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 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 eroding a stone from the bottom.

Date: November 8, 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 process of chipping is analogous to dropping a stone from a specified height and causing the stone to crack from hitting the ground. One can imagine this happening to a stone in a stream by being tossed and turned by the force of the river, causing the stone to collide with other stones on the bottom of the riverbed.

We model the interaction between two stones as a normal force. The point where a stone collides with another stone is the point where the normal force is concentrated. The stone breaks into two pieces at this point. The processes by which normal 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 our discrete model. We will represent a stone as a polygon in k-dimensional space with n vertices.

Definition 2.1. *Let s be a polygon consist of n vertices $s = \{\mathbf{v}_0, \ldots, \mathbf{v}_{n-1}\}$ where each vertex $\mathbf{v}_i \in \mathrm{R}^k$ is represented by 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:

$$\mathbf{c}_s = \frac{\int x g(x) dx}{\int g(x) dx} \tag{1}$$

Where $g(x) = 1$ for values $x \in s$ in the stone and $g(x) = 0$ otherwise. We will also introduce the notion of a convex vertex, which will be useful when defining our chipping process.

Definition 2.2. *A convex vertex $\mathbf{v}_c$ on a stone s with n vertices is a vertex $\mathbf{v}_i$ which, when removed from the polygon s to create a new polygon $s' = \{\mathbf{v}_0, \ldots, \mathbf{v}_{i-1}, \mathbf{v}_{i+1}, \ldots, \mathbf{v}_{n-1}\}$ 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. 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 pseudo-probability distribution $\mathrm{P}(0, 1)$, 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 $t \sim \mathrm{P}(0,1)$ and let $\mathbf{p}_1 = \mathbf{v}_j t + (1-t)\mathbf{v}_{j-1}$. Recall that $t \sim \mathrm{P}(0,1)$ denotes that t is drawn from the pseudo-probability distribution $\mathrm{P}(0,1)$.
(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 $\mathbf{p}_2 = \mathbf{v}_{j+1}$ if $l_r = l_2$ and $\mathbf{p}_2 = \mathbf{v}_{j-1}$ if $l_l = l_2$.
(6) Create a new stone s' whose vertices are given by

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

Here, the vertex $\mathbf{v}_j$ has been replaced by $\mathbf{p}_l$ and $\mathbf{p}_r$ in the list of vertices for s (where $\mathbf{p}_l$ is the point we chose on line $\mathbf{l}_l$ and $\mathbf{p}_r$ is the point we chose on line $\mathbf{l}_r$). The new stone s' now has one more line segment and vertex than the old stone s.

Now that we have defined a chip $Chip(\mathrm{P}, A, s)$, we can define our entire model as a chipping process.

Definition 2.3. *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 recursively conducted* k *chips. Thus, a chipping process takes a stone* s *and creates* $s_1 = Chip(\mathrm{P}, A, s)$*,* $s_2 = Chip(\mathrm{P}, A, s_1)$*, . . . ,* $s_k = Chip(\mathrm{P}, A, s_{k-1})$*, 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.

For our simulations, we use a pseudo-probability distribution $\mathrm{P}(0,1)$ which is a normal distribution with mean μ and standard deviation σ. We select $\mu \in [0,1]$ and all realizations of the normal distribution which fall outside of $[0,1]$ are discarded and we select a new element from the

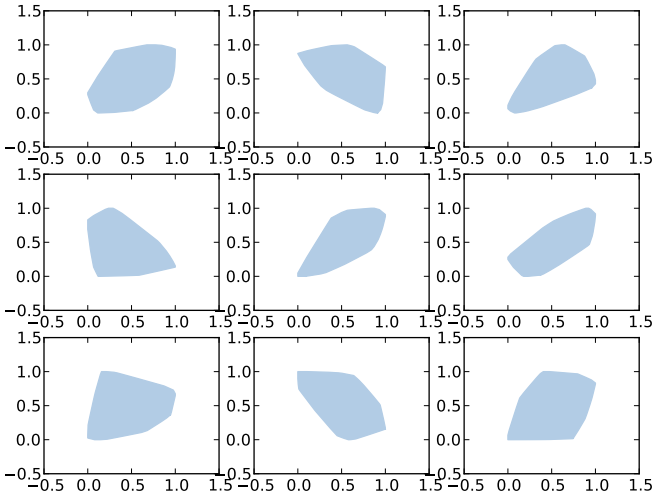

FIGURE 1. Chipping Process for a Unit Square

normal distribution. We will denote a pseudo-probability distribution which is using a normal distribution with $\mathrm{P}(0,1) = \mathrm{N}(\mu, \sigma)$.

