CONTINUOUS AND DISCRETE MODELS FOR SIMULATING STONE EROSION

JONATHAN ALLEN, JOHN WANG

Abstract. In this paper we explore various techniques for modeling erosion processes on a stone. We develop and analyze an algorithm for modeling chipping by representing a stone as a polygon. We also develop a method for modeling gradual stone erosion through a Fast Fourier Transform (FFT) representation of a stone.

1. Introduction

Stones in riverbeds and on the ocean floor have very distinct shapes and contours. These stones tend to be smooth and flat, but the reason for this is not obvious. This paper examines how erosion affects the shape of stones. In particular, we examine two different mechanisms for stone erosion, a discrete one and a continuous one, and produce models for each one.

The first mechanism we study is the chipping of stones. One can imagine stones in a riverbed being tossed around and colliding with other stones. We create a discrete, polygonal model of a stone and a means of chipping the stone through a shear force. This model predicts that stones will tend to be flat and elongated. Our simple model of the chipping process can result in relatively good estimations of stone shape.

The second mechanism we study is the wearing down of stones by gradual erosion. This mechanism appears when stones are gradually worn away by sand, water, or some other agent. We start of with a simple smooth model of the stone using spectral methods with the Fast Fourier Transform. This leads to problems, since the size of the stone decreases over time and changes shape, so we need to resample. Most of our work involves designing a method for this resampling. We are then able to demonstrate our continuous model on a few simple problems, namely the processes of smoothing a stone and the process of erroding a stone from the bottom.

Date: November 7, 2013.

2. Discrete Model

In this section, we shall describe a model for simulating the chipping of a stone. In particular, we think of the erosion of a stone as coming from shear forces. This discrete model for eroding the stone allows us to simulate random processes and view the shape of a stone after some number of time steps. We provide computer simulations for the erosion process and analyze the model through these simulations.

2.1. **Shearing Stones.**

The discrete model for stone erosion is based on the idea that stones are eroded via chipping. The process of chipping is analogous to dropping a stone from a specified height and applying a shear force to a particular point on the stone. Intuitively, one can think of a stone in a stream being eroded by being tossed and turned by the force of the river. This tossing and turning causes the stone to collide with other stones on the bottom of the riverbed, causing chipping to occur.

We model the interaction between two stones as a shear force, a force that causes two parts of a stone to move in opposite directions. The point where a stone collides with another stone is the point where the shear force is concentrated. This causes the stone to break into two pieces. The processes by which shear forces cause breakages in materials are extremely complicated, so we will use the following simple assumption: we shall assume that the amount of volume that is sheared off of a stone is proportional to the amount of force incident on the stone.

2.2. **Definitions.**

This section will define how we represent a stone in the discrete model. We will represent a stone as a polygon in k-dimensional space with n vertices.

Definition 2.1. *A polygon s consisting of n vertices is defined by the set of vertices $s = \{\mathbf{v}_0, \ldots, \mathbf{v}_{n-1}\}$ where each vertex $\mathbf{v}_i \in \mathrm{R}^k$ is a tuple. Vertices $\mathbf{v}_i$ and $\mathbf{v}_{i+1}$ are connected by a line segment for all $i \in \{0, 1, \ldots, n\}$ and the line segments do not intersect each other, except at the vertices.*

When referring to vertices, we will use the convention of taking vertex indices modulo n for convenience. Hence $v_{i \mod n} \equiv v_i$. Note that this means that $v_n = v_0$ so that vertex v_{n-1} has a line segment connecting it to v_0.

The centroid $\mathbf{c}_s$ of a stone s is the center of mass of the stone when the stone has uniform density. In other words, the centroid is the point $\mathbf{c}_s \in \mathrm{R}^k$ where the following holds:

$$\int_V (\mathbf{r} - \mathbf{c}_s) dV = 0 \tag{1}$$

Where V is the area of the stone, using all values $\mathbf{r} \in V$.

Definition 2.2. *A pseudo-probability distribution P.*

We will also introduce the notion of a convex vertex, which will be useful when defining our chipping process.

Definition 2.3. *A convex vertex $\mathbf{v}_c$ on a polygon s consisting of n vertices is a vertex $\mathbf{v}_i$ which, when removed from the polygon s to create a new polygon $s' = \{\mathbf{v}_0, \ldots, bvecv_{i-1}, \mathbf{v}_{i+1}, \ldots, \mathbf{v}_{n-1}\}$ has the property $V(s) \geq V(s')$ (where $V(\cdot)$ denotes the volume of a stone).*

In the following sections, we will assume that one cannot chip a vertex which is not convex. The reasoning relies on our intuition for how a stone chips. Non-convex vertices can be thought of as moving in towards the center of a stone, and chipping a stone can be thought of being caused by the stone crashing into the ground. It is impossible for a stone to be chipped on a vertex which is not convex if the stone is falling onto a flat surface (the protruding vertices surrounding the convex vertex will be chipped first). Therefore, we make the assumption that convex vertices are the only ones which can be chipped through our mechanism.

2.3. **The Chipping Process Model.** With this in mind, we shall now develop a model for the chipping of a stone based on shearing. We will confine our work to R^2. The main idea of the model is that we will represent a stone as a two-dimensional polygon, and chip off a constant amount of area of the stone at every time step. In this way, we will attempt to capture the process of a stone colliding with another stone on a riverbed. Randomness will be introduced into the model by randomly selecting somewhere on the stone to start the shearing.

