2020

Modeling the Evolution of Barrier Islands

Greg Robson
Virginia Commonwealth University

Modeling the Evolution of Barrier Islands

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Science at Virginia Commonwealth University.

by

Gregory Caldwell Robson
Master of Science

Director: Dr. David Chan, Associate Professor
Department of Mathematics and Applied Mathematics

Virginia Commonwealth University
Richmond, Virginia
May, 2020

Table of Contents

List of Figures

Abstract

Barrier islands form off the shore of many coastal areas and serve as the first line of defense, protecting littoral communities against storms. To study the effects that climate change has on barrier islands, we use a cellular model of wind erosion, surface dynamics, beach dynamics, marsh dynamics, and vegetation development. We will show the inhibition of movement when vegetation is present.

Chapter 1

Introduction

1.1 Background Information

Barrier islands are small islands, typically long and narrow, which lie offshore many coastal regions. They are ecologically diverse and undergo continual reshaping through wind and water erosion. Along many coastlines, barrier islands are the first line of defense for littoral communities against the impact of large, damaging storms. Changes in global climate seem to correlate with more frequent storm patterns and often more energetic storms [7]. We model the evolution of barrier islands in part to aid in predicting how global climate change affects their evolution.

The evolution of barrier islands depend on sediment erosion from wind and water, beach dynamics, vegetation, as well as sea level rise and wave conditions [8]. Many barrier islands, such as Hog Island in Virginia or the Outer Banks in North Carolina, have a marsh or lagoon on the nearshore side, which is also important to the development of the island. We expand on preexisting models to create a model which considers all of these features on the entire island.

In 1991, E.B. Rastetter wrote his ISLAND model that simulates annual changes in vegetation, geomorphology, water table depth, and average groundwater salinity on a cross-sectional transect [11]. He details three submodels: a vegetation submodel, a geomorphology submodel, and a groundwater submodel which operate on one-year time steps. The algorithm simulates three classifications of plants in various life stages using a Markov process. At each time step there is a probability that a seed will germinate or die. If the seed germinates, at the next time step there is a probability that the sprout will mature or die. If the sprout matures, at the next time step there is a probability that the plant will survive or die. This procedure is simulated for the necessary number of life stages per plant type which included grasses, annuals, and perennials [11].

In particular, the ISLAND model is not broad in some aspects and perhaps too specific in another. In our model, we simulate processes over an entire island, not horizontal transects. Additionally, we are more interested in understanding the impact plants have on the geomorphology of the island and less interested in the specifics of each plant's life cycle.

In 1995, Werner employed a cellular automaton that views sand on the island as movable slabs [14]. He implemented erosion and deposition rules that rely on the presence or absence of sand and shadow zones. We will describe the shadow zones with details of a more recent model. Werner's model is often the basis for most modern island and dune models. However, his model fails to consider how plants impact the erosion of sand, which would naturally affect the formation of dunes. His model also does not include storms or marine processes, which are essential to the evolution of barrier islands. [14].

In 2007, J. M. Nield and A. C. W. Baas wrote their DECAL (acronym for Discrete ECogeomorphic Aeolian Landscape) algorithm. The DECAL algorithm is a cellular automaton which models dune formation in association with vegetation. They reference Werner's model by saying that "in polling *without* replacement, every grid cell is polled exactly once per iteration... polling *with* replacement allows some slabs to remain dormant.... [this] smooths the landscape forms." In polling with replacement, some slabs are selected multiple times meanwhile some slabs are not selected at all. In polling without replacement, each grid cell is polled once. When simulating wind erosion, the DECAL model polls *without* replacement to avoid this artificial smoothness. Instead, they define an angle of repose (30° for bare sand) for which bare sand avalanches and a shadow zone, in the lee of the dunes. The shadow zones here are determined by a probability function. This function yields the probability that the transported slab gets deposited or erodes to the next adjacent neighbor [9].

The goal of the DECAL model is to understand and classify the different shapes of dunes, especially in response to the presence of plants. In particular, they observe that "nebkha dunes form in association with mesquite-type shrubs and hairpin parabolic dunes form in association with vegetation succession" [9]. DECAL does not consider marine processes or storms.

In 2016, Keijsers, De Groot, and Riksen created their DUBEVEG model (acronym for DUne BEach VEGetation). DUBEVEG is a cellular model which consists of three components: aeolian transport, marine processes, and vegetation. Similar to the DECAL algorithm, DUBEVEG also polls cells without replacement. The downsides of DUBEVEG is that wind is assumed to be unidirectional and constant velocity throughout the simulations. Moreover, they only model the dune side of the island and not the entire island. Lastly, the model shapes only the initial foredune and does not extend into the "interior" dune field.

There have been numerous models which simulate a smooth beach profile, such as Dean [3], Davidson-Arnott [2], and Fiedler et al. [6]. These models do not consider beyond the beach profile. Davidson-Arnott based their model off of The Bruun Model [1] from 1962 which only considers a two-dimensional profile. While barrier islands have been heavily researched for decades, there do not exist models which simulate evolution of the entire island with vegetation. We have combined aspects of several of the aforementioned models to do just that.

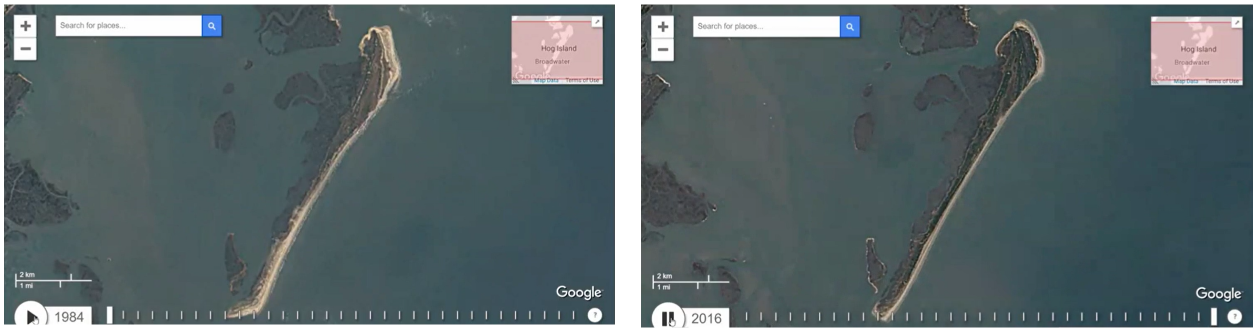

Figure 1.1: Left: Hog Island 1984. Right: Hog Island 2016.

The goal for our model is to closely mimic a realistic process by which dune fields form across an island, to simulate a realistic beach profile through marine processes and form an initial foredune, to simulate vegetation both as a biotic feature and an ecological force in preventing erosion, and to simulate and maintain a low-elevation growing marsh. In Figure 1.1 we have images of Hog Island taken 32 years apart. Notice how the sandy coast appears to retreat towards the vegetated regions. The marsh remains fairly stagnant but undergoes some slight reshaping. Also notice how the north and south tips of the island have changed shape a bit more drastically than the beach and marsh.

In Chapter 2 we describe the different components of the model: plants, aeolian transport and avalanche, and marine and marsh processes. In Chapter 3 we present the results from four simulations. Two involve a strong, constant northeastern wind. Of these two, one island is without any vegetation and one island is covered with vegetation. Also, two simulations involving a strong, constant western wind with one island without vegetation and one island covered in vegetation. Finally, in Chapter 4, we summarized the results and discuss future directions.

Chapter 2

Model Structure

We constructed a cellular model that manipulates several $n \times m \times 2$ matrices which represent different components of a barrier island. We use timesteps, T, of two-weeks in length. The elevation matrix H stores an integer at each location (i, j) where $H(i, j, 1)$ represents the number of rectangular slabs, relative to sea level, with thickness δ meters and dimension $b_1 \times b_2$ square meters (in our model, $b_1 = b_2 = b$). For positive integers at cell (i, j) we have that number of slabs above sea level, and for negative integers we have an elevation of $H(i, j, 1) \cdot \delta$ meters below sea level. $H(i, j, 2)$ stores integers $x \in \{-1, 0, 1, 2\}$ to separate the island into geographical categories. These categories are as follows:

$H(i, j, 2) = -1$: location (i, j) is underwater,

$H(i, j, 2) = 0$: location (i, j) is in the dune field,

$H(i, j, 2) = 1$: location (i, j) is on the beach, and

$H(i, j, 2) = 2$: location (i, j) is in the marsh.

The other matrices represent four species of plants and are named $P_k(i, j, 1)$ for $k \in [4]$. P_k stores a value $x \in \{-1, [0, 1]\}$ which represents the percent of the $a \times b$ cell (i, j) which is occupied by plant species k. For $P_k(i, j, 1) > 0$, we say the plant *exists* at cell (i, j). If $P_k(i, j, 1) \geqslant 0$ we say the cell is *viable* and if $P_k(i, j, 1) = -1$ then cell (i, j) is considered *unviable* for plant species k. A cell (i, j) being unviable is typically because the cell is underwater or due to unfavorable species–specific elevations (which can be understood as a plant's access to fresh water). The second layer in plant matrices, $P_k(i, j, 2)$ also stores categorical data and is defined as follows:

$$P_k(i, j, 2) = h\text{: species } k \text{ has roots at elevation } h \cdot \delta.$$

This is useful in determining root exposure and plant burial, two important aspects of plant death. Due to a lack of available data, this aspect of the model is not currently used.

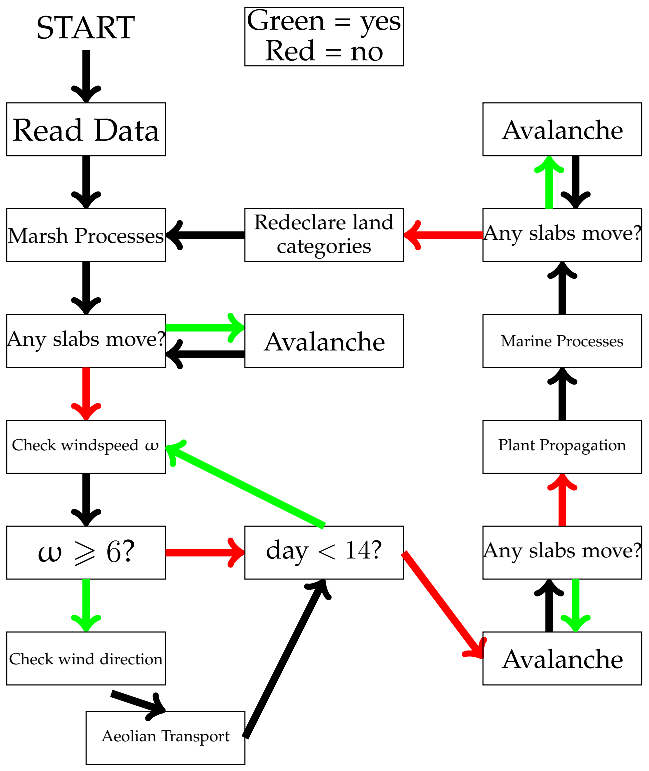

Figure 2.1: Flow chart of model

Each time step, T, we simulate the evolution of barrier islands through the following list of components: plant growth/death, aeolian (sand) erosion, avalanche/smoothing process, updating a beach profile, and updating a marsh. The diagram in Figure 2.1 shows the order of processes in this model. We first go through the marsh processes.

