COASTLINES AND FRACTAL GEOMETRY: 
ESTIMATING LENGTH AND GENERATING ISLANDS
Introduction

The first connections that were made between coastlines and fractals were found in posthumous papers written by Louis Fry Richardson. In his pioneering work, Richardson pointed out that the question “how long is the coastline of Great Britain?” has no explicit answer and can only be estimated.\(^1\) Fractals and their properties were first brought to light and described in 1977 by the father of fractal geometry, Benoit Mandelbrot, in his book entitled “Fractals: Form, Chance, and Dimension.”\(^2\) The most general definition given by Mandelbrot of a fractal is a shape made of parts similar to the whole in some way.\(^3\) Rather than being described by an algebraic function, fractals are best described as an iterative process.\(^4\) Fractals are made of an infinite number of iterations or repetitions of the similar component, making each part look like the whole. One interesting fractal property is that as the number of iterations increase, the length of the fractal increases, which results in an **infinite length** as the number of iterations approaches infinity.

Richardson’s theory states that the length of a coastline like Great Britain’s will approach infinity if one continues to measure in smaller and smaller size increments.\(^5\) Although coastlines aren’t geometrically self-similar, they are statistically self-similar. This means that each portion is a reduced image of the whole where some variation from the whole takes place.\(^6\) As the unit of measurement becomes smaller, the coastline contains more and more detail. Every inlet has another small inlet within it and every peninsula has smaller peninsulas branching out from it.\(^7\) These characteristics imply that coastlines are natural fractals, which in turn makes the measuring of coastlines difficult. In a practical sense, the length of a coastline is not infinite, but one cannot deny that using a smaller measuring stick results in a larger length over the same coastline. This property causes inconsistencies in the recording of coastlines all over the world, because nations use differing units of measurement.\(^8\) Because coastlines are natural fractals, we can therefore represent and approximate their lengths using fractal methods.

Standardizing the measuring of coastlines would eliminate inconsistencies and ambiguity in recorded coast lengths. In this paper we investigate the existence of a size increment that would optimize the measuring of the coastline’s length. We assume that coastlines are not a complete geometric fractal and therefore do not have an infinite length. It follows that there is an

---

\(^2\) Ibid., 1.
\(^6\) Bassingthwaighte, *Fractal Physiology*, 33.
\(^7\) Ibid., 11.
optimal unit of measurement at which to measure length. We theorize that eventually the act of decreasing the increment size will no longer make a significant difference in the length and therefore act as a good standard of measurement.

Methods

To begin measuring the length of a coastline, we first needed to generate a coastline that displayed a fractal nature. The random midpoint displacement method is one way to generate a fractal random walk that approximates Brownian motion and, when controlled properly, results in a nice looking island. This method involves setting the first point of a line equal to zero and the end point equal to a Gaussian random number. Then we find the midpoint of these and displace it by a predetermined factor. This process results in two lines, and the midpoints of these are calculated and displaced again. The process is then repeated a specified number of times, or iterations.\(^9\) In a complete geometric fractal, this would continue for an infinite number of iterations.

The displacement factor previously mentioned comprises a Gaussian random number multiplied by a scaling factor.\(^10\) The scaling factor can be chosen arbitrarily, but is best if the factor exponentially decreases as the number of iterations \(i\) increases. We used \(2^{-H_i}\) as the scaling factor, where \(H\) is a constant Hurst component, and the \textit{randn} function from MatLab to generate a pseudorandom number for each displacement. The Hurst component controls the smoothness by encouraging the curve to “follow itself” along its path.

Our recursive code uses the midpoint displacement method to create vectors containing the displaced points on the interval \([0,1]\). We plotted the generated vector against a \textit{linspace} of the same length, which resulted in a controlled version of a random Brownian walk. We created an island by plotting two of these vectors against each other. In our code, we set the first and last components of the vectors equal to one rather than zero and a Gaussian random number, causing the ends of the vectors to meet when plotted against each other.

Once we had fractal islands, we could proceed with the measurement of the length of their coastlines. We went back to the basics and used the Pythagorean Theorem to find the lengths between consecutive points and summed them up to measure the total length. The collection of points from our code created a curve that was piecewise and non-differentiable, and therefore Pythagorean Theorem seemed the best and easiest method to measure the length. In addition, the simple calculations resulted in fairly quick computations no matter how many iterations we performed.