We shall now give a formal definition of the chipping process. A chip for a probability distribution P, stone s, and area A is denoted $Chip(\mathrm{P}, A, s)$ and is represented as follows:

(1) Select a vertex $\mathbf{v}_j$ from s. To select this vertex, we find the centroid $\mathbf{c}_s$ of stone s and choose an angle $\gamma \in [0, 2\pi)$ uniformly at random. Now, define the ray $\mathbf{l}$ as the ray with an initial point of $\mathbf{c}_s$ which extends outwards infinitely at an angle of γ from the horizontal. The vertex with the minimum perpendicular distance to $\mathbf{l}$ will be $\mathbf{v}_j$. Repeat this process as many times as necessary until we have selected a convex vertex.

(2) Given vertex $\mathbf{v}_j$, define $\mathbf{l}_l$ as the line segment which connects $\mathbf{v}_j$ and $\mathbf{v}_{j-1}$. Similarly, define $\mathbf{l}_r$ as the line segment connecting $\mathbf{v}_j$ and $\mathbf{v}_{j+1}$ (recall that our convention is to take the vertex indices modulo n).
(3) Select $\mathbf{l}_1$ from the set $\{\mathbf{l}_l, \mathbf{l}_r\}$ uniformly at random (i.e. $Pr[\mathbf{l}_1 = \mathbf{l}_l] = Pr[\mathbf{l}_1 = \mathbf{l}_r]$). Call the line segment which was not selected $\mathbf{l}_2$.
(4) Select a point $\mathbf{p}_1$ at random from $\mathbf{l}_1$. To do this, we select a random $t \in [0, 1]$ such that $\mathbf{p}_1 = \mathbf{v}_j t + (1 - t)\mathbf{v}_{j-1}$. The distribution of t is defined by the probability distribution P so that $t \sim \mathrm{P}$.
(5) Select the point $\mathbf{p}_2$ which lies on $\mathbf{l}_2$ for which the polygon defined by $\{\mathbf{p}_1, \mathbf{v}_j, \mathbf{p}_2\}$ has an area of A. If no such polygon exists, then choose $\mathbf{p}_2 = \mathbf{v}_{j+1}$.
(6) Create a new stone s' whose vertices are given by

$$s' = \{\mathbf{v}_1, \ldots, \mathbf{v}_{j-1}, \mathbf{p}_1, \mathbf{p}_2, \mathbf{v}_{j+1}, \ldots, \mathbf{v}_n\}$$

Here, the vertex $\mathbf{v}_j$ has been replaced by $\mathbf{p}_1$ and $\mathbf{p}_2$ in the list of vertices for s. The new stone s' now has one more line segment and vertex than the old stone s.

A chipping process for a probability distribution P, stone s, area A, and iterations k is denoted $ChipProcess(\mathrm{P}, A, s, k)$. A chipping process returns a stone s' which has iteratively been through k chips. Thus, a chipping process takes a stone s and creates $s_1 = Chip(\mathrm{P}, A, s)$, $s_2 = Chip(\mathrm{P}, A, s_1)$, $\ldots$, $s_k = Chip(\mathrm{P}, A, s_{k-1})$, and returns $s' = s_k$.

2.4. **Simulations.** The chipping process model allows us to simulate changes to a stone. A stone which undergoes a chipping process should have a shape which is close to the stones which undergo erosion and chipping in the real world (if our chipping process model is an accurate description of erosion).

To examine the accuracy of our model, we performed computer simulations using the Python programming language and the Shapely library. With computer simulations, we were able to create multiple instances of a chipping process and analyze the shapes of the resulting stones. We were also able to use multiple probability distributions P and areas A to generate different chipping processes.

Figure 1 shows the results for nine chipping process runs. The parameters used in these runs were $\mathrm{P} = \mathrm{N}(0.5, 0.15)$, $A = 0.05$, $k = 50$, and $s = \{(0, 0), (1, 0), (1, 1), (0, 1)\}$. The chipping processes each started out with a stone which was a unit square. The stones were chipped 50

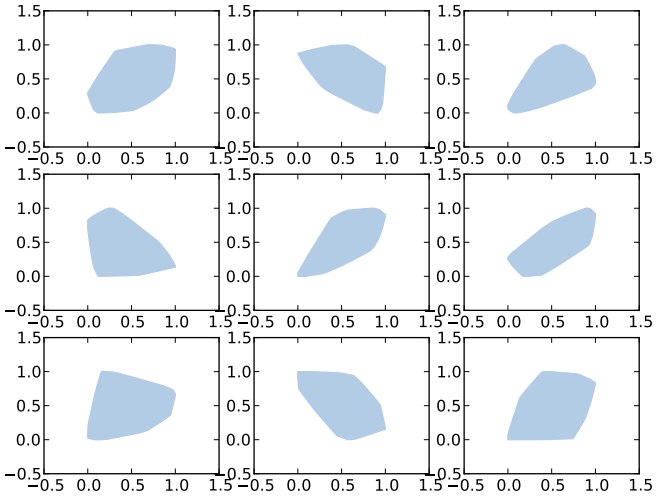

FIGURE 1. Chipping Process for a Unit Square

times by the method described by the model. The probability distribution P is a normal distribution so that $t = 0.5$ is the most likely t to choose. In other words, it is most likely that $\mathbf{p}_1$ will be chosen as the midpoint of the line segment $\mathbf{l}_1$.