If any slabs move, we check to see if anything should avalanche. Then, we check aeolian transport if the wind speed is strong enough. This is followed by determining the wind direction and polling slabs to be eroded downwind. We check windspeed, ω, 14 times per timestep, to represent each day during each timestep T. At that point, if a cell moves, we implement avalanche again, to ensure a smooth terrain. We then allow plants to grow, spread, or die before implementing marine processes, to maintain a smooth beach. We check avalanche once again before declaring the land categories and going to the next time step, $T + 1$.

2.1 Plant Propagation

2.1.1 Introduction of Plant Species

Perhaps the most important aspect of our model is the plants. In particular, modeling an entire barrier island evolution with the inclusion of plants is novel. Our goal here is to simulate the life cycle of several relevant species of plants and show their impact on the evolution of a barrier island. In this section, we detail the methods we employed to simulate plant growth and death, the spread or propagation of plant species, and the impact plants have on erosion.

The four species included in the model and their respective matrices are: Ammophila breviligulata (P_1), Spartina patens (P_2), Morella cerifera (P_3), and Spartina alterniflora (P_4). Ammophila breviligulata is the main dune-building grass present on Hog island. Spartina patens and Spartina alterniflora are both marsh grasses, but Spartina (marsh) grows exclusively in the marsh while Spartina patens behaves more similarly to a dune grass, in most characteristics. That is, it tends to grow on dunes rather than in the marsh. Morella is a woody shrub.

Plant Parameters 1		
Parameter	Name	Value (in meters)
$[m_1, M_1]$	Viable Elevations P_1	$[1, 5]$
$[m_2, M_2]$	Viable Elevations P_2	$[0.75, 3]$
$[m_3, M_3]$	Viable Elevations P_3	$[1.5, 2.5]$
$[m_4, M_4]$	Viable Elevations P_4	$[-0.5, 1]$

Table 2.1

We determine that a cell (i, j) is viable for species k if $m_k \leqslant H(i, j, 1) \cdot \delta \leqslant M_k$. Otherwise, the cell is unviable (see Table 2.1).

2.1.2 Grass Growth and Dispersal

Plant propagation varies by species. In particular, grasses propagate differently from Morella cerifera, the shrub. When simulating plant growth, we check all locations where species k exists. For each cell (i, j) for which $P_k(i, j, 1) > 0$, there is probability p_{grow} that $P_k(i, j, 1)$, $k \neq 3$, increases by $\alpha_{\text{propagate}}$.

When simulating seed dispersal, we also check all locations where species k exists. For each cell (i, j) for which $P_k(i, j, 1) > 0$, we consider the 8-cell neighborhood around (i, j) (see Figure 2.2). There is probability p_{spread} that $P_k(i', j', 1)$ for $k \neq 3$, increases by $\alpha_{\text{propagate}}$. It is possible that $P_k(i, j, 1) > 1$ but we will address our solution later. Note that, for grasses, growth and dispersal can be performed simultaneously by considering only propagation and introducing cell (i, j) to the 8-cell neighborhood.

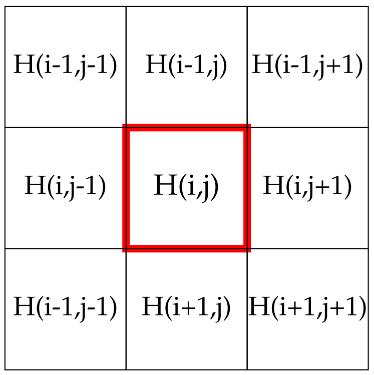

Figure 2.2: 8-cell neighborhood for grass propagation

2.1.3 Shrub Growth and Dispersal

To simulate shrub growth, for each cell (i,j) where $P_3(i,j,1) > 0$, there is probability p_{grow} that $P_3(i,j,1)$ increases by a percentage α_{shrub} of itself, where the probability for the value of α_{shrub} is uniformly distributed over the provided interval. That is, in the event that $P_3(i,j,1)$ does grow, we can define an equation for P_3' to be:

$$P_3'(i,j,1) = P_3(i,j,1) \cdot \left(1 + \alpha_{\text{shrub}}\right), \tag{2.1}$$

where P_k' is the new percent that species k occupies on cell (i,j), after growing. Since the woody shrub is much larger than a blade of grass, it is important for Morella cerifera to grow with respect to its existing self. Let us now define for an arbitrary matrix M, let M' be the updated version of matrix M, for the next time step.

Seed dispersal for the shrub is done by local birds who eat the seeds of Morella cerifera and deposit them throughout the island, after digestion. To simulate this, for each cell (i,j) where $P_3(i,j,1) \geqslant 0$, there is probability p_{drop} that $P_k(i,j,1)$ increases by $\alpha_{\text{propagate}}$. Again, we will fix this later.

If a plant species newly exists on a cell (i,j) which formerly did not contain any of that plant species, we set $P_i(i,j,2) = H(i,j,1)$ to record the initial elevation of the new plant.

2.1.4 Plant Death

There are three ways that a plant may die in our model: a random life cycle process, a plant being placed at an unsupportive elevation, and death by competition. To simulate routine plant death, for each cell (i,j) for which $P_k(i,j,1) > 0$. There is a probability p_{death} that $P_k(i,j,1)$ decreases by β. If $P_k(i,j,1) < \beta$ and decreases by β during this timestep, then $P_k'(i,j,1) = 0$

To simulate plant death by elevation, we consider the viability of each cell. If $\big(H(i,j,1)\cdot$

Plant Parameters 2		
Parameter	Name	Value
p_{grow}	Probability for growth	0.5
p_{spread}	Probability for propagation (grass)	0.5
p_{drop}	Probability for propagation (shrub)	0.1
$\alpha_{\text{propagate}}$	Growth amount when propagating	$[0.01, 0.05]$
α_{shrub}	Growth (shrub)	$[0.05, 0.15]$
β	Death (routine)	$[0, 0.04]$
γ_1	Erosion Coefficient P_1	$\frac{2}{3}$
γ_2	Erosion Coefficient P_2	$\frac{1}{3}$
γ_3	Erosion Coefficient P_3	1
γ_4	Erosion Coefficient P_4	$\frac{1}{3}$

Table 2.2

$\delta) \notin [m_k, M_k]$ and $P_k(i,j,1) > 0$, we set $P_k(i,j,1) = 0$. As an exception, the marsh will have similar elevations to the beach but these regions have very different vegetative properties. We reflect this in our model by determining the viability of Spartina alterniflora by each grid cell's land category. That is, for any cell (i,j) where $H(i,j,2) = 2$, we set $P_k(i,j,1) = 0$ for $k \neq 4$. Likewise, for any cell (i,j) for which $H(i,j,2) \neq 2$, we force $P_4(i,j,1) = 0$.

There is still one more special case when considering plant death by elevation. Without the presence of plants, islands tend to drift westward (that is, it moves with the tidal forces). In particular, Morella cerifera responds to this. That is, Morella cerifera prefers to grow on the nearshore side of dunes away from the salt water spray of the ocean. As the sand moves away from the beach, any Morella cerifera which lives on the foreshore slope will experience rapid root exposure through wave attack and high exposure to salinity. This is not conducive to Morella cerifera. So, similarly to how we force marsh slabs to contain only Spartina alterniflora, we require that the beach does not contain any Morella cerifera. That is, if $H(i,j,2) = 1$ we force $P_3(i,j,1) = 0$.

Lastly, a cell (i,j) is *oversaturated* with plants if $P_1(i,j,1) + P_2(i,j,1) + P_3(i,j,1) + P_4(i,j,1) = 1 + a$, for $a > 0$. This indicates that the total plant density on the specified cell is greater than 100%, which is not possible. Since Morella cerifera out–competes the grasses when competing for room to grow, [12] a cell (i,j) with $\sum_{k=1}^{4} P_k(i,j,1) = 1 + a$, we define for each of the grasses:

$$P'_k(i,j,1) = P_k(i,j,1) - \frac{a}{\text{\# of non-Morella species present on cell } (i,j)}, \tag{2.2}$$

where P'_k represents the corrected population density for the plant species corresponding to P_k, $k \neq 3$.

2.2 Aeolian Transport

2.2.1 Main Goal

Wind is an influential force of evolution on the island. We consider windspeeds between ω_{min} and ω_{max} to trigger aeolian (wind) erosion [4]. Windspeeds above ω_{max} are classified as storms and are beyond the scope of this model. Windspeeds below ω_{min} are too light to erode sand [4]. For windspeeds ω between $\omega_{\text{min}} \leqslant \omega \leqslant \omega_{\text{max}}$, we simulate a natural process by which wind carries sand downwind and deposits them to adjacent cells. To simulate proper weather patterns, we allow aeolian transport, at most, 14 times in a time step (once for each day).

2.2.2 Windspeed

We first determine the windspeed based on Hog Island wind data [10], from 2007 to 2012. If the windspeed is greater than or equal to ω_{min} then the erosion process is initiated. We calculate the probabilities of the event that a wind direction is selected using the following parameters: $d_{\text{tot (speed)}} = d_{\text{still}} + d_{\text{storm}} + \sum_{k=\omega_{\text{min}}}^{\omega_{\text{max}}} d_k$. Here, d_{still} is the number of days with windspeed less than the minimum threshold velocity needed to erode bare sand, ω_{min}, d_{storm} is the number of days with windspeed greater than the minimum threshold velocity for storms (or maximum velocity for regular wind events), ω_{max}, and

Wind Erosion Parameters 1		
Parameter	Name	Value
ω_{min}	Minimum windspeed (to erode)	6 m/s
ω_{max}	Storm threshold windspeed	16 m/s
d_{still}	Number of (real data) events with windspeed $< \omega_{\text{min}}$	2116
d_k	Number of (real data) events with windspeed $\geqslant \omega_{\text{min}}$	$[0, 47]$
d_{storm}	Number of (real data) events with windspeed $> \omega_{\text{max}}$	0
$d_{\text{tot (speed)}}$	Number of (real data) wind events	2184

Table 2.3

d_k is the number of days with windspeed exactly equal to k m/s, where k is an integer between ω_{min} and ω_{max}.

The event of a specific windspeed being randomly selected occurs with probabilities $p_{\text{still}} = \frac{d_{\text{still}}}{d_{\text{tot (speed)}}}$, $p_k = \frac{d_k}{d_{\text{tot (speed)}}}$, and $p_{\text{storm}} = \frac{d_{\text{storm}}}{d_{\text{tot (speed)}}}$. We call the determined windspeed ω which is randomly selected. The probability that $\omega = k$ is p_k. The probability of a storm occurring is p_{storm} and the probability of no wind event occurring (i.e. the windspeed is too low) is p_{still}.