The next step was to confirm that the fractal island simulations we had created actually simulated fractals and coastlines. We did this by mass producing island lengths and comparing

---


\(^10\) Ibid., 487.
them with fractal characteristics and visually determining if they were controlled enough to be considered islands. We created and measured 1100 islands: 11 numbers of iterations of 100 islands each, holding the Hurst component constant. We examined this data to confirm that the length increased as the iterations increased. We created and plotted various sets of 9 islands and visually confirmed that each island combination of Hurst component and iteration number displayed self-similarity. We also used these visual plotted sets to determine which Hurst component created the most realistic islands. Additionally, we created and measured 630 islands comprising 7 different numbers of iterations and 3 Hurst components to analyze which independent variable (including the random number) had the greatest effect on the length of the islands.

Finally, we began our exploration into the existence of an optimal unit of measurement for natural coastlines. The accuracy of a simulated coast length increases as the number of iterations increases, but the detail increases to an even greater degree. We cannot measure the exact length because with each increase in the number of iterations we find a level of detail that wasn’t accounted for with the previous iteration. The inability to calculate the exact length of the coastline is an example of the repetitive nature of fractal islands.

Therefore, in order for an optimal unit of measurement to exist, the following condition must occur. If the standard error of the measurements increases with each iteration increase, then a point will eventually be reached at which the standard error for a specific iteration is greater than the difference in mean length to the next level of iteration. At this point the standard error is large enough that an increase in the number of iterations will no longer return a more accurate estimate of the length. Although the estimated length will continue to increase, suggesting a higher level of accuracy, the range of the standard error will become so large that the new estimate will supply no additional meaningful information. Using the larger set of data, we plotted the mean and the standard error separately against the iteration number to examine the slopes of each graph and determine whether an optimal unit of measurement exists.

**Empirical Model**

In order to explore the behavior of coastline lengths we evaluated the following multiple regression model using ordinary least squares:

\[
L = \beta_1 + \beta_2 i + \beta_3 H + \varepsilon
\]

Where \( L \) serves as the dependent variable and is the length of the coast measured by the function coast_length.m. The explanatory variables are \( i \) and \( H \) where \( i \) is the number of iterations ranging from 9 to 20 and \( H \) is the Hurst component ranging from 0.7 to 0.9. The error term, \( \varepsilon \), is the unexplained variation caused by MatLab’s randn random number generator.
Figure 1: Island with 9 iterations and Hurst component 0.7. The least controlled of our samples; results in jagged edges, unrealistic shapes, and overlapping lines and “lakes”.

Figure 2: Island with 9 iterations and Hurst component 0.8. Somewhat controlled, but still contains unwanted characteristics such as excessive overlapping lines.

Figure 3: Island with 9 iterations and Hurst component 0.9 results in controlled edges and realistic coastlines. Does not contain a lot of detail, but still simulates a coastline.
Figure 4: Island with 12 iterations and Hurst component 0.7 results in much more detail than the last iteration, though the edges are rugged and contain lots of crossover lines in the black clusters.

Figure 5: Island with 12 iterations and Hurst component 0.8. Displays smooth perimeter lines and less unrealistic jagged borders with the same amount of detail as the previous figure. Still contains some unwanted lakes.

Figure 6: Island with 12 iterations and Hurst component 0.9 shows almost no black clusters of crossover; includes detail and controlled coastlines.
Figure 7: Island with 15 iterations and Hurst component 0.7 displays too much uncontrolled data. The detail is jagged and contains many lakes and excessive crossover.

Figure 8: Island with 15 iterations and Hurst component 0.8 results in less black clusters of crossover and lakes. It has the likeness of a realistic island.