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 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 random points on a stone from the centroid. We will generate a large number of random points, and plot their distances from the centroid.

More precisely, a random point $\mathbf{p}_s$ on the stone s will be selected by the following process:

(1) Produce a random angle $\theta \in [0, 2\pi)$, drawn uniformly from the range $[0, 2\pi)$.
(2) Draw a ray from the centroid of the stone with an angle θ from the horizontal (x-axis).
(3) The ray will intersect with the stone in exactly one location (since the stone is closed). Set the random point $\mathbf{p}_s$ as the point where the ray intersects an edge on the stone.

When we plot the distances, we will scale the distances to the centroid of a point $\mathbf{p}_s$, denoted by $d(\mathbf{p}_s, \mathbf{c}_s)$. We will sample q random points from the stone, and plot the scaled distances. Let $\min_d = \min d(\mathbf{p}_s, \mathbf{c}_s)$ be the minimum observed distance and $\max_d = \max d(\mathbf{p}_s, \mathbf{c}_s)$ be the maximum observed centroid distances for the q samples on a stone s. We will define the scaled distance to the centroid of a stone s as $d'_s(\mathbf{p}_s, \mathbf{c}_s)$ with the following formula:

$$d'_s(\mathbf{p}_s, \mathbf{c}_s) = \frac{d(\mathbf{p}_s, \mathbf{c}_s) - \min_d}{\max_d - \min_d} \tag{2}$$

We can get intuition for this metric by examining common shapes. In a circle, all of the distances to the centroid would be the same. In an ellipse, one would expect a skewed curve, with more points lying closer to the centroid. One can see this by examining all points on an ellipse which have distances from the centroid lying in $[d, d + \epsilon]$. The slices of the ellipse which are carved out will have strictly greater total angle than the angles carved from using distances in $[d + \epsilon, d + 2\epsilon]$.

Figure 2 provides a sample of scaled distances computed for an ellipse with a formula of $x^2 + \frac{y^2}{1.5} = 1$. We computed 20,000 random points on the ellipse, computed the scaled distances for each point from the centroid, then plotted it in figure 2. One can see that the resulting distribution is skewed towards closer points, just as we had predicted. Now, we will show the results of generating stones and aggregating the scaled distances across multiple stones from our chipping process.

Figure 3 shows the distribution of vertex distances which were simulated with the same parameters as used for figure 1. We generated 100 stones using our chipping process where each stone was chipped 50 times. Then we sampled 400 random points from each resulting stone. Figure 3 shows that the resulting distribution is skewed, with most

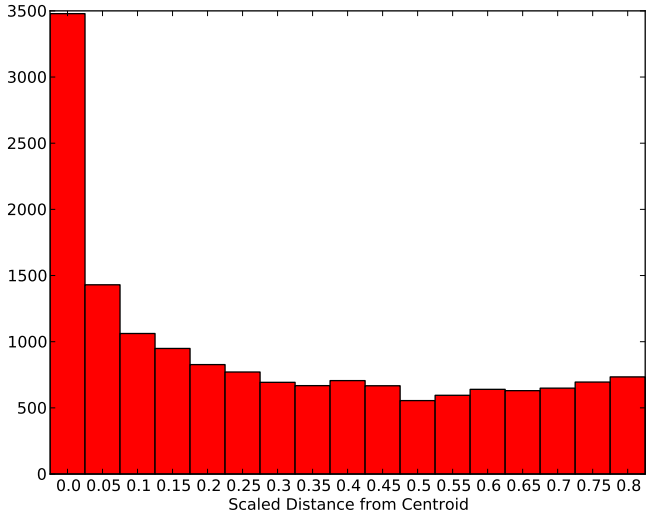

FIGURE 2. Histogram of Scaled Distances for an Ellipse ($x^2 + \frac{y^2}{1.5} = 1$)