2.2.3 Wind Direction

When aeolian transport of slabs occurs, we also determine the wind direction based on the same data mentioned above [10]. Wind directions were provided as degree measures on the unit circle with $0°$ being north, $90°$ being west, and so on. We used neighborhoods of $22.5°$. These neighborhoods are shown in the figure below. If X represents one of the eight major cardinal directions, d_X represents the number of days with wind direction in the neighborhood of X. Naturally, $d_{\text{tot (dir.)}} = \sum_X d_X$ where the sum is over all of the eight cardinal directions that X can represent. Then, each wind direction X is selected with probability $p_X = \frac{d_X}{d_{\text{tot (dir.)}}}$

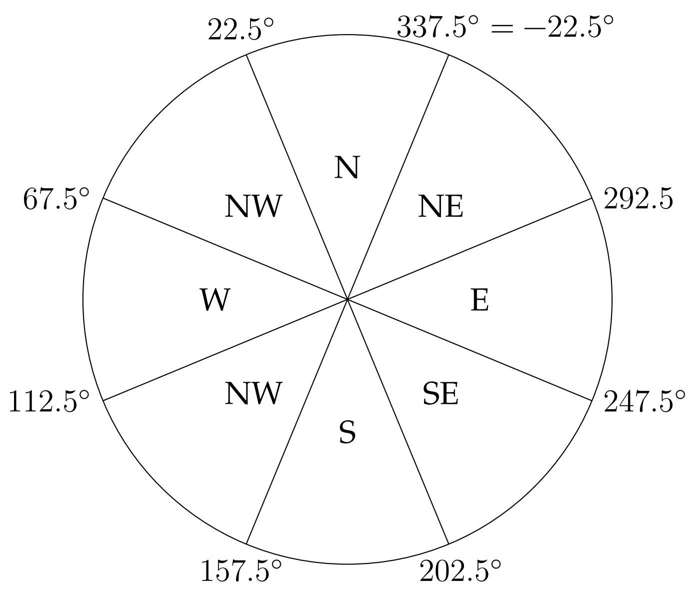

Figure 2.3: Cardinal directions viewed on a unit circle rotated $90°$ counter-clockwise

Wind Erosion Parameters 2		
Parameter	Name	Value
d_X	Number of days with wind direction X	$[0, 405]$
$d_{\text{tot (dir.)}}$	Total number of days of wind data	2184
p_{downwind}	Probability that slab erodes downwind	0.5
p_{port}	Probability that slab erodes to the port side of downwind cell	0.25
$p_{\text{starboard}}$	Probability that slab erodes to starboard side of downwind cell	0.25

Table 2.4

2.2.4 Transporting Slabs (Aeolian)

After determining the wind direction for an eroding slab, there is a probability, p_{erosion}, for each cell (i, j) that the slab erodes to a downwind neighborhood, or its *destination cell*. p_{erosion} depends on the local vegetation. We define a weighted value for the percent cover of plants, PC_{erosion}:

$$PC_{\text{erosion}}(i, j) = \gamma_1 P_1(i, j, 1) + \gamma_2 P_2(i, j, 1) + \gamma_3 P_3(i, j, 1) + \gamma_4 P_4(i, j, 1), \tag{2.3}$$

$0 \leqslant PC_{\text{erosion}} \leqslant 1$. Now, we define p_{erosion}:

$$p_{\text{erosion}} = (1 - PC_{\text{erosion}}(i, j)) \cdot \frac{\omega - \omega_{\min}}{\omega_{\max} - \omega_{\min}}. \tag{2.4}$$

Note that as as PC_{erosion} increases, p_{erosion} decreases and as ω increasing, p_{erosion} increases. However, a destination cell is only viable if the angle between the elevation of the two cells is less than 15° [9]. Also, note that p_{erosion} might be negative if $\omega < \omega_{\min}$ however, in this case, as mentioned above, nothing happens so this case is negligible.

With a probability of p_{erosion}, we move the topmost slab for cell (i, j) downwind. That is, we set $H'(i, j, 1) = H(i, j, 1) - 1$ and deposit the slab to one of three downwind cells. When choosing the destination cell, we have a probability, p_{downwind}, that the slab moves with the wind and probabilities, p_{port} and $p_{\text{starboard}}$, for each adjacent destination. Here, *port* and *starboard* are, respectively, left and right, with respect to the direction of the the wind. We do not implement a probability function, when determining destination, but one can consider differences in elevation between the eroding cell and the possible destination cells as inputs to realize a more desirable destination cell. In the Avalanche section, we will explain why this decision is also negligible.

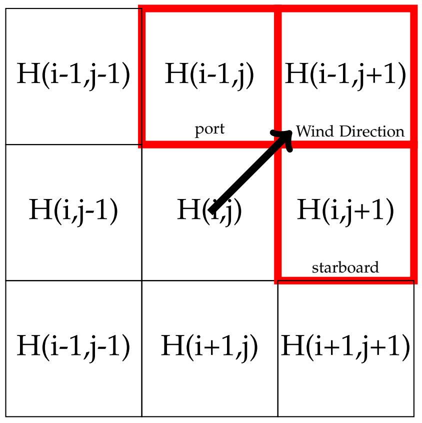

Figure 2.4: Neighborhood and potential destinations for an eroding slab

2.2.5 Dealing with Slabs on the Boundary

When dealing with slabs moving outside of the perimeter, we delete them from the matrix. This is expected to be an infrequent scenario, since the island data has a significant amount of negative elevations (water) serving as a "boundary" to the island. Other models have treated the matrix as a small torus, so slabs which exit from the top will be replaced at the bottom. These models have mostly considered large dune fields, like a desert. In our model, we have large negative elevations the further you get from the island. We do not treat our matrix like a torus since the island would run into the coastline before its western coast would collide with the eastern coast.

2.3 Avalanche

2.3.1 Main Goal

There is the possibility that unrealistically steep hills may be formed. The goal here is to simulate a natural gravitational smoothing process to ensure that each elevation cell $H(i, j, 1)$ has a realistic number of slabs compared to its neighbors.

2.3.2 Angle of Repose & Adjustment for Plant Presence

We define the *angle of repose*, denoted θ_0, to be the smallest angle for which avalanching occurs on bare sand. We define the angle between a cell and the downwind cell $\theta_{(i',l')}$ and if

$$\tan^{-1}\left(\frac{|H(i,j,1) - H(i',j',1)|}{L} \cdot \delta\right) = \theta_{(i',j')} \geqslant \theta_0, \tag{2.5}$$

then cell (i', j') is in violation of the angle of repose [9]. Note that (i', j') represent neighbors for (i, j), where the neighborhood is the von Neumann neighborhood of the four immediately adjacent cells. To account for the presence of plants, we implement a two-step process. First, we check if $H(i, j, 1)$ violates the angle of repose with one of its neighbors. If $H(i, j, 1)$ does violate the angle of repose, then it avalanches with probability

$$p_{\text{avalanche}} = \alpha \cdot \frac{\theta_{(i',j')}}{\theta_0} \cdot (1 - PC(i, j)),$$

where α is an avalanche parameter. For our model, we have set $\alpha = 1$. So, as $\theta_{(i',j')}$ increases, $p_{\text{avalanche}}$ increases and as $PC(i, j)$ increases, $p_{\text{avalanche}}$ decreases.

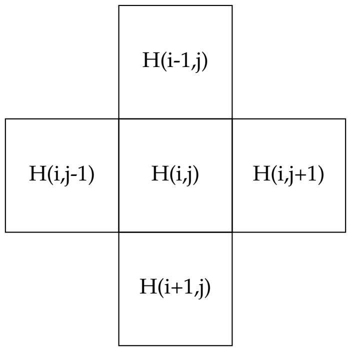

Figure 2.5: von Neumann neighborhood of cell (i, j)

2.3.3 Frequency of Avalanche and Determining Neighborhoods

The goal is that any eroded slab does not, again, create a steep angle with one of its neighbors. To ensure this, we invoke avalanching every time a slab is moved. Because of this, it is important to have a smaller neighborhood than the 8-cell neighborhood used in Aeolian Transport.

Suppose that cell (i, j) had too many slabs. Suppose that cell $(i-1, j)$ was in violation of the angle of repose with cell (i, j). Figure 2.6 illustrates the slab avalanching to the north. Suppose that cell $(i-1, j+1)$ violates the angle of repose with cell $(i-1, j)$ by also being too shallow. We iterate avalanche again and the slab will then avalance again to the east, illustrated in Figure 2.7.

We allow every cell violating the angle of repose to avalanche until a complete iteration returns no avalanches. This confirms that the topography of the island has been smoothed for each cell or that the plant density is significant enough to prevent further avalanching.

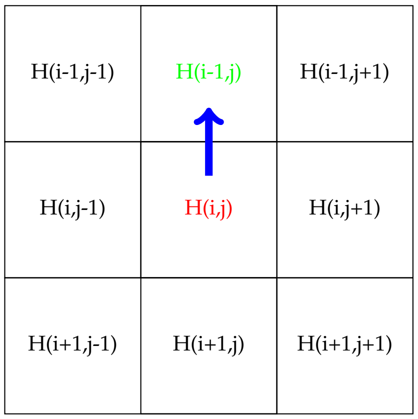

Figure 2.6: A slab avalanching to the north

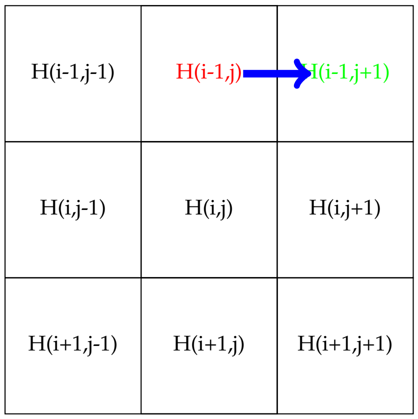

Figure 2.7: A slab avalanching to the east

2.4 Marine Processes

2.4.1 Main Goal

Erosion and deposition of sediment by tides is modeled once every 2 weeks to produce a smooth beach profile. We define $\lambda_{\text{beach}} = 2.5$ to be the maximum vertical elevation (with respect to mean sea level) that waves reach during high tide and $\lambda_{\text{underwater}} = 5m$ to be the horizontal distance that the equilibrium slope extends into the water. We let eastern grid locations with elevations between 0 meters and λ_{beach} meters be the beach and we smooth the profile for $\lambda_{\text{underwater}}$ meters into the water to account for both high and low tides [8]. We then calculate the angle of the beach profile and add or delete slabs under the beach profile to fit the linear curve. Cells which have slabs added to them can be considered as having sediment supplied to them by the sea, and cells which have slabs deleted from them can be considered as having undergone erosion through wave attack.

2.4.2 Identify beach

To identify the coastline, for each row i in our matrix H, we check each column of this row transect one at a time. We look for the largest value of j for which $H(i, j, 1)$ is non-negative. This location (i, j) represents one point on the island where the ocean meets the beach. We store the column entry j for this easternmost cell in a column array, $\text{ColumnArray}(i) = j$. If no such cell exists, we set $\text{ColumnArray}(i) = 0$. This tells us that the entire row of the matrix is underwater.

Then, for rows i with $\text{ColumnArray}(i) \neq 0$, we identify the easternmost cell (i, j) for which $\Big(H(i, j, 1) - H(i, \text{ColumnArray}(i), 1)\Big) \cdot \delta \geqslant \lambda_{\text{beach}}$ and set $\text{OppositeLocation}(i) = j$. This represents the location on the beach that the tallest wave during high tide can reach [8]. For each i, any cell (i, j) where j is such that $\text{OppositeLocation}(i) \leqslant j \leqslant \text{ColumnArray}(i)$, we say that (i, j) is in the beach by setting $H(i, j, 2) = 1$.