This distribution was chosen because it is more likely that shear forces will cause a symmetric split about the vertex $\mathbf{v}_j$. However, it is still possible for non-symmetric splits to occur, i.e. when $t \neq 0.5$. The normal distribution captures this nicely (all realizations of t which are not in the interval $[0, 1]$ were discarded).

Figure 1 displays the typical results of the chipping process model. None of the stones resemble a square and it would be hard for anyone to have guessed that these shapes came from a common origin. In fact, the randomization from the distribution P makes all of the shapes slightly different.

To examine the chipping process more closely, we can look at how circular or elliptical the resulting shapes are. One would expect rocks in a stream or riverbed to be elliptical in shape. We can therefore examine the distance of each vertex from the centroid of the stone. In a circle, all of the distances would be the same. In an elipse, one would expect to have an even number of points which are close and far away.

Figure 2 shows the distribution of vertex distances which were simulated with the same parameters as used for figure 1. One can see that the distribution looks fairly normal, with a mean of a distance

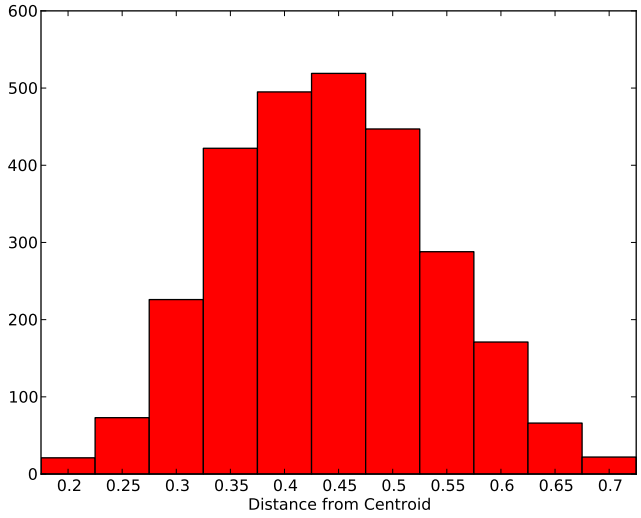

FIGURE 2. Histogram of Vertex Distance From Centroid

of 0.472. The distribution of vertex distances from the centroid show that the shapes that result from this set of chipping process conditions are neither elliptical nor spherical. However, a probability distribution $P = N(0.5, 0.15)$ tends to create a distribution of vertex distances from the centroid which are normal.

Choosing a different distribution for P yields dramatically different results. For example, if we choose $P = N(0.05, 0.15)$, we obtain the distribution shown in figure 3.

One can see that the distribution of vertex distances from the centroid are skewed so that there are far more vertices which are further away from the centroid. The resulting stones are much flatter than the stones shown in figure 1. The intuitive explanation for this is that there is a much higher probability that the shear will not be symmetric, so that the area that is removed will be much thinner. Because of this, it is more likely that longer, thinner stones will result.

3. Continuous Model

We next moved on to develop a continuous erosion model that operates on smooth surfaces in contrast to our previous discrete one that operated on piecewise linear surfaces. Physically this new model represents a surface being worn down gradually, such as a bar of soap in water or an ice cube melting.

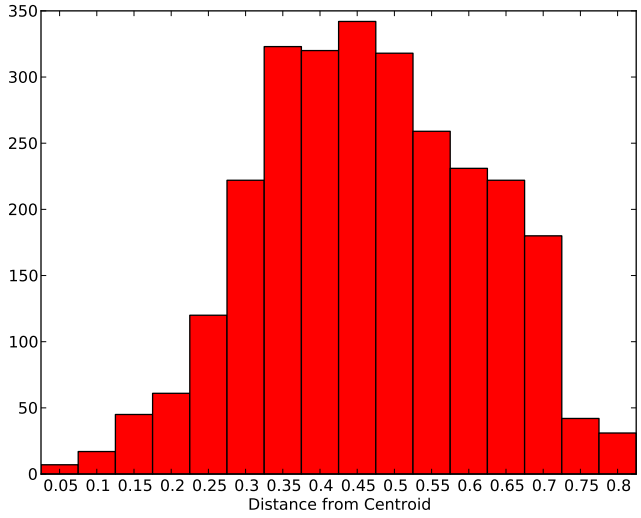

FIGURE 3. Histogram of Vertex Distance From Centroid for $\mathrm{P} = \mathrm{N}(0.05, 0.15)$

Continuous Erosion Processes. There are a wide range of different erosion processes, and most of them are far too complex for the scope of this paper. To generate simple simulations, we use key properties of the surface such as curvature and position as input variables to erosion functions.

We will start by introducing an explicit representation of a surface. This choice of using an explicit representation instead of an implicit one may seem odd to the reader, but we would like to stress that this was a conscious decision on our part because it paves the pay for the use of spectral methods to represent the surface, which we will discuss more in Section 3. To represent the surface explicitly, we will introduce a parameter $s \in [0, 2\pi)$ that increases when moving counterclockwise around the surface. Now the surface can be represented at any time t by $\vec{x}(t, s) = (x_1(t, s), x_2(t, s))$, for some functions $x_1(t, s), x_2(t, s)$. In contrast to modeling erosion as a discrete process, as was done in the previous section, we will now model it with the following differential equation

$$\hat{\mathbf{n}}(t, s) = (-\dot{x}_2(t, s), \dot{x}_1(t, s))$$

$$\frac{d(\vec{x}(t, s))}{dt} = g(t, s)\, \hat{\mathbf{n}}(t, s) \tag{2}$$

for some function $g(\vec{x}(s))$ that depends on the erosion process.