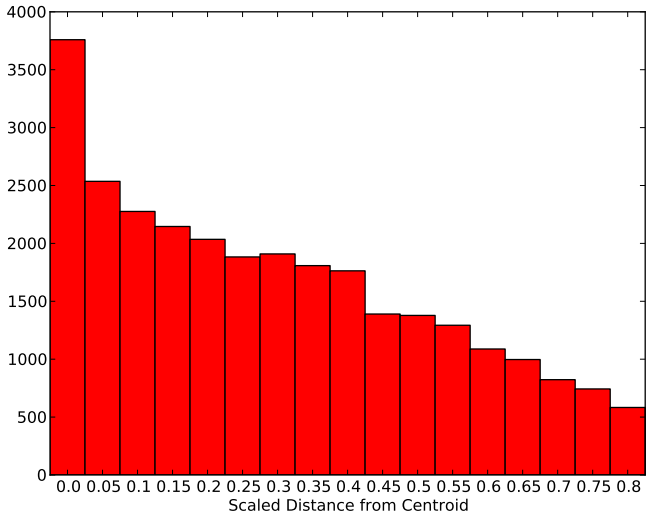

FIGURE 3. Histogram of Scaled Distances for 100 Generated Stones

points being closer to the centroid. The distribution of distances from the centroid shows that the shapes that result from this set of chipping process conditions have a similar distribution as an ellipse would.

We see that our chipping process model indeed provides a good approximation of a stone. The resulting stone distribution shows that the stones we generated were flat and long, much like stones one would find in the real world. We shall now present our continuous model of erosion, which will model the gradual erosion a stone.

3. Continuous Model

We next moved on to develop a continuous erosion model that operates on smooth surfaces, in contrast to our previous discrete model 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.

Continuous Surface. We will start by introducing a set of parametric equations to represent a surface. A parametric representation was chosen over an implicit one, so that we can use spectral methods and the FFT when integrating Equation 3, which we will discuss in detail in Section 3.

Let us introduce $s \in [0, 2\pi)$ as our parameter with s increasing counterclockwise along the surface. The coordinates of any point on the surface can be represented at time $t \geq 0$ by two functions $x(t, s), y(t, s)$. Coordinates can be written more compactly as

$$\mathbf{r}(t, s) \coloneqq (x(t, s), y(t, s))$$

Note: The reader should keep in mind that derivative operators still apply to each coordinate individually, e.g.

$$\frac{\partial \mathbf{r}(t, s)}{\partial t} \coloneqq \left(\frac{\partial x(t, s)}{\partial t}, \frac{\partial y(t, s)}{\partial t} \right)$$

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

$$\dot{\mathbf{r}}(s) \coloneqq \frac{\partial \mathbf{r}(s)}{ds}$$
$$\ddot{\mathbf{r}}(s) \coloneqq \frac{\partial^2 \mathbf{r}(s)}{ds^2}$$

The equations for the initial shape that we will use in our simulations are shown below and a plot of the shape is shown in Figure 4 (T_i is the i$^{\text{th}}$ Chebyshev polynomial)

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

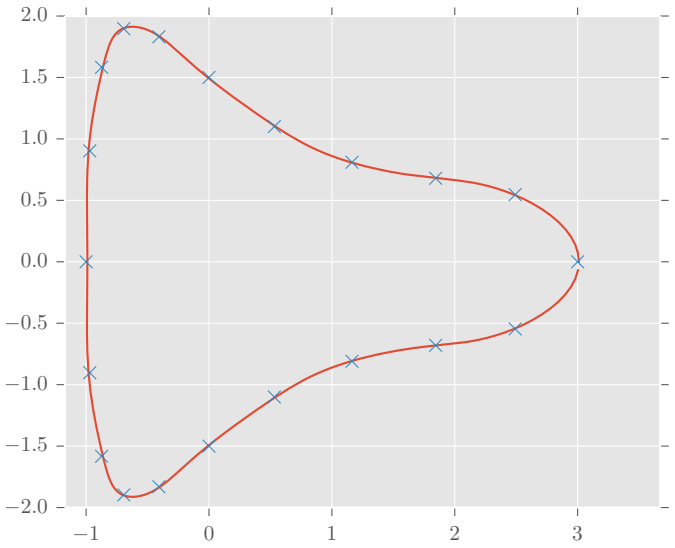

FIGURE 4. Example shape.