2.4.3 Create beach profile

Once we have declared the beach, we use the same two column arrays to produce a third column array ϕ. This represents the angle of the slope of the beach for each of the row transects. We use ϕ to determine the proper elevation for each cell (i, j) that is in the beach profile. For cells (i, j) with $\text{ColumnArray}(i) \neq 0$, we calculate $\phi(i)$:

$$\phi(i) = \tan^{-1}\left(\frac{H(i, \text{OppositeLocation}(i), 1)}{\text{AdjacentLength}(i)}\right), \tag{2.6}$$

where $\text{AdjacentLength}(i) = b_1 \cdot \big(\text{OppositeLocation}(i) - \text{ColumnArray}(i)\big)$. Once we have calculated ϕ for every row i in matrix H, we smooth our beach profile with a linear equation. If $\text{ColumnArray}(i) = 0$, we set $\phi(i) = 0$. For rows i with $H(i, j, 2) = 1$ and

$\phi(i) > 0$, we create our beach profile:

$$H'(i,j,1) = H(i, \text{OppositeLocation}(i), 1) - \frac{H(i, \text{OppositeLocation}(i), 1)}{\text{AdjacentLength}(i)} \cdot \big(\text{OppositeLocation}(i) - j\big), \tag{2.7}$$

where $H'(i,j,1)$ is the adjusted elevation at cell (i,j) for all j such that $\text{OppositeLocation}(i) \leqslant j \leqslant \text{ColumnArray}(i)$.

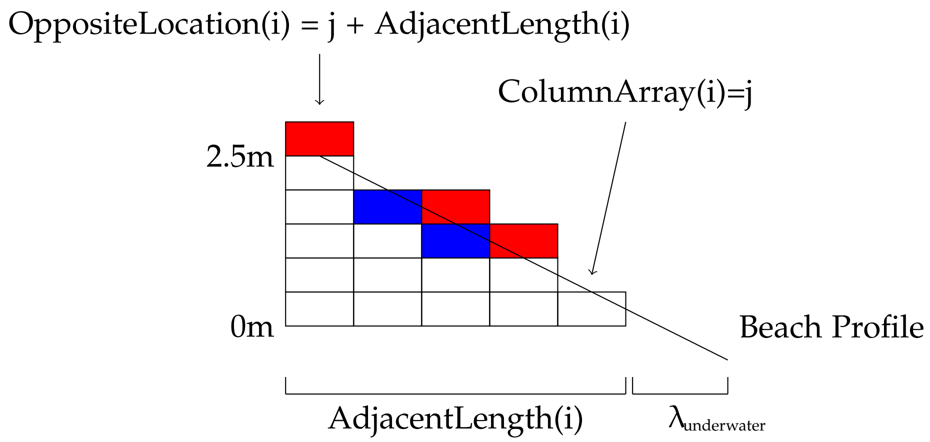

Figure 2.8: Black line = equilibrium profile. Red slabs would be deleted, blue slabs would be added.

In Figure 2.7, the black line represents the equilibrium profile. Red slabs represent slabs that would be deleted, if they exist. Blue slabs represent slabs that would be added, if the elevation is sufficiently below the equilibrium profile.

2.5 Marsh / Swamp Processes

2.5.1 Main Goal

When simulating marsh processes, we maintain a clearly defined marsh region where there is high plant density of only Spartina alterniflora. Our goal is to simulate a constant flattening process which expands the marsh. We use field data to determine realistic elevations for the marsh [13].

2.5.2 Identify marsh

Similar to marine processes, for each row i in our matrix H, we check every column j along the transect to identify the western coastline. We identify the smallest j for which $H(i, j, 1) \geqslant 0$ and set $\text{MarshColumnArray}_1(i) = j$. If no such column exists, we set $\text{MarshColumnArray}_1(i) = 0$. Again, this is represents a row that is entirely under water. For rows i with $\text{MarshColumnArray}_1(i) \neq 0$, we identify the smallest j for which $H(i, j, 1) \cdot \delta \geqslant \lambda_{\text{marsh}}$ and set $\text{MarshColumnArray}_2(i) = j$. Here, λ_{marsh} is the maximum elevation of cells in the marsh and this value comes from field data and $\lambda_{\text{marsh}} = 1\text{m}$ [13]. If $\text{MarshColumnArray}_1(i) = 0$, we set $\text{MarshColumnArray}_2(i) = 0$.

For rows i with $\text{MarshColumnArray}_1(i) \neq 0$ and $\text{MarshColumnArray}_2(i) \neq 0$, we define our marsh profile to be all cells (i, j) which lie between the two marsh column arrays. More rigorously, we set $H(i, j, 2) = 2$ for all j with $\text{MarshColumnArray}_1(i) \leqslant j \leqslant \text{MarshColumnArray}_2(i)$. If all columns j in row i have elevation below λ_{marsh} (that is, if $\text{MarshColumnArray}_1(i) \neq 0$ but $\text{MarshColumnArray}_2(i)$ is not defined) then we set $H(i, j, 2) = 2$ for every column j for which $H(i, j, 1) \cdot \delta \geqslant 0$. This means that we can have an entire row transect along the island that is *entirely* declared as the marsh.

2.5.3 Update Plants

Once the marsh has been declared, we check that only Spartina alterniflora is present in the marsh. Spartina alterniflora only survives in the marsh. As the marsh evolves over time, we ensure that there are no other grasses in the marsh and we ensure that there is no Spartina alterniflora anywhere else on the island. We check every cell (i, j) for which $H(i, j, 2) = 2$ and $P_k(i, j, 1) > 0$ for any k. In particular, if $P_k(i, j, 1) > 0$ for $k \neq 3$, we set

$P'_k(i, j, 1) = 0$, where P'_k replaces P_k in the next time step. Likewise, we check every cell (i, j) for which $H(i, j, 2) \neq 2$ and $P_3(i, j, 1) > 0$ and set $P'_3(i, j, 1) = 0$.

2.5.4 Simulate Marsh Growth

To simulate marsh growth, for each row i with $\text{MarshColumnArray}_1(i) \neq 0$, we identify the westernmost column j which is *not* in the marsh (i.e. $j = \text{MarshColumnArray}_2(i)+1$). With probability p_{marsh}, we allow the topmost slab on cell (i, j) to be deposited one cell to the west of the marsh. That is, cell $(i, \text{MarshColumnArray}_2(i) + 1)$ gives one of its slabs to cell $(i, \text{MarshColumnArray}_1(i) - 1)$. Since our sand slabs are units of elevation with thickness δ, it is impossible to flatten the marsh with thickness smaller than δ. Because of this, we can view the marsh as transporting slabs along a conveyor belt to the next available location. So our slabs "roll" along each transect and are deposited in the first column j containing negative elevation (i.e. $H(i, j, 1) < 0$).

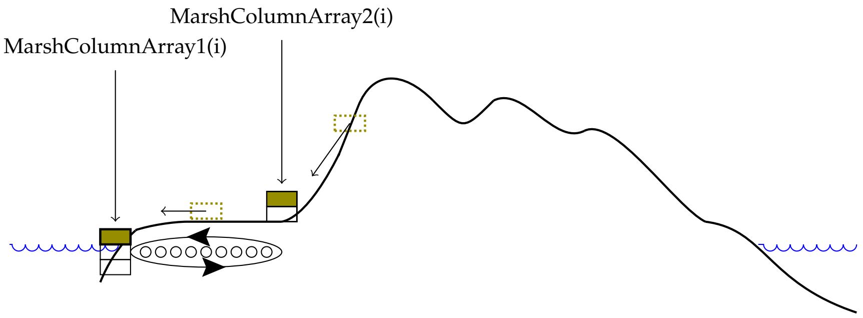

Figure 2.9: Marsh dispersing a single slab to simulate marsh growth

Chapter 3

Simulation Results

We simulated the evolution of a long thin island with 800x100 squared slab area under four different conditions. These conditions are: northeast wind with very high plant density, northeast wind with no plants, westward wind with very high plant density, and westward wind with no plants. Here, *westward* wind can be defined as wind which comes from the east and moves west across the island. Northeast wind is defined similarly. All four of our simulations have constant unidirectional wind with velocity ω_{max} and $\delta = 0.1$. Below in Figure 3.1 is the starting island used for each simulation where the colorbar represents number of slabs present at each grid point and the white contour represents grid points where the elevation is exactly zero:

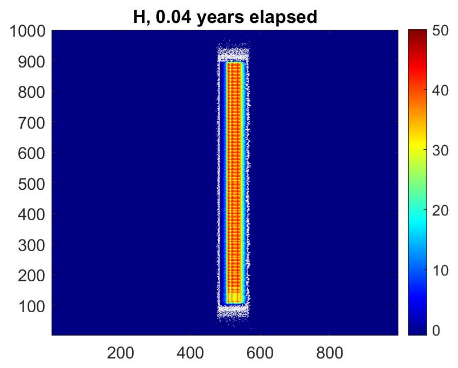

Figure 3.1: Initial island

Simulation 1:

This simulation represents the evolution of an island with constant wind, coming from the southwest and moving northeast across the island. This island is equipped with a very high plant density.

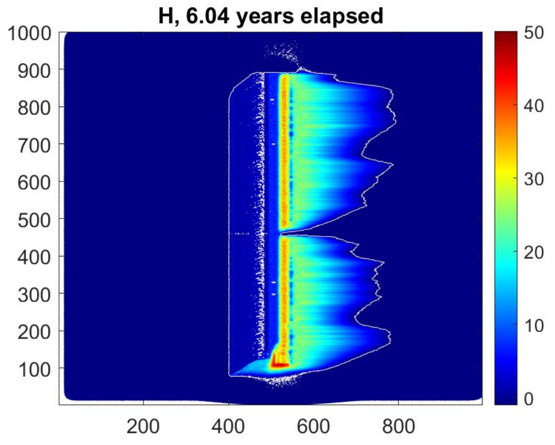

Figure 3.2: 6 years, northeast wind, high plants

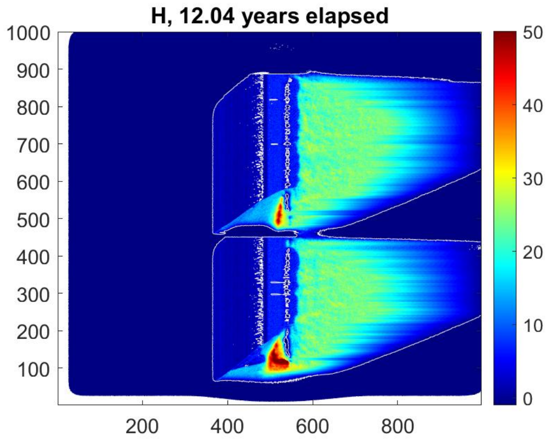

Figure 3.3: 12 years, northeast wind, high plants

With high plant density, we expect aeolian transport should have less effect on the island. Observing the north and south tips of the island, we see that there is very little northeastern movement. The most drastic growth on the island is to the east, with a northeast slope on the south side of each beach portion. We see that marine processes has a stronger influence on the development of this island than aeolian transport. Marine processes begins to develop the beach profile by adding slabs to the eastern side of the island. At around four years, marine processes actually severed the island in two. The dune field (cyan region in Figures 3.4 and 3.5) was pushed to the west, creating a thin cyan transect running east to west. At this point, the island has two separate marshes (maroon regions in Figure 3.4 north of the thin cyan line and south of it). Afterwards, the dune field and the coast (yellow region in Figures 3.4 and 3.5) moves northeast, with the wind. We end with two marshes in a parallelogram shape where the island remains connected by a thin strip of dunes. We will contrast this simulation with the next simulation 2. We will show that this island, in simulation 1, actually shows a considerably