To simplify our equations, we introduce the following vector calculus notation

$$\begin{aligned}\dot{\vec{x}}(t,s) &\coloneqq (\dot{x}_1(t,s), \dot{x}_2(t,s)) \\ &\coloneqq \left(\frac{dx_1(t,s)}{ds}, \frac{dx_2(t,s)}{ds}\right) \\ \ddot{\vec{x}}(t,s) &\coloneqq (\ddot{x}_1(t,s), \ddot{x}_2(t,s)) \\ &\left(\frac{d^2x_1(t,s)}{ds^2}, \frac{d^2x_2(t,s)}{ds^2}\right)\end{aligned}$$

Example Surface. The initial shape that we will use for our simulations is shown in Figure 4.

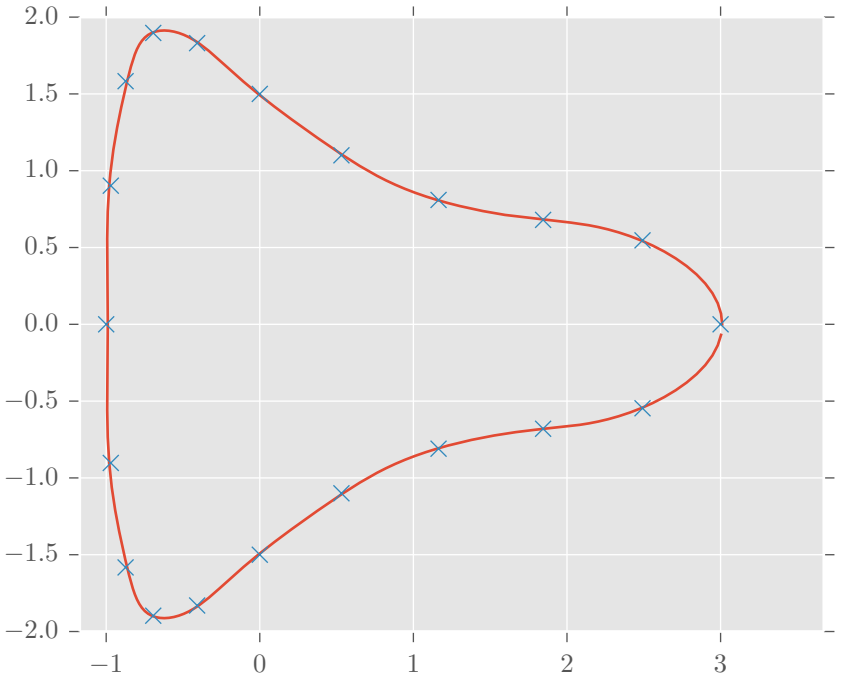

FIGURE 4. Example shape.

The equations for the shape are the following (T_i is the i$^{\text{th}}$ Chebyshev polynomial)

$$x_1(0,s) = \cos(s)\left(2T_0\left(\frac{s-\pi}{\pi}\right) + T_2\left(\frac{s-\pi}{\pi}\right)\right)$$
$$x_2(0,s) = \sin(s)\left(2T_0\left(\frac{s-\pi}{\pi}\right) + T_4(\left(\frac{s-\pi}{\pi}\right))\right)$$

The method used to represent the shape numerically will be explained in Section 3, but first we will take a moment to explain the various erosion processes that we will model in this section.

Smoothing Process. One of the processes that we will look at is a soap bar being worn down by a person holding it in their hand, which we will call the *Smoothing Process*. In this process, the corners of the surface will get worn down quickest, because they are the first parts of the surface to come into contact with the person's hand. Technically the furthest protruding bumps will be worn down quickest, but to simplify the problem, we consider every point of the surface to wear down at a rate proportional to the curvature at the point. We will use a signed curvature κ, so that a bump will have positive curvature, and an indentation will have negative curvature.

$$\kappa(t,s) = \frac{\dot{\vec{x}}(t,s) \times \ddot{\vec{x}}(t,s)}{\left\|\dot{\vec{x}}(t,s)\right\|^3}$$

$g(t,s)$ for this model is the following

$$g(t,s) = \tan^{-1}(\beta\,(\kappa(t,s) - \alpha)) + \frac{\pi}{2} \tag{3}$$

For our simulations we chose $\alpha = 1,\ \beta = 5$ so that $g(s) \leq 3$ ($\kappa(0,s) \in [-2,2]$ for all $s \in [0,2\pi]$). This restriction is not based on the physics of the problem, and was chosen purely for the aesthetic look of the resulting simulation.