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.

Continuous Erosion Processes. All of the processes that we will look at in this section, will move points on the surface in the direction of the surface normal, which is defined as

$$\hat{n}(\mathbf{r}(s)) = (-\dot{y}(s), \dot{x}(s))$$

We can now model our erosion process with the following differential equation, where $g(\mathbf{r}(s))$ is a function specific to the erosion process being modeled.

$$\frac{\partial \mathbf{r}(t,s)}{\partial t} = g(\mathbf{r}(s))\ \hat{n}(\mathbf{r}(s)) \tag{3}$$

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, bumps on the object will get worn down quickest, because they are the first parts 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 assume that every point on the surface wears 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.

$$\kappa(\mathbf{r}(s)) = \frac{\dot{\mathbf{r}}(s) \times \ddot{\mathbf{r}}(s)}{\|\dot{\mathbf{r}}(s)\|^3}$$

To make bumps erode faster than indentations, we will use the following damping function

$$g(\mathbf{r}(s)) = \tan^{-1}(\beta\,(\kappa(\mathbf{r}(s)) - \alpha)) + \frac{\pi}{2} \tag{4}$$

For our simulations we chose $\alpha = 1,\ \beta = 5$ so that $g(\mathbf{r}(s)) \approx 1$ at the corners on our initial surface. This restriction is not based on the physics of the problem, and was chosen purely for the aesthetic look of the resulting simulation.

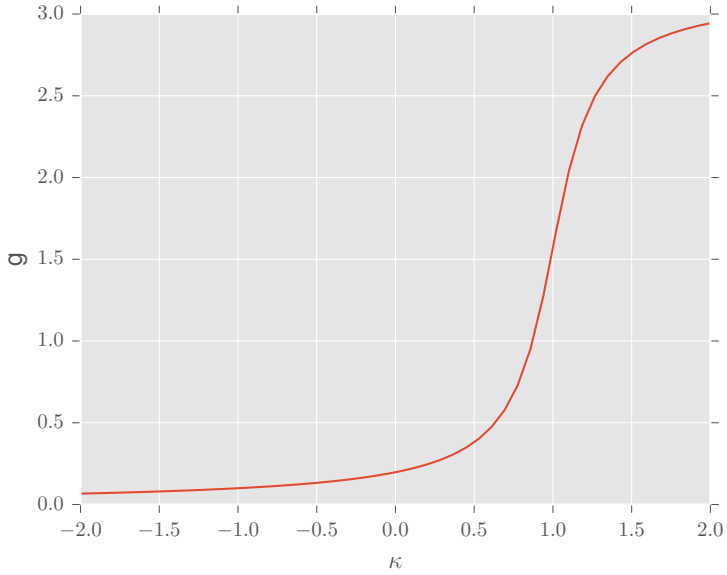

FIGURE 5. Plot of $g(\kappa)$ for *Smoothing Process*.

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 $y(s)$) on the surface will erode quickest.

$$f(s) = \frac{1}{\alpha\,(y(s) - y_{min}) + \beta} - \gamma$$

$$g(s) = \begin{cases} f(s) & \text{for} \quad f(s) > 0 \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

For our simulations we chose $\alpha = 10,\ \beta = \gamma = 0.1$ so that $g(\mathbf{r}(s)) \approx 1$ for $y(s) \approx y_{min}$.