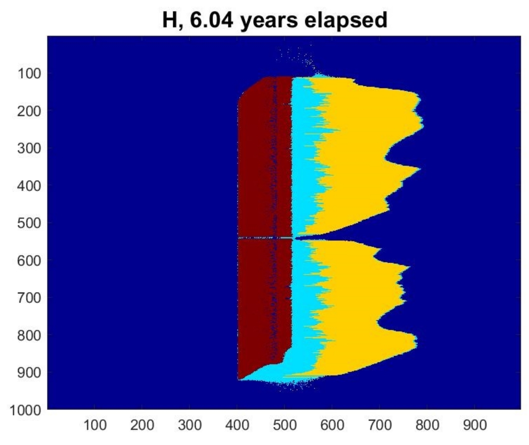

Figure 3.4: 6 year, northeast wind, high plants

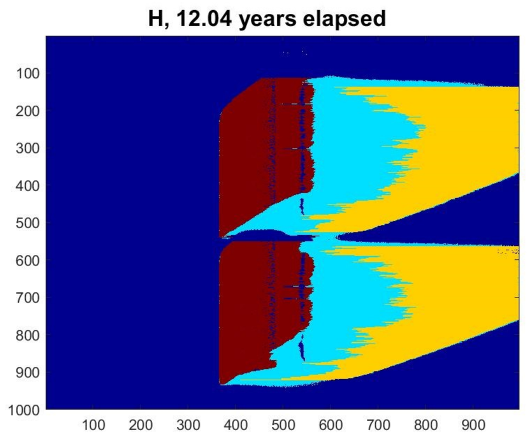

Figure 3.5: 12 years, northeast wind, high plants

Figure 3.4 & 3.5: Maroon = marsh, Cyan = dunes, Yellow = beach

more stable island than the simulation with no plants present.

Simulation 2:

This simulation represents the evolution of an island with constant wind, coming from the southwest and moving northeast across the island. This island has *no* plants present.

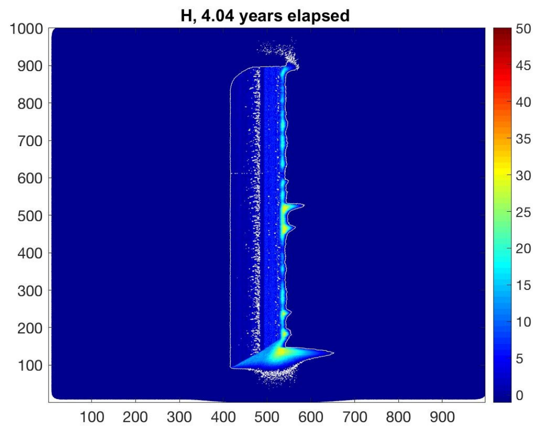

Figure 3.6: 2 years, northeast wind, no plants

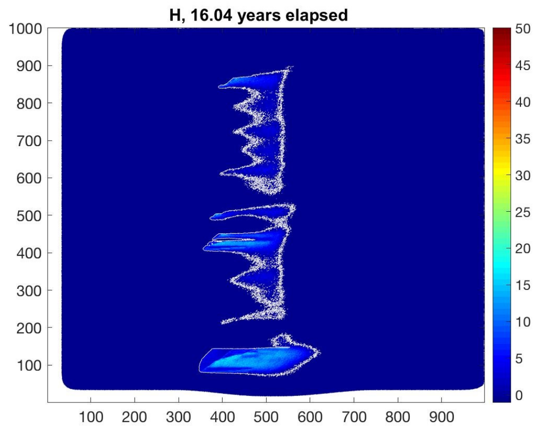

Figure 3.7: 12 years, northeast wind, no plants

In Figure 3.6, we see the progression of the island in only four years. After 16 years,

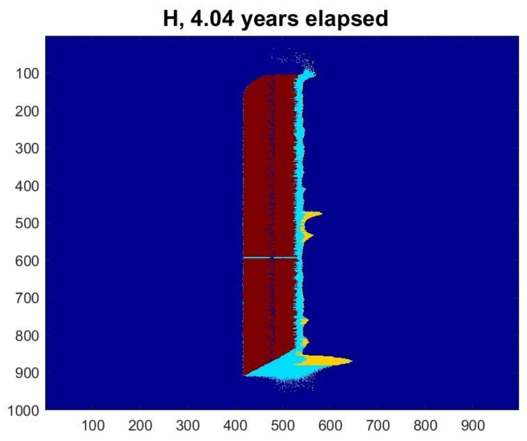

Figure 3.8: 2 years, northeast wind, no plants

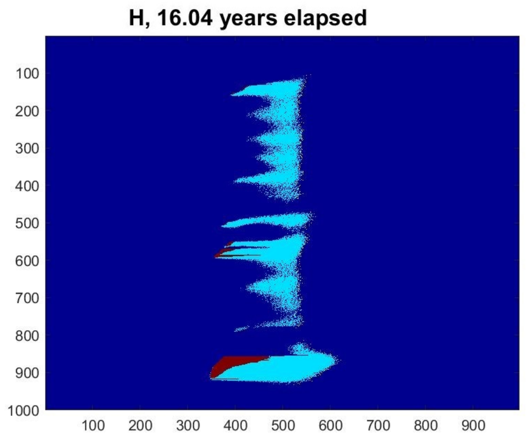

Figure 3.9: 12 years, northeast wind, no plants

Figure 3.8 & 3.9: Maroon = marsh, Cyan = dunes, Yellow = beach

in Figure 3.7, the island has nearly disintegrated. What is peculiar, between simulations 1 and 2, is that marine processes seems to play a drastically less significant role when plants are not present. The reason this occurs in the model is that with no plants present, the probability of avalanching is significantly higher. Throughout the simulation, the island is smoothing itself through avalanche and reducing the elevation of tall dunes. Marine processes is given an island topography that is less rigid and flatter and so there are fewer slabs to be added. This quick disintegration also shows how much more aeolian erosion occurs without plants on the island.

In Figure 3.8, we see that the vast majority of island is classified as a dune field, which is very unrealistic. This should be remedied by including the impact of high tide versus low tide in marsh processes. There should also be some ebb and flow from tidal forces helping to flatten and spread the marsh.

We find, from this simulation, that the island is less stable than the island in simulation 1. As the sediment moves away from the marsh, the marsh flattens itself until it is submerged. Meanwhile, the sediment is dispersed and settles underwater so we see the island disappear by the 100th year (Figure 3.10, above).

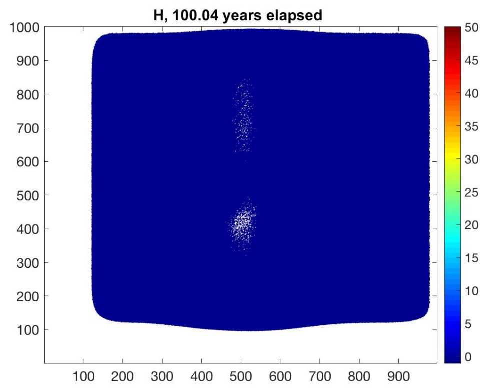

Figure 3.10: 100 years, northeast wind, no plants

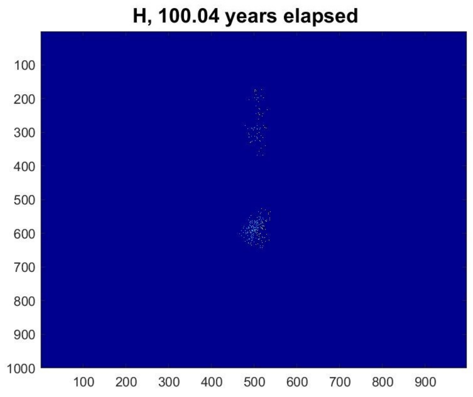

Figure 3.11: 100 years, northeast wind, no plants

Figure 3.10: Maroon = marsh, Cyan = dunes, Yellow = beach

Simulation 3:

This simulation represents the evolution of an island with constant wind, coming from the east and moving west across the island. This island is equipped with a very high plant density.

We again find some surprising results. After only 10 years (Figure 3.14) of constant westward wind, we see that the island separates from the beach and creates a lake in the

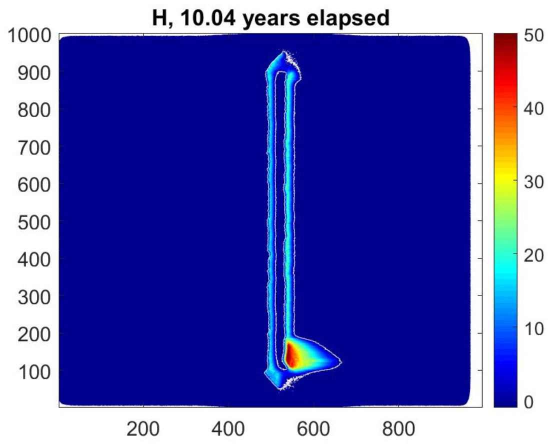

Figure 3.12: 10 years, west wind, high plants

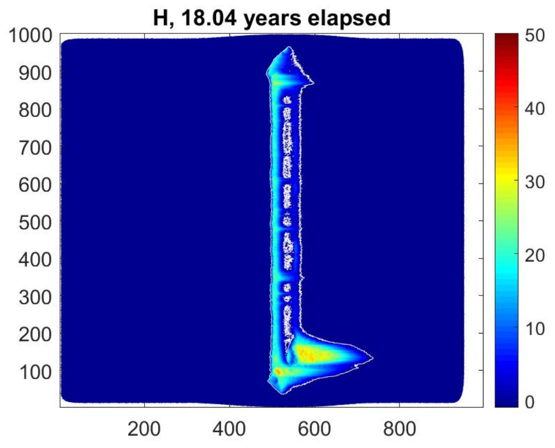

Figure 3.13: 18 years, west wind, high plants

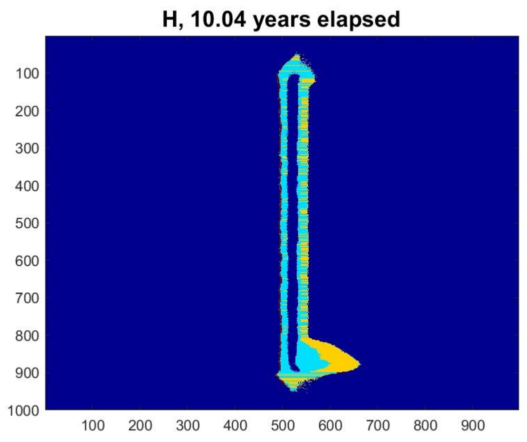

Figure 3.14: 10 years, west wind, high plants

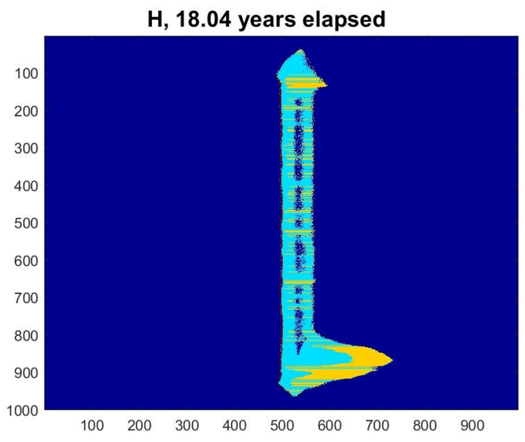

Figure 3.15: 18 years, west wind, high plants

Figure 3.14 & 15: Maroon = marsh, Cyan = dunes, Yellow = beach

middle. The high plant density prevents significant wind erosion, but with a constant westward wind, there is some westward motion on the island. This can be seen by the formation of the lake in the middle. Meanwhile, marine processes will apply a constant eastward ebb, pulling the coastline further east, widening the lake.