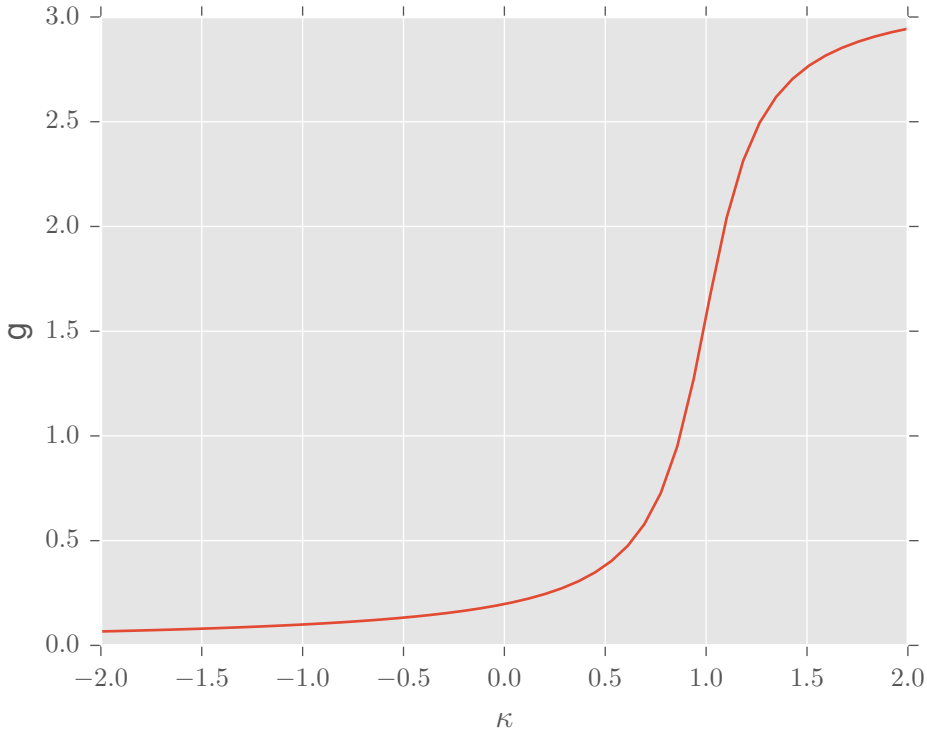

FIGURE 5. Plot of $g(\kappa)$ for *Smoothing Process*.

Bottom Recession Process. Our second process is an object sitting in a bath of acid, which we will call our *Bottom Recession Process*. In this process, the lowest points (smallest values of $\vec{x}_2$) on the surface will erode quickest. $g(t, s)$ for this model is the following

$$x_{2,min}(t) = \min_s(x_2(t, s))$$

$$f(s) = \frac{1}{\alpha\left(x_2(s) - x_{2,min}(t)\right) + \beta} - \gamma$$

$$g(s) = \begin{cases} f(s) & \text{for} \quad f(s) > 0 \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

For our simulations we chose $\alpha = 10,\ \beta = \gamma = 0.1$ so that $g(s) \approx 1$ for $x_2(s) \approx x_{2,min}$.

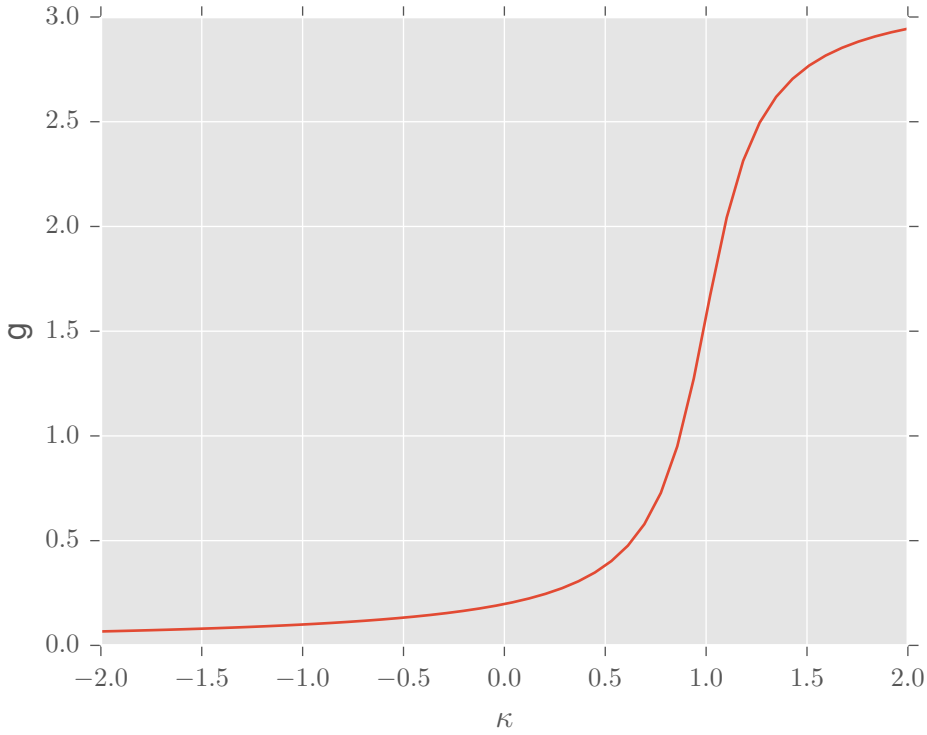

FIGURE 6. Plot of $g(y)$ for *Bottom Recession Process*.

Numerical Modeling. We chose to model the surface parametrically with a Discrete Fourier series and use the Fast Fourier Transform (FFT) for implementing Equation 2. This is because spectral methods will maintain the smoothness of the surface much better than a local approximation of the surface using finite differences which only acts locally at points. When we refer to smoothness, we are really referring to how many derivatives of the function are bounded, or in other words, how quickly the Fourier coefficients go to 0. Since, the surface is in $\mathbb{R}^2$, we will use the Real Fast Fourier Transform ($\mathcal{F}$), since it is slightly faster than the standard Fast Fourier Transform (equations for the $\mathcal{F}$ can be found in the appendix).