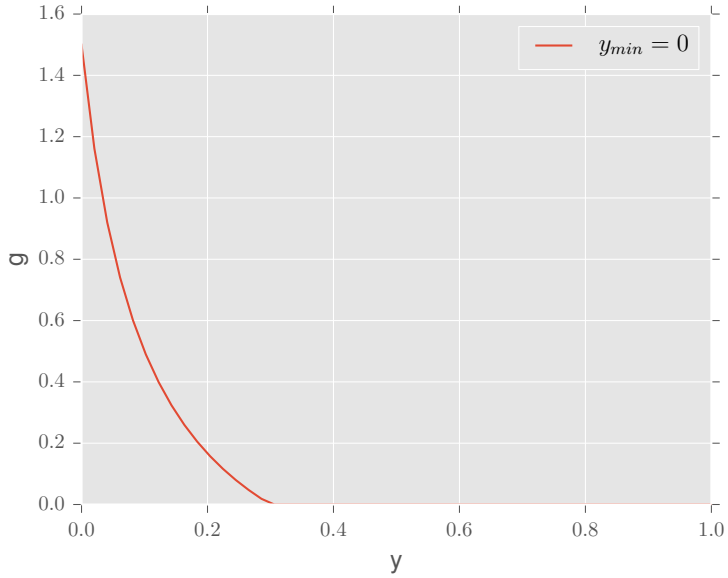

FIGURE 6. Plot of $g(y)$ for *Bottom Recession Process*.

Numerical Modeling. We chose to model the parametric surface functions x, y with discrete Fourier series. This is because spectral methods using the DFT 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. The DFT and surface derivatives can be calculated quickly using the Fast Fourier Transform (refer to Appendix A for more details).

To integrate Equation 3, we sample the surface at parameter values $S_n = 2\pi \frac{n}{N}$, $n = 0, \ldots, N-1$, which gives us a matrix of initial evaluation points

$$\begin{aligned} X_0 =& \mathbf{r}\,(0, S) \\ =& \begin{bmatrix} x(0, S_1) & y(0, S_1) \\ \vdots & \vdots \\ x(0, S_N) & y(0, S_N) \end{bmatrix} \end{aligned}$$

Note: When refering to vectors and matrices in our numerical models, we will use uppercase letters to distinguish them from continuous functions.

Forward time integration of Equation 3 can then be implemented with standard ODE time stepping integration methods. In this paper, we used SciPy's `odeint` method for integrating Equation 3.

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 surface forms loops as time is increased (a physical impossibility).

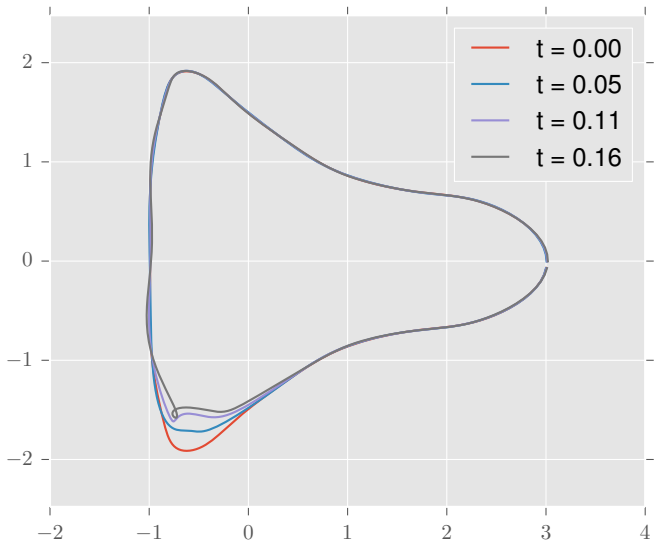

FIGURE 7. Time integration of 3 showing loop artifact.

To investigate why these loop artifacts occur, we will show a zoomed in portion of the same figure, with the evaluation points shown.

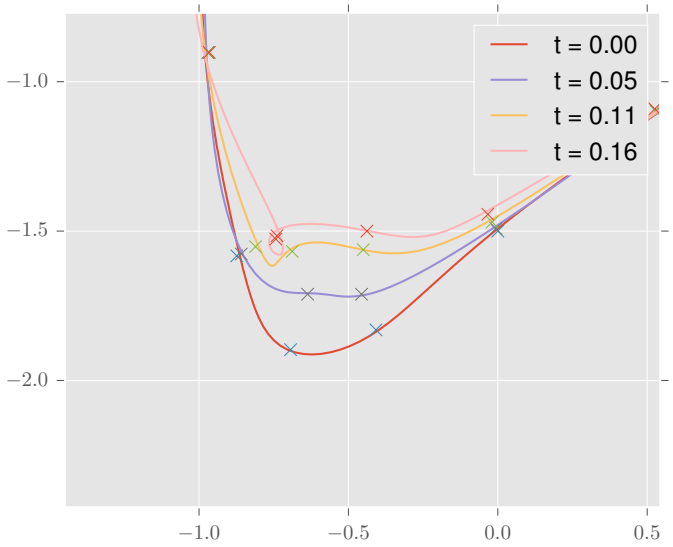

FIGURE 8. Expanded view of loop artifact.