The marsh characteristically has higher plant density than the dunes. Since this island has very high plant density, this makes aeolian transport more difficult, but especially in the marsh. This causes the lake to begin to fill in 8 years later (Figure 3.15) of constant westward wind, since the dunes are more mobile than the marsh, which is west of the dunes. Afterwards, we see a trend similar to what occurred in Simulation 1, where the beach is dragged eastward.

Figure 3.16 shows an aggressive eastward trend, which is incredibly peculiar for constant westward wind. This is likely due to an overactive marine processes. With constant westward wind, we expect to see the dune region cover up the marsh. We do maintain a marsh and we do see expansion of the dunes. The dunes do force a much narrower marsh than in the first two simulations. We see an expansion of the beach and both expansions (of the dune and of the beach) occur in the wrong direction.

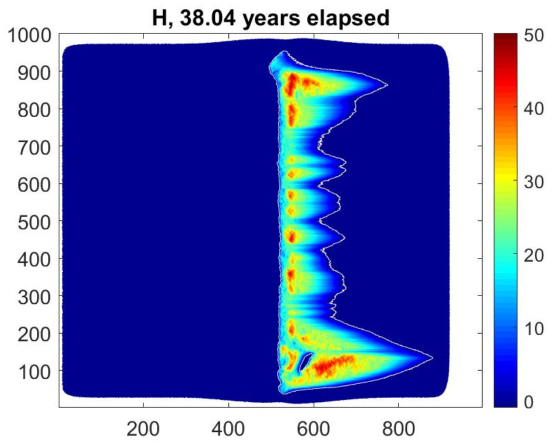

Figure 3.16: 38 years, west wind, high plants

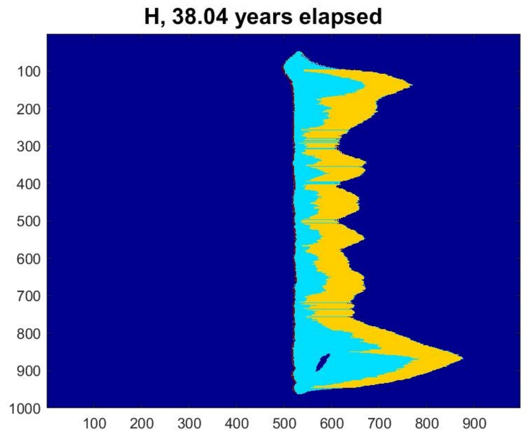

Figure 3.17: 38 years, west wind, high plants

Figure 3.17: Maroon = marsh, Cyan = dunes, Yellow = beach

Simulation 4:

This simulation represents the evolution of an island with constant wind, coming from the east and moving west across the island. This island is has *no* plants.

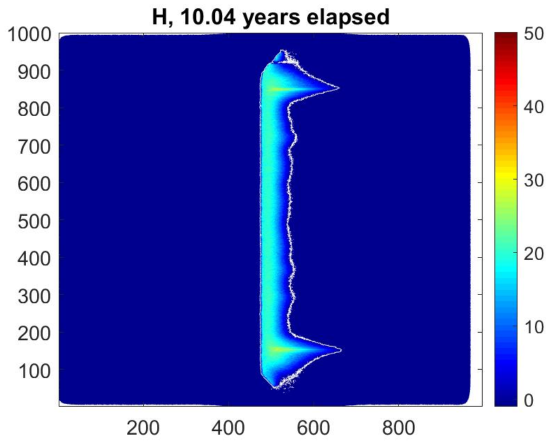

Figure 3.18: 10 years, west wind, no plants

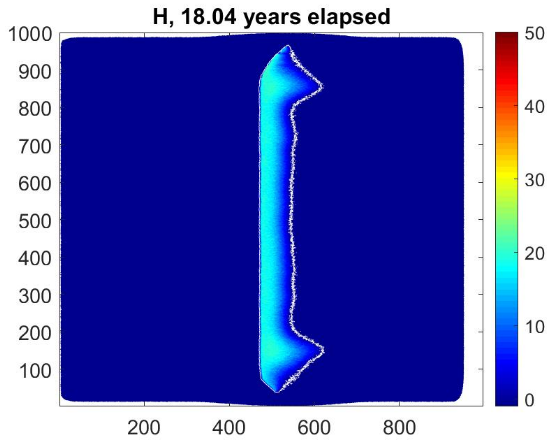

Figure 3.19: 18 years, west wind, no plants

Figure 3.18 shows the evolution under these conditions after 10 years and Figure 3.19 is 18 years. Similar to simulation 3, we see the north and south tips of the island being most mobile and forming a large C-shape with the island. Unlike simulation 3, the lack of plants allows significant mobility of the slabs in the dune fields, so there is no sep-

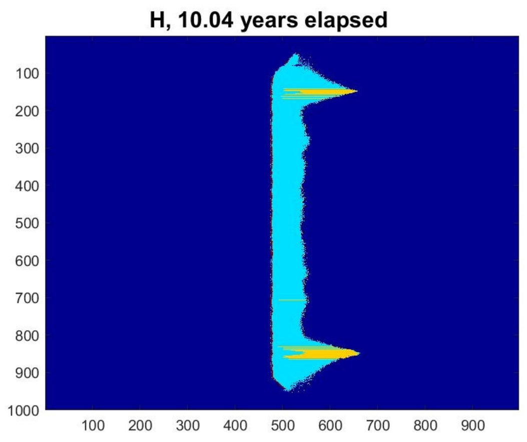

Figure 3.20: 10 years, west wind, no plants

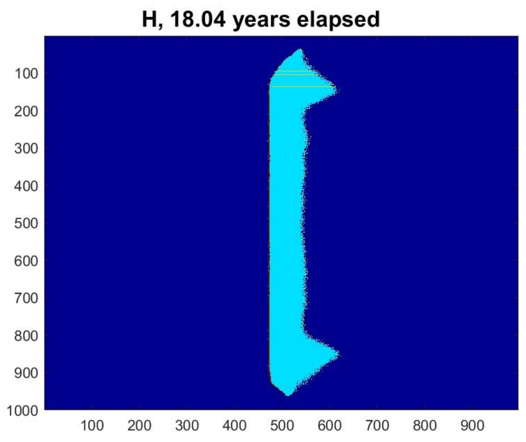

Figure 3.21: 18 years, west wind, no plants

Figure 3.20 & 21: Maroon = marsh, Cyan = dunes, Yellow = beach

aration and no lake forms; the island is very smooth. Figure 3.21, when compared to Figure 3.15 from simulation 3, shows how drastic this homogeneity is. There is only a dune field!

Figures 3.22 and 3.24 (next page) show the evolution of an island with no plants experiencing constant westward wind after 38 years. The northern and southern tips of these islands are significantly shorter and less pointy than the tips of the island shown in Figure 3.16. They also show a more uniform westward drift of the island when compared to Figure 3.16, although it is very gradual. This suggests that aeolian transport may not be as influential as we would like.

Figures 3.23 and 3.25 show the final resulting island after 100 years. The results show the beach retreats to the west, diminishing the C-shape that was most drastic in the first 10 years. We see a westward shift of the island but it is more gradual than expected. Compared to simulation 3, there is a modest difference in overall westward mobility. This, again, suggests that aeolian transport is not as effective as we would like, but also that the marsh is currently too stable. We expect to see a more significant westward drift over 62 years. Additionally, this island does not disintegrate, despite having no plants, like the island in simulation 2. The stability of the western marsh maintained the island's integrity under westward wind.

Lastly, Figures 3.22 and 3.23 demonstrate the need for shadow zones in aeolian transport. Giving slabs only the opportunity to erode to the next adjacent cell does not accurately represent the way that a fairly flat dune with no plants would erode by wind. Instead, we see an evolution of shape that is much too gradual. We also see a westward shift that is much too gradual.

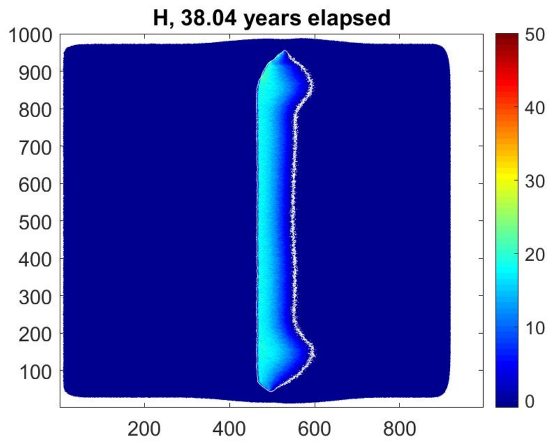

Figure 3.22: 38 years, west wind, no plants

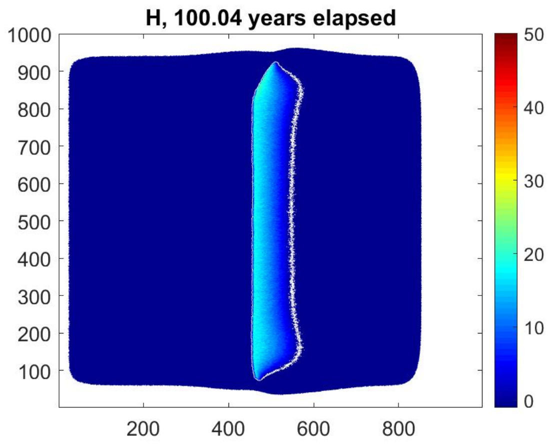

Figure 3.23: 100 years, west wind, no plants

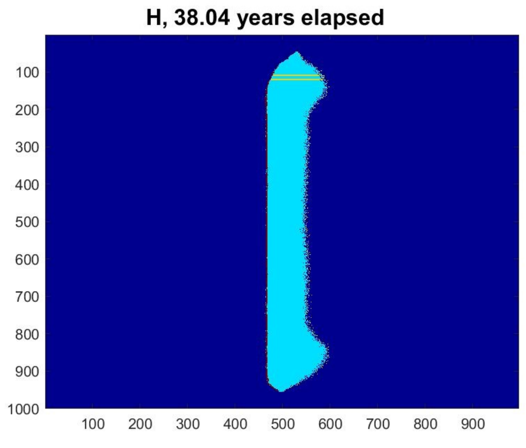

Figure 3.24: 38 years, west wind, no plants

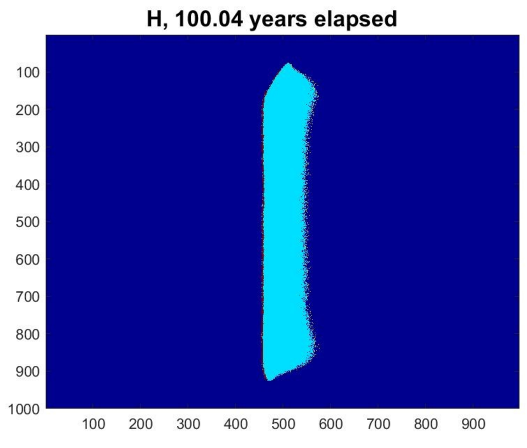

Figure 3.25: 100 years, west wind, no plants

Figure 3.24 & 25: Maroon = marsh, Cyan = dunes, Yellow = beach

Chapter 4

Conclusion

In constructing our model, we expanded upon many island models. Most of our processes mimic Nield and Baas DECAL algorithm [9] as well as Keijsers et. al DUBEVEG model [8]. In addition to their aeolian transport, marine processes, and plant erosion models, we incorporated Rastetter's plant propagation [11] across the entire island and we modeled the nearshore marsh with a unique process not contained in any literature. We were successful in updating the island's land categories throughout the simulation process. We also were able to implement plant species preferences to reflect preferences in elevation or location (i.e. marsh versus dune).