To model the surface parametrically, we define a parameter vector

$$\boldsymbol{s}_k = 2\pi \frac{n}{N}\ , \text{ for } n = 0, \ldots, N-1$$

then we generate a matrix of evaluation points

$$\boldsymbol{x} = \begin{bmatrix} x_1(\boldsymbol{s}_1) & x_2(\boldsymbol{s}_1) \\ \vdots & \vdots \\ x_1(\boldsymbol{s}_N) & x_2(\boldsymbol{s}_N) \end{bmatrix}$$

The FFT allows us to calculate derivatives easily with the following algorithm (a detailed derivation of the algorithm can be found in [2])

(1) $\hat{\mathbf{x}} = \mathcal{F}(\boldsymbol{x})$
(2) $\hat{\mathbf{w}}_k = ik\hat{\mathbf{x}}_k$, for $k = 0, \ldots, N-1$
(3) Because of a lack of symmetry in the Fouerier frequencies, if p is odd, then we set $\hat{\mathbf{w}}_{N/2} = 0$.
(4) Finally

$$\frac{d^p \boldsymbol{v}}{ds^p} = \mathcal{F}^{-1}(\hat{\mathbf{w}})$$

Forward time integration of Equation 2 can now be implemented with a standard ODE time stepping integration methods. In this paper, we used SciPy's `odeint` method for integrating Equation 2.

Point Collision Problems. Unfortunately, the simplicity gained from modeling the surface with a Fourier series is soon lost when we start time stepping. As can be seen in Figure 7, the process works well for a short time period, but then the surface forms loops (a physical impossibility).

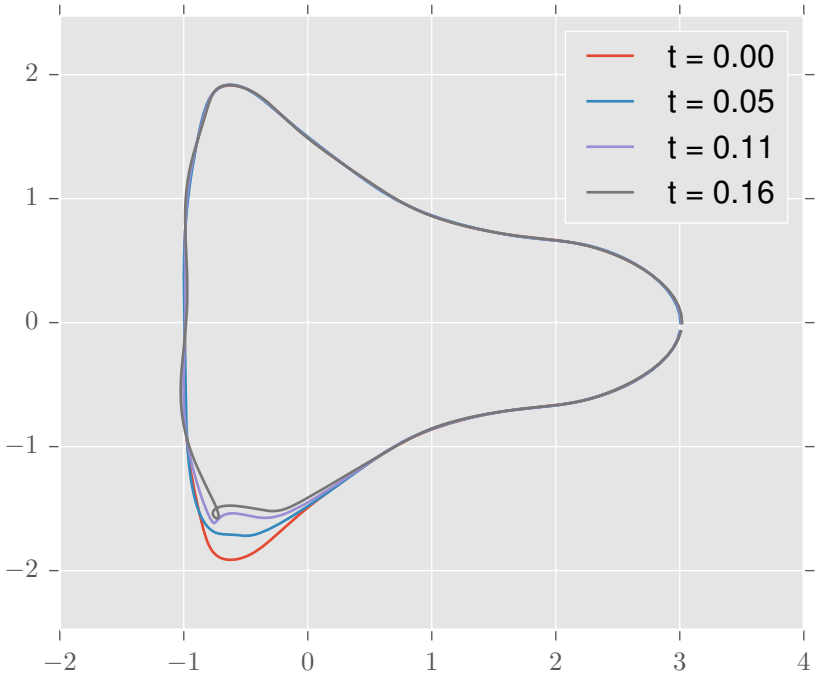

FIGURE 7

To investigate why this occurs, we will show a zoomed in portion of the same figure, with the evaluation points shown.