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 resample to get a set of evenly spaced evaluation points on the surface. This wouldn't be too difficult of a task, since we can integrate and sample at any point on the surface via the FFT. Unfortunately, if we pick points on the surface at arbitrary values of s, we lose spectral accuracy, because we are now sampling at unevenly spaced values of our parameter s, which we will call $\tilde{S}$. The Discrete Fourier transform assumes that our points are sampled at evenly spaced values of s. Although we are working with the discrete Fourier transform, we need to return to the continuous equations to understand the problem. Redistribution in the continuous sense is a parameter transformation

$$\tilde{s} = h(s)$$

with the condition of even spacing being

$$\left\| \frac{\partial \mathbf{r}(s)}{\partial \tilde{s}} \right\| = constant$$

Our shape function now becomes

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

There is absolutely no guarentee that the Fourier series of this composite function will decay at a reasonable rate, so using the DFT with our chosen resolution N could very poorly capture the surface shape. The method is not withough hope, since the technique of parameter transformation is utilized extensively in spectral methods such as the transformation $h(s) = \cos(s)$ used for Chebyshev spectral methods to cluster evaluation points near the endpoints of the domain. We need to find a way to generate a parameter transformation $h(s)$ where $(x \circ h)(s)$ has a rapidly decaying Fourier series.

Continuous Redistribution. If we fix a time $t = t_1$, then we can define a differential equation that will allow us to redistribute the points and ensure that our redistributed points will still have a rapidly decaying Fourier series.

Physically $\dot{\mathbf{r}}(t_1, s)$ is the surface tangent vector, and $\ddot{\mathbf{r}}(t_1, s)$ is the rate of change of this surface tangent vector with respect to s. More importantly though, $\|\dot{\mathbf{r}}(t_1, s)\|$ is inversely proportional to the density of evaluation points on the surface at s. We would like to shift the evaluation points away from denser areas, thus we need move the evaluation points in the direction of larger $\|\dot{\mathbf{r}}(t_1, s)\|$, which is the projection of $\ddot{\mathbf{r}}(t_1, s)$ onto the surface tangent vector.

$$a_t(t, s) = \ddot{\mathbf{r}}(t, s) \cdot \dot{\mathbf{r}}(t, s) \tag{6}$$

We can now set up a differential equation for this redistribution process.

$$\frac{\partial \mathbf{r}(t_1, \tilde{s}(t'))}{\partial t'} = a_t(t_1, \tilde{s}(t')) \frac{\dot{\mathbf{r}}(t_1, \tilde{s}(t'))}{\|\dot{\mathbf{r}}(t_1, \tilde{s}(t'))\|}, \quad \tilde{s}(0) = s \tag{7}$$

To simplify the numerical computation, we can combine Equations 3 and 7 to get a single ODE

$$\frac{\partial(\mathbf{r}(t,s))}{\partial t} = g(t,s)\ \hat{\mathbf{n}}(t,s) + a_t(t,s)\ \frac{\dot{\mathbf{r}}(t_1,s)}{\|\dot{\mathbf{r}}(t_1,s)\|} \tag{8}$$