Some of our concerns with an overactive marine processes can be remedied by decreasing the frequency of the process. Having marine processes occur every two week timestep may be excessive. While $\lambda_{\text{beach}} = 2.5$ meters is the highest elevation above mean sea level that the wave runup reaches, it may be unrealistic to expect that amount of hydrodynamic energy at such a regular frequency. It should also be noted that some older models have included functions to calculate the vertical limit of wave runup from a calculated hydrodynamic energy function [8].

Mentioned in the results section, the inclusion of an overwash process may also balance out the work done by marine processes, although there are islands that evolve without experiencing overwash. The only simulated erosion by water is marine processes which seem to be dragging the coastline further to the east, based on the results. Including a process that moves slabs across the island, by water erosion, should drastically alter the evolution of the north and south tips of the islands. In turn, this should also change the evolution of the coastline. Nonetheless, further changes need to be made to marine processes to more accurately shape the tips of the island.

Additionally, the overwash process may also help to improve the marsh processes. The simulations with a constant east-to-west wind display a very small marsh that grows much more slowly than anticipated. One explanation for this is that the marsh processes do not simulate a fluctuation in sea level (tide changes) significantly enough. The depth of water immediately adjacent to the marsh does not change to a smooth slope through marsh processes. The influence of high and low tide would both flatten and widen the marsh but it would also keep the marsh's coastline relatively shallow. Without the influence of tides, as the marsh undergoes the flattening process, the slabs which are deposited to the west of the marsh are being placed in gradually deeper water. While the tidal impact should be the most immediate solution, another potential remedy for this phenomenon is the overwash process. By including erosion by water, across the island in a westward direction, we can expect more frequent deposition of slabs to the west of the marsh, in turn helping to create a more gradual slope between the edge of the marsh and its adjacent underwater cells.

Further concerns with the results are from the discovery that computation with this model is not very efficient. The advice of Nield and Baas should echo, "the inclusion of shadow zones is imperative for the development of recognizable dune forms through the interactions between moving slabs" [9]. The smoothing process used in our model relied entirely on avalanching slabs, which are moved one at a time. We chose to omit shadow zones in our model because the smoothing process of the island detailed in DECAL [9] and DUBEVEG [8] all include an angle of repose for avalanching. It seemed more efficient to split wind erosion and smoothing into different categories for the sake of efficiency. Our cellular model uses fairly simple calculations and should be efficient. Unfortunately, avalanching slabs proved to occupy the vast majority of computing time and is likely the cause for inefficiency. Instead, correcting an eroded slab in the aeolian transport procedure is likely a more efficient method.

An added benefit of the shadow zones should be that wind erosion becomes more impactful on the island. In simulation 4, in particular, the evolution over more than 60 years between Figures 3.22 and 3.23 is overwhelmingly too insignificant for an island with no plants and constant ω_{max} windspeed. Allowing for slabs to erode to more distant neighbors would allow for more energetic wind erosion.

In addition to overwash, we also plan to introduce a storm process to the model. The barrier islands are small and often one storm is enough to alter the island's geomorphology. The inclusion of storms will make the model more difficult to validate.

Our model is able to simulate the evolution of an entire barrier island. We have results that show the existence and evolution of dunes. We have shown the existence and growth of a marsh. We have marine processes that maintains a smooth beach profile. We have simulated favorable growing conditions for four species of plants. Most importantly, we have shown that the evolution of a barrier island is greatly impacted by the presence of plants on the island. These facets makes our model novel.

Bibliography

[1] Bruun, P., 1962. *Sea level rise as a cause of shore erosion*. Journal of Waterways and Harbors Division, ASCE 88, 117-130.

[2] Davidson-Arnott, R. G. D., 2005. *Conceptual Model of the Effects of Sea Level Rise on Sandy Coasts*. Journal of Coastal Research, 21(6), 1166–1172.

[3] Dean, R. G., 1991. *Equilibrium Beach Profiles: Characteristics and Applications*. Journal of Coastal Research, 7: 53-84.

[4] Delgado-Fernandez, I., Davidson-Arnott R. G. D., 2011. *Meso-scale aeolian sediment input to coastal dunes: The nature of aeolian transport events*. Geomorphology, 126: 217-232.

[5] Durán, O. and Moore, L., 2014 *Barrier island bistability induced by biophysical interactions*. Natural Climate Change, 5.

[6] Fiedler, J. W., Smit, P. B., Brodie, K. L., McNinch, J., Guza, R. T., 2018. *Numerical modeling of wave runup on steep and mildly sloping natural beaches*. Coastal Engineering, 131: 106-113.

[7] Hayden, B. P., Hayden, N. R., 2003. *Decadal and Century-Long Changes in Storminess at Long-Term Ecological Research Sites*. Climate Variability and Ecosystem Response at Long-Term Ecological Research Sites: 262-285.

[8] Keijsers, J. G. S., et al., 2016. *Modeling the Biogeomorphic Evolution of Coastal Dunes in Response to Climate Change*. Journal of Geophysical Research, 122: 1161-1181.

[9] Nield, J. M., and Baas, A. C. W., 2008. *Investigating Parabolic and Nebkha Dune Formation Using a Cellular Automaton Modelling Approach*. Earth Surface Processes and Landforms, 33: 724-740.

[10] Porter, J., D. Krovetz, W. Nuttle and J. Spitler. 2019. *Hourly Meteorological Data for the Virginia Coast Reserve LTER 1989-present*. Virginia Coast Reserve Long-Term Ecological Research Project Data Publication knb-lter-vcr.25.40.

[11] Rastetter, E.B., Turner, M. G. and Gardner, R. H., 1991. *Quantitative Methods in Landscape Ecology: The Analysis and Interpretation of Landscape Heterogeneity*. Ecological Studies, 82: 353-378.

[12] Sinclair, M. N., Woods, N. N., and Zinnert, J. C., 2020. *Seasonal facilitative and competitive trade-offs between shrub seedlings and coastal grasses*. Ecosphere 11(1):e02995.

[13] USACE-TEC and JALBTCX. 2014. *High resolution LiDAR Data for Hog Island, VA, 2013*. Virginia Coast Reserve Long-Term Ecological Research Project Data Publication knb-lter-vcr.228.9.

[14] Werner, B.T., 1995. *Eolian dunes: Computer simulations and attractor interpretation*. Geology; 23(12): 1107-1110.

[15] Zinnert, J.C., Via, S.M., Nettleton, B.P., Tuley, P.A., Moore, L.J., and J.A. Stallins, 2019. *Connectivity in coastal systems: barrier island vegetation influences upland migration in a changing climate*. Global Change Biology 25: 2419-2430.

Appendices

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%  VARIABLE DEFINITION  %%%%%
                                                                                 %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
                                                                                 %

                                                                                 %
time=2601;        %time               NUMBER OF 2-week INTERVALS

                                  %
delta=0.1;    %slab thickness      1/100 of meter thick


%Windspeed=5;     %wind velocity
%%% Should include pseudo-random selection of "categories" of windspeed,
%%% similar to WindDir, below.

Windmin=6;  %minimum velocity needed to move a slab of sand
                                                                                 %
StormThreshold=16;  %winds above this value will be classified as "HWE's"
                                                                                 %

                                        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%WindDir=7;  %wind direction    %  1=N  2=NE  3=E  4=SE  5=S  6=SW  7=W  8=NW
 %                                                                          %
                                        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%




L=1;        %hop length
                                                                                 %
Pe=1;       %probability of erosion
                                                                                 %
Pd=1;       %probability of deposition
                                                                                 %
n=1;        %number of aeolian iterations per year
                                                                                 %
numslabg=0;      %my variable
                                                                                 %
flag=0;
q=0;

MarshAreaArray=zeros(time,1);   MarshVolumeSum=zeros(time,1);
```

```
BeachAreaArray=zeros(time,1);   BeachVolumeSum=zeros(time,1);
DuneAreaArray=zeros(time,1);    DuneVolumeSum=zeros(time,1);

MarshVolumeArray=zeros(time,1);
BeachVolumeArray=zeros(time,1);
DuneVolumeArray=zeros(time,1);

%

%ELEVATION MATRIX

     %
    fname='FinalLongSkinnySIMULATIONisland';

              %
    ext='.txt';
                                                                             %
    H=load([fname,ext]);
                                                                             %
    snum='FinalLongSkinnySIMULATIONisland';

              %
    lines = dataread('file', 'FinalLongSkinnySIMULATIONisland.txt', '%s', 'delimiter',
'\n');
%     load('Hquarantine2.mat');
%     H=Hquarantine2;
%     evalin( 'base', 'clear Hquarantine2' )
        ROW=size(H,1);
        COLUMN=size(H,2);


%water table
                                                                         %
    W=zeros(ROW,COLUMN);

       %
%salinity
                                                                         %
    S=zeros(ROW,COLUMN);

%GRASS #1 (Ammophila)(burial resistant)
                                                                         %
    P1=zeros(ROW,COLUMN,2);

         %
    P1Burial=2;         %meters of burial until death
                                                                              %
    P1ErosionCoefficient=(2/3);     %used in formula   d=c1P1+c2P2+c3P3+c4P4

   %
    %%fname='Ammophila';

%
    %%ext='.txt';

%
    %%P1=load([fname,ext]);

%
    %%snum='Ammophila';

%
    %%lines = dataread('file', 'Ammophila.txt', '%s', 'delimiter', '\n');
%     load('P1.mat');
%     P1=P1proper;
%     evalin( 'base', 'clear P1proper' )

%GRASS #2  (Spartina)
    P2=zeros(ROW,COLUMN,2);

         %
    P2Burial=0.5;         %meters of burial until death
                                                                              %
    P2ErosionCoefficient=(1/3);
    %%fname='Spartina';

%
    %%ext='.txt';

%
    %%P2=load([fname,ext]);

%
    %%snum='Spartina';

%
    %%lines = dataread('file', 'Spartina.txt', '%s', 'delimiter', '\n');
%     load('P2.mat');
%     P2=P2proper;
%     evalin( 'base', 'clear P2proper' )
```