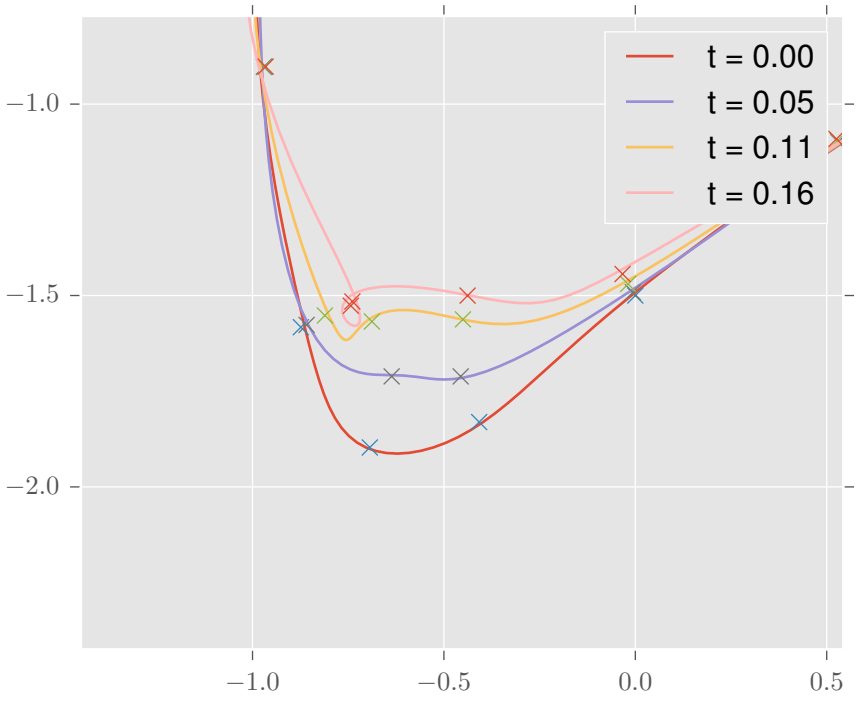

FIGURE 8

Looking closely, it can be seen that areas of high curvature form when evaluation points get close to each other, and this eventually leads to surface looping. Our guess is that this occurs because we are using discrete time stepping, so there are errors in the approximation of the shape's change with time. This is fine when the evaluation points are far from each other, but when they get close to each other, the errors become significant and affect the direction of the surface unit normal vectors. The folding causes unit normal vectors of nearby points to point at each other and which subsequently causes the points to cross over each other.

Redistributing Points. A seemingly obvious way to fix this problem would be to redistribute the points evenly on the surface at every step. This wouldn't be too difficult of a task, since we can easily integrate and interpolate with the FFT and find a new set of evaluation points, $\boldsymbol{x}$ that are evenly spaced on the surface.

Unfortunately, if we arbitrarily move points around, we lose spectral accuracy, because our new points will be sampled at different parameter

values $\tilde{s} \neq s$, but the Discrete Fourier transform assumes that the points are sampled at s.

The DFT is still assuming the function is sampled at

$$\boldsymbol{s}_n = 2\pi \frac{n}{N} \text{ , for } n = 0, \ldots, N-1$$

When we redistribute, we are sampling at frequencies $\tilde{\boldsymbol{s}} \neq \boldsymbol{s}$. In the continuous sense, this redistribution is defined by a parameter transformation $\tilde{s} = g(s)$, s.t.

$$\tilde{s}(s) = g(s), s.t. \frac{d\|\vec{x}\|}{d\tilde{s}}$$

Our shape function becomes

$$\tilde{x}(s) = (x \circ g)(s)$$

The Fourier series for this composite function may decay slowly, in which case the DFT will not accurately represent the surface.

This parameter transformation is okay if there exists some smooth (ideally analytic) function, $h(s)$, such that $h(s) = \tilde{s}$. This technique of parameter transformation is utilized extensively in spectral methods involving Chebyshev polynomials which use $h(s) = \cos(s)$ to cluster evaluation points near both endpoints of the domain.

Redistributing points in such a way to ensure that $h(s)$ exists is difficult though, so we will first describe some different methods we attempted at eliminating the surface folding.

Continuous Redistribution. Rather than , such that $(x \circ g)(s)$ is smooth as described above we decided to model our redistribution by adding a term to Equation 2, which is shown below. This works out nicely, because now the redistribution is smooth, so $h(s)$ must exist, as long as Equation 5 has a solution.

If we look at $\vec{x}(t, s)$ at some time $t = t_1$, then we can .

$$\begin{aligned} \frac{d\vec{x}_r(t_1, s, t')}{dt'} &= a_t(t_1, s, t') \frac{\dot{\vec{x}}_r(t_1, s, t')}{\left\|\dot{\vec{x}}_r(t_1, s, t')\right\|} \text{ , } \quad \vec{x}_r(t_1, s, 0) = \vec{x}(t_1, s) \\ &= \frac{\ddot{\vec{x}}_r(t_1, s, t') \cdot \dot{\vec{x}}_r(t_1, s, t')}{\left\|\ddot{\vec{x}}_r(t_1, s, t')\right\| \left\|\dot{\vec{x}}_r(t_1, s, t')\right\|} \frac{\dot{\vec{x}}_r(t_1, s, t')}{\left\|\dot{\vec{x}}_r(t_1, s, t')\right\|} \end{aligned} \tag{5}$$

Now, since $\vec{x}(t_1, s)$ is infinitely differentiable, $\vec{x}_r(t_1, s, t')$ is infinitely differentiable for all t'.

Now,

Physically $\dot{\vec{x}}_r(t_1, s, t')$ is the surface tangent vector, and $\ddot{\vec{x}}_r(t_1, s, t')$ is the rate of change of this surface tangent vector with respect to s. The reasoning behind 5 is that $\left\|\dot{\vec{x}}_r(t, s)\right\|$ is inversely proportional to the density of points on the surface at s; we want to shift points away from denser areas, which means moving them in the direction of larger $\left\|\dot{\vec{x}}_r(t_1, s, t')\right\|$, which is precisely the projection of $\ddot{\vec{x}}(t_1, s, t')$ onto the surface tangent vector.

To simplify the numerical computation, we combine 2 and 5 to get a single ODE

$$\frac{d(\vec{x}(t,s))}{dt} = g(t,s)\,\hat{\mathbf{n}}(t,s) + a_t(t,s,0)\,\frac{\dot{\vec{x}}(t_1,s)}{\left\|\dot{\vec{x}}(t_1,s)\right\|} \tag{6}$$

As can be seen in Figure 9, this dramatically improves our algorithm.