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

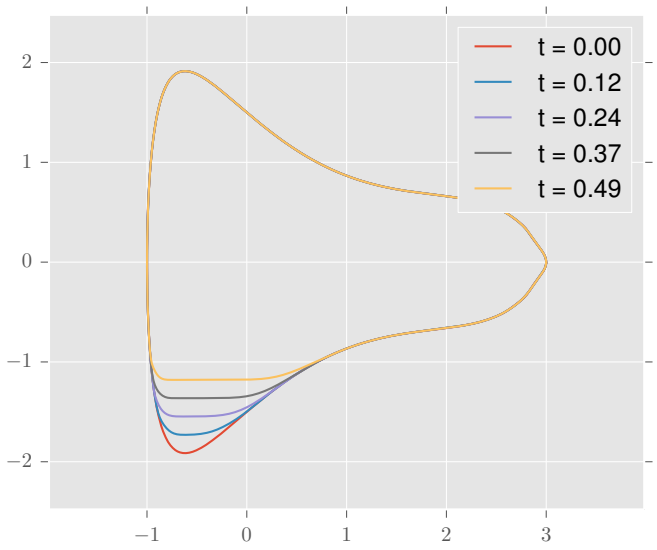

FIGURE 9. *Bottom recession process* simulation with point redistribution.

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.

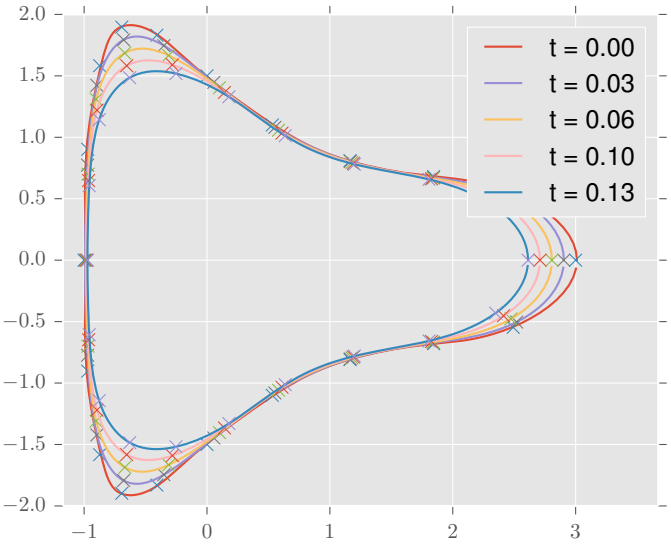

FIGURE 10. *Smoothing process* simulation with evaluation points shown.

Higher dimensions. A next logical step would be to generalize the algorithm to a 2-dimensional surface that would be parameterized by $s_1, s_2 \in [0, 2\pi)$. Much of the algorithm would be similar, except that we would need to use the 2-dimensional discrete Fourier transform (refer to Appendix A).

The surface normal vector equation for a surface in 3-dimensional space is the following

$$\mathbf{n}(t, s_1, s_2) = \frac{\partial \mathbf{r}(t, s_1, s_2)}{\partial s_1} \times \frac{\partial \mathbf{r}(t, s_1, s_2)}{\partial s_2}$$

Curvature κ could be generalized to mean curvature H

$$\begin{aligned} H(t, s_1, s_2) &= \frac{\nabla \cdot \mathbf{n}(t, s_1, s_2)}{2} \\ &= \frac{1}{2}\left(\frac{\partial \mathbf{n}_1(t, s_1, s_2)}{\partial s_1} + \frac{\partial \mathbf{n}_2(t, s_1, s_2)}{\partial s_2}\right) \end{aligned}$$

Avoiding singularities when mapping the 2-dimensional surface to an N x N array would likely be difficult, but the mapping only needs to be used to sample the initial points on the object, and the redistribution algorithm can be oblivious to the mapping.

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 A. The DFT

The Discrete Fourier transform is defined as

$$\begin{aligned}\hat{\boldsymbol{x}}_k &:= \mathcal{F}_k(\boldsymbol{x}) \\ &:= \sum_{n=0}^{N-1} e^{-2\pi i k(n/N)} \boldsymbol{x}_n \qquad \text{for } k = 0, \ldots, N-1\end{aligned}$$

And the inverse transform is defined as

$$\begin{aligned}\boldsymbol{x} &:= \mathcal{F}_n^{-1}(\hat{\boldsymbol{x}}) \\ &:= \frac{1}{N} \sum_{k=0}^{N-1} e^{2\pi i k(n/N)} \hat{\boldsymbol{x}}_k \qquad \text{for} \quad n = 0, \ldots, N-1\end{aligned}$$

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

$$\frac{\partial^p \boldsymbol{x}}{\partial s^p} = \mathcal{F}^{-1}(\hat{\mathbf{w}})$$

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.