Figure 9: Island with 15 iterations and Hurst component 0.9 contains controlled detail and a smoother edge. Lakes and overlaps are minimized, the coastline is clearly defined.
Figure 10: Island with 15 iterations and Hurst component 0.9. The magnified peninsula comes from the selected portion of the island on the “northern” side. The detail shown in the magnified portion matches that of the rest of the entire island, displaying the fractal characteristic of self-similarity.
Results and Discussion

Figures 1 through 9 (shown above) are examples of islands at iterations of 9, 12, and 15, each with Hurst components of 0.7, 0.8, and 0.9 using the MatLab function *islands.m*. Visually, we can see that the Hurst component seems to have a greater effect on the physical structure of the island when compared to the effect of the number of iterations. As the Hurst component increases, the coastlines become less jagged and more realistic. We can visually conclude that the Hurst component 0.9 resulted in the most realistic islands, regardless of the number of iterations. Though the larger iterations give more detail and information about the coastline and therefore results in larger lengths, the Hurst component controls the randomness of the displacement appropriately. With close examination of various sets like these nine examples, we confirmed that all displayed self-similarity. An example of this is shown in Figure 10 which shows an image of a peninsula magnified to show the detail next to the island as a whole for comparison. These two sections, the part and the whole, look as though they could have come from the same scale, thus confirming the self-similarity of our fractal island simulations.

The estimates recorded in Table 1 confirm the relationship between coast length, the number of iterations and the Hurst component.

**Table 1.** Coastline Regression Estimates. Dependent variable: L = Coast Length

<table>
<thead>
<tr>
<th>Variable</th>
<th>Coefficient</th>
<th>Std. Err.</th>
<th>P &gt;</th>
<th>t</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Constant</td>
<td>67.28499</td>
<td>2.254045</td>
<td>0.000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Iterations</td>
<td>2.759955***</td>
<td>0.0977252</td>
<td>0.000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hurst</td>
<td>-104.9853***</td>
<td>2.393769</td>
<td>0.000</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

F-test: 0.000

R²: 0.8127

The coefficients are reported with their associated p-value for the null hypothesis that the estimated value is equal to zero. Std. Err = standard error.

*** Significant at the 1% level

The p-value indicates that both the number of iterations and the Hurst component are significantly different from zero at less than the 1% level. This means that across samples and more than 99% of the time the number of iterations and the Hurst component are relevant to coast length. The coefficient estimates for the number of iterations and the Hurst component reveal that the Hurst component has a much stronger impact on coast length than the number of iterations.

The F-test reveals that both iterations and the Hurst component are jointly significant at less than the 1% level. This means that across samples and more than 99% of the time our
chosen variables are jointly relevant to coast length. The $R^2$ indicates that roughly 81% of the variation in coast length is explained by the number of iterations and the Hurst component. The remaining 19% of unexplained variation is due to the error term, $\varepsilon$, which captures the influence of the random number generator.

Even though the `midptfrac.m` and the `islands.m` functions used to create our islands were controlled with a scaling factor, the resulting Brownian walk is still unpredictable, and when taken together these vectors can create some unrealistic coastlines that contain overlapping lines or loops. For our purposes, we will call these crossover sections “lakes” within our island. The code `coast_length.m` does not differentiate between the perimeter of the island and the sections within. Consequently, some of the reported coastline lengths are overstated because both the perimeter of the island and the perimeter of the lakes on the interior are included in the measurement. The Hurst component controls the jaggedness of the coastline, so for the purposes of our examination we were able to minimize the amount of overlap by setting $H = 0.9$. The standard errors reported in Figure 11 were calculated with $H = 0.9$.

The change in standard error over an increase in the number of iterations must be positive for an optimal unit of measurement to exist, because the standard error must increase as the number of iterations increase. Our results suggest that this condition does not occur within the generated data. Figure 11 demonstrates that the standard error for each measurement decreases exponentially as the number of iterations increases:

![Figure 11: Standard Error vs Iterations](image)
The decrease in the standard error follows from the method of midpoint displacement used to create the coastlines. With each additional iteration, the distance the midpoint is displaced decreases by the scaling factor $2^{-H}$. As the midpoint is displaced by a smaller and smaller amount, the coastline converges to the theoretical natural fractal coastline. Imagine that a person is walking the perimeter of a natural fractal island and gathering geographical data points with a GPS. The resulting points are analogous to the points that create our simulated islands with $islands.m$. As we increase the number of iterations within our function, we are theoretically gathering more GPS coordinates in between the ones previously gathered along the natural fractal coastline. With the increasing amount of data, we continue to get closer to the length of the theoretical coastline with a decreasing standard error. Because the standard error decreases with additional iterations, we failed to support the existence of an optimum unit of measurement.