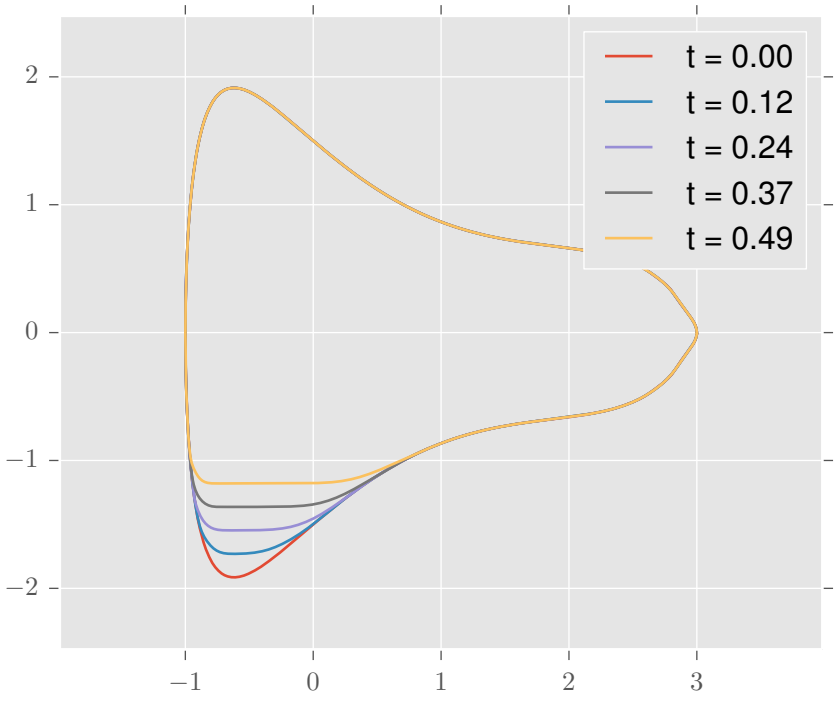

FIGURE 9. *Bottom recession process* simulation with point redistribution.

If we plot the evaluation points (Figure 10), it can be seen that the points nicely redistribute themselves as time increases to stay evenly spaced on the surface.

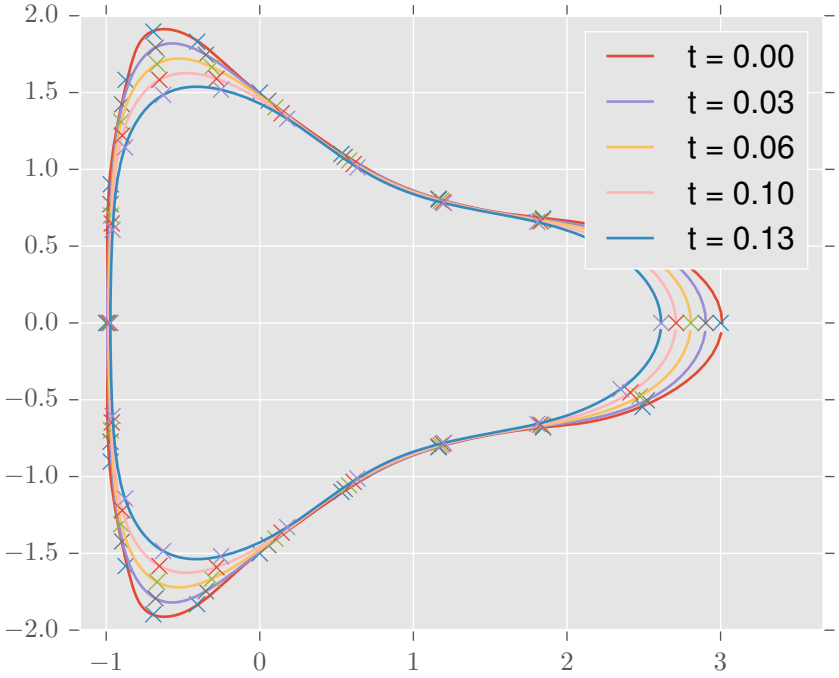

FIGURE 10. *Smoothing process* with evaluation points shown.

4. Conclusion

In this paper, we presented two models which could determine how stones are shaped by erosion. The first model was a discrete model based off of a mechanism for chipping a stone via shear forces. The second model was a continuous model that used Fourier transforms to model a stone being worn down. Each model targets a different aspect of erosion, and taken together, one can better understand the actual mechanism for the erosion of stones.

Appendix

Real Fourier Transform. The Real Fourier transform is defined as

$$\hat{\mathbf{x}} \coloneqq \mathcal{RFFT}(x)_k$$
$$\coloneqq \sum_{n=0}^{N-1} e^{-2\pi i k(n/N)} x_n \qquad \text{for} \quad k = 0, \ldots, N/2+1$$

Note: the terms FFT and Fourier transform are used interchangeably in the paper. Technically they are not synonyms, but it is common practice to use the terms interchangeably.

And the inverse transform is defined as

$$x \coloneqq \mathcal{RFFT}^{-1}(\hat{\mathbf{x}})$$
$$\coloneqq \frac{1}{N}\left(\sum_{k=0}^{N/2+1} e^{2\pi i k(n/N)} X_k + \sum_{k=N/2+2}^{N-1} e^{2\pi i k(n/N)} \bar{X}_k\right) \qquad \text{for} \quad n = 0, \ldots, N-1$$

References

[1] Fast Fourier Transform. (2013, 10 8). Retrieved from Wikipedia: http://en.wikipedia.org/wiki/Fft

[2] Trefethen, L. N. (2000). Spectral Methods in MATLAB. Philadelphia: SIAM.