted in the discussion of Figs. 2 and 8 that the log slopes ϕ of these regression curves are approximately the same as c. Might a proof of such a claim take the form of a demonstration (using statistical methods) that the mean or most-probable value of ϕ is c? Might the important properties be best expressed as probabilities rather than conventional mathematical certainties, as in geometry?

If proofs are to be found, the 1D case is probably the best place to start.

13. References.

[1] Benoit Mandelbrot, *Fractals; Form, Chance, and Dimension* (W. H. Freeman and Company, San Francisco, 1977).

[2] Mark Buchanan, *Ubiquity* (Three Rivers Press, New York, 2000); *Nexus* (W. W. Norton & Co., New York-London, 2002).

[3] Philip Ball, *Critical Mass* (Farrar, Straus, and Giroux, New York, 2004).

[4] P. S. Dodds and J. S. Weitz, Phys. Rev. E, **65**, 056108-1 (2002).

[5] P. S. Dodds and J. S. Weitz, Phys. Rev. E, **67**, 016117-1 (2003).

[6] G. W. Delaney, S. Hutzler, and T. Aste, Phys. Rev. Letters, **101**, 120602 (2008).

[7] Paul Bourke's web site is: http://paulbourke.net/randomtile

[8] The author's web site is http://john-art.com.

[9] John Shier, Proceedings of ISAMA 11, June 13-17, 2011, in Hyperseeing, Summer, pp.131-140 (2011).

[10] John Shier, "Statistical Geometry in One Dimension", Hyperseeing, Summer (2012).