Conclusion

After examining our plots and statistics of the lengths, we were able to conclude that our MatLab code generated successful simulations of fractal islands, because they displayed characteristics that describe fractal nature. With our arbitrary scaling factor, they were accurate enough to successfully analyze properties of fractal island structures. Through statistical examination we confirmed the expected relationships between coast length, the number of iterations, and the Hurst component. These relationships demonstrate that the Hurst component is the single most important indicator of coast length and add perspective to the influence of the random number generator on the model. The number of iterations and the Hurst component are both statistically significant to the length of the coastline at the 99% level of confidence. This validates our ability to control the outcome of our $islands.m$ function with different or varying scaling factor inputs. With more exploration into the scaling factors which control the random nature within the displacement, one could generate fractal islands that contain no crossover lines or “lakes” and compute their length, resulting in an improved realistic quality to the island simulations.

Accurate and consistent measurement of both natural and political boundaries is a vital practical matter that can only be achieved through a standard unit of measurement. Although our findings do not support the existence of a statistically optimal unit of measurement, an arbitrarily determined standard unit of measurement could still be beneficial. The fractal characteristics of natural coastlines suggests that in order for a unit of measurement to be declared “good enough” a certain amount of error must be considered acceptable. We recommend that the policy makers of national cartographic societies collaborate with the International Organization for Standardization (ISO) in selecting a unit of measurement that reflects an acceptably low error tolerance to represent the standard unit of measurement when measuring coastlines or international boundary lines.
References


Appendix:

function B = midptfrac(A,H,i)

% Inputs:
% % A is an empty vector (consists of zeros) to be filled with the displaced points.
% Its first and last components are both set to one and its length k is equal to \(2^n + 1\) where \(n\) is the number of iterations (because the length must be divisible by \(2^n\) times for indexing purposes).
% % H is the constant Hurst component that controls the displacement and causes the resulting line to look smoother.
% % i is the iteration counter which increases by one every time the function calls itself.
% % Outputs:
% % B is a vector containing the displaced points on the interval \([0,1]\).

i = i+1; % start iteration count
scale = 2^(-H*i); % create scaling factor
N = size(A,1); % identify first and last indices
B = A; % copy the empty matrix
d = (N+1)/2; % find the midpoint index
B(d) = mean([A(1),A(end)]) + scale*randn; % add the average of the components and the displacement

if N > 3 % if N > 3, the correct number of iterations are not complete, continue with next level
    B(1:d) = midptfrac(B(1:d),H,i); % first half of vector
    B(d:N) = midptfrac(B(d:N),H,i); % second half of vector
end % if N <= 3, the iterations are complete (the next index d would not be an integer)
end
function D = mydist(x,y);

% Inputs:
% x and y are two vectors of the same length
% Outputs:
% D is the length of the line created when x and y are plotted against each other

n = numel(x);
for i = 1:n-1
    d(i) = sqrt((x(i+1)-x(i))^2 + (y(i+1)-y(i))^2); % Pythagorean Theorem
end
D = sum(d); % sum all lengths
end

% Plots a graph of fractal components and island

function length = islands(N,H)
    n = N; % iterations
    h = H; % smoothing

    k = 2^n + 1; % length of vector must be divisible by 2 n times
    i = 0; % iteration counter
    A = zeros(k,1); % empty vector
    A(1) = 1;
    A(end) = 1;

    % call midptfrac function for two vectors
    Bx = midptfrac(A,h,i);
    By = midptfrac(A,h,i);

    % plot against linspace
    u = linspace(0,1,numel(Bx));
    figure(1)
    plot(u,Bx,'r-',u,By,'b-')
    % plot against each other, makes island
    figure(2)
    plot(Bx,By,'k-')
end
function L = coast_length(N,H,C)
 n = N; \%iterations
 h = H; \%smoothing
 c = C; \% how many islands we create and measure

 k = 2^n + 1; \% length of vector must be divisible by 2 n times
 i = 0; \% iteration counter
 A = zeros(k,1); \% empty vector
 \% set initial values
 A(1) = 1;
 A(end) = 1;
 L = zeros(c,1); \% empty vector to be filled with lengths of c islands

 for c = 1:C

 \% call midptfrac function for two vectors c times
 Bx = midptfrac(A,h,i);
 By = midptfrac(A,h,i);

 L(c) = mydist(Bx,By);
 end