```
%SHRUB #1  (Morella)(bird-dispersed seeds)
    P3=zeros(ROW,COLUMN,2);

         %
    P3Burial=1.5;         %meters of burial until death
                                                                                %
    P3ErosionCoefficient=1;
    %%fname='Morella';

%
    %%ext='.txt';

%
    %%P3=load([fname,ext]);

%
    %%snum='Morella';

%
    %%lines = dataread('file', 'Morella.txt', '%s', 'delimiter', '\n');
%     load('P3.mat');
%     P3=P3proper;
%     evalin( 'base', 'clear P3proper' )
%
                                                                             %
%SECOND TYPE OF SPARTINA!!!
     P4=zeros(ROW,COLUMN,2);

          %
     P4Burial=0.5;

%
     P4ErosionCoefficient=(1/3);
%     load('P4.mat');
%     P4=P4proper1;
%     evalin( 'base', 'clear P4proper1' )
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Number of slabs that can possibly be moved on the entire grid as well as the
highest and lowest point on the grid                                     %
% for i=1:size(H,1) %for each row
                                                                          %
%     for j=size(H,2):-1:1 %for each column
                                                                          %
%         if (H(i,j)-W(i,j))>0            %ensuring we only consider positive
values                                                                    %
%             numslabg=numslabg+floor((((delta*H(i,j))-W(i,j))/delta));    %total
number of moveable slabs on the grid                                    %
%         end
                                                                          %
%     end
                                                                          %
% end
                                                                          %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Hstar serves as the "temporary" island replica as we move slabs
                                                                        %
%during each timestep.
                                                                        %
%Hstar is an important variable because, in this submodel, we conside every entry
in the elevation matrix H during this timestep and we                %
%determine whether one or more slabs will be moved from each cell, or grid location,
in H.                                                               %
%If we were to move cells from H instead of Hstar in this submodel, we would be
reconsidering the same slab several times during each timestep.        %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
H(:,:,2)=zeros(ROW,COLUMN);
Hstar=H;




M=max(max(H(:,:,1)));
%M=(4/3)*M;
m=min(min(H(:,:,1)));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% INITIAL PLANT CONDITIONS %
```

```
                                                                          %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%
                                                                          %
% Each plant species has a preferred "altitude" that it grows best in
                                                                          %
% (min and max among all species is 0.75m and 5m, respectively).
                                                                          %
% We set each cell of "appropriate" elevation to be covered by 30% of the
                                                                          %
% respective plant species. For cells which are submerged under water, we
                                                                          %
% set the proportion of plant coverage to -1.
                                                                          %
%
                                                                          %
% We have deleted the initialization of plant population since we are using
                                                                          %
% real data.
                                                                          %
%It is still important to ensure that the 2nd layer of each
                                                                          %
% plant matrix is initiated.
                                                                          %

                                                                          %
% for i=1:size(H,1)
                                                                            %
%     for j=1:size(H,2)
                                                                            %
%         if (delta*H(i,j,1))>=1 && (delta*H(i,j,1))<=5

 %
%             P1(i,j,1)=0.01;
%             P1(i,j,2)=H(i,j,1);
%         elseif (H(i,j,1)-W(i,j))<0
                                                                              %
%             P1(i,j,1)=-1;
%             P1(i,j,2)=0;%
%         end
                                                                            %
%
                                                                            %
%         if (delta*H(i,j,1))>=0.75 && (delta*H(i,j,1))<=3

 %
```

```
%             P2(i,j,1)=0.01;
%             P2(i,j,2)=H(i,j,1);%
%         elseif (H(i,j,1)-W(i,j))<0
                                                                              %
%             P2(i,j,1)=-1;
%             P2(i,j,2)=0;%
%         end
                                                                            %
%
                                                                            %
%         if (delta*H(i,j,1))>=1.5 && (delta*H(i,j,1))<=2.5

 %
%             P3(i,j,1)=0.01;
%             P3(i,j,2)=H(i,j,1);%
%         elseif (H(i,j,1)-W(i,j))<0
                                                                              %
%             P3(i,j,1)=-1;
%             P3(i,j,2)=0;%
%         end
                                                                            %
%
                                                                            %
%         if (delta*H(i,j,1))>=-0.5 && (delta*H(i,j,1))<=1

 %
%             P4(i,j,1)=0.01;
%             P4(i,j,2)=H(i,j,1);%
%         elseif (H(i,j,1)-W(i,j))<-0.5
                                                                              %
%             P4(i,j,1)=-1;
%             P4(i,j,2)=0;%
%         end
                                                                            %
%     end
                                                                            %
% end
                                                                            %
% [P1,P2,P3,P4]=PlantPropagation(H,P1,P2,P3,P4,W,S,delta,P1Burial,P2Burial,P3Burial,
P4Burial);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%%%%%%%%%%%%%%%%%%%%%%%%     %%% Would like to create "booklet" or
%BEFORE IMAGE%%%%%%%%%%%     %%% "slideshow" for all images.
figure                 %     %%% SHOULD title each figure!!!
colormap(jet())        %
clims=[-1 max(max(H(:,:,1)))];           %
imagesc(H(:,:,1),clims)%
colorbar               %
%caxis([m,M])           %
%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Before 3-D plot %%%%%%%%%%%%%%%%
%figure                         %
%b=bar3(H);                     %
%colormap(jet())                %
%colorbar                       %
%for k = 1:length(b)            %
%    zdata = b(k).ZData;        %
%    b(k).CData = zdata;        %
%    b(k).FaceColor = 'interp';  %
%end                            %
%caxis([m,M])                   %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
tic % MAIN LOOP IN CODE %
                                                                               %
for t=1:time
                                                                           %


    Ptot=P1(:,:,1)+P2(:,:,1)+P3(:,:,1)+P4(:,:,1);
    TimeStep=t

                                                                           %
    %Below is a matrix which carries a weighted "percent coverage" of the
                                                                           %
    %plant species where the "ErosionCoefficients" help to determine the
                                                                           %
    %"root strength" of each species, or their tendency to PREVENT erosion.
                                                                           %
```

```
                                                                              %
    PC=(P1ErosionCoefficient.*P1(:,:,1))+(P2ErosionCoefficient.*P2(:,:,1))+
(P3ErosionCoefficient.*P3(:,:,1))+(P4ErosionCoefficient.*P4(:,:,1));
                                      %

                                                                              %
    %(1) Run Swamp
                                                                              %
    [Hstar,ColumnArraySwamp1,ColumnArraySwamp2,AdjascentLengthSwamp,
OppositeLocationSwamp]=SwampProcesses(Hstar,delta,L,flag,PC);
     %
    H=Hstar;
                                                                              %

                                                                              %
    %(2) Aeolian Transport
                                                                              %
    %WindDir=randi([1,8]);
    for day=1:14
        Windspeed=16;
        WindDir=7;
%         WindSpeedParam=randi(2183);
%         if WindSpeedParam>=1 && WindSpeedParam<=2116
%             Windspeed=0;
%         elseif WindSpeedParam>2116 && WindSpeedParam<=2183
%             Windspeed=6;
%         elseif WindSpeedParam>2183
%             Windspeed=16;
%         end
%
%         WindParam=randi(2107);
%         if WindParam>=1 && WindParam<=148
%             WindDir=1;
%         elseif WindParam>=149 && WindParam<=401
%             WindDir=8;
%         elseif WindParam>=402 && WindParam<=774
%             WindDir=7;
%         elseif WindParam>=775 && WindParam<=1030
%             WindDir=6;
%         elseif WindParam>=1031 && WindParam<=1210
%             WindDir=5;
%         elseif WindParam>=1211 && WindParam<=1615
%             WindDir=4;
%         elseif WindParam>=1616 && WindParam<=1865
%             WindDir=3;
%           elseif WindParam>=1866 && WindParam<=2107
%               WindDir=2;
%           end
%
%
        if Windspeed>=6
            [H,Hstar]=AeolianTransport(Hstar,delta,L,H,W,Windspeed,Windmin,
StormThreshold,WindDir,n,Pe,Pd,PC);
      %
        end
    end


    %(3) Avalanche
                                                                          %
    [Hstar,flag]=AvalancheUPDATEflagonly(Hstar,delta,L,flag,PC);

   %

                                                                          %
    %(4) A lot of avalanching (if necessary)
                                                                          %
    while flag==1
                                                                          %
        [Hstar,flag]=AvalancheUPDATEflagonly(Hstar,delta,L,flag,PC);

   %
    end
                                                                          %


    % Should move Plant Propogation to BEFORE Marine Processes to ensure
    % Morella does not appear on east side of foredune (as is below).


    %(5) Plant Propogation
                                                                          %
%     if (1==mod(t,26))  %  choose 26 for 2-week timesteps;    52 for 1-week timesteps
                                                                    %
%         [P1,P2,P3,P4]=PlantPropagation(H,P1,P2,P3,P4,W,S,delta,P1Burial,P2Burial,
P3Burial,P4Burial);                                                    %
                                                                           %
%     end
```

```
    %(6) Marine Processes
                                                                               %
    [Hstar,PlantColumnArray,PlantColumnArray2,P3,flag]=MarineProcessesSLOPE(Hstar,delta,
L,flag,PC,P3);                                                             %
    H=Hstar;

    %Redeclare PC since P3 has been updated
    PC=(P1ErosionCoefficient*P1(:,:,1))+(P2ErosionCoefficient*P2(:,:,1))+
(P3ErosionCoefficient*P3(:,:,1))+(P4ErosionCoefficient*P4(:,:,1));
                                   %

    %(6.5) Avalanche after Marine Processes
    while flag==1
                                                                               %
        [Hstar,flag]=AvalancheUPDATE(Hstar,delta,L,flag,PC);
                                                                               %
    end

    %(7) Redeclare all categories in H
    [Hstar,P1,P2,P3,P4]=LandCategorization(Hstar,P1,P2,P3,P4,PlantColumnArray,
PlantColumnArray2,ColumnArraySwamp1,ColumnArraySwamp2);

    H=Hstar;
    [MarshAreaArray,BeachAreaArray,DuneAreaArray,MarshVolumeSum,BeachVolumeSum,
DuneVolumeSum,MarshVolumeArray,BeachVolumeArray,DuneVolumeArray]=
LandCategoryCounter(H,t,L,MarshAreaArray,BeachAreaArray,DuneAreaArray,
MarshVolumeSum,BeachVolumeSum,DuneVolumeSum,MarshVolumeArray,BeachVolumeArray,
DuneVolumeArray);



if (1==mod(t,52))
    q=q+1;
%%%%%%%%%%%%%%%%%%%%%
                                                                                 %
%"During" IMAGE%%%%%%
                                                                                  %
% figure              %

%
% colormap(jet())     %
% clims=[-1 max(max(H(:,:,1)))];%
% %hold on;
% %contour(H(:,:,1),[-1 -1],'Color','red');
% imagesc(H(:,:,1),clims);   %

       %
% filenamestring=strcat(['image',int2str(q),'.jpg']);
% saveas(gcf,filenamestring);
% %imwrite(filenamestring,'IslandMovie')
% %caxis([m,M])        %

 %

figfontsize=14;
figure
colormap(jet())
contourf(1:997,1:1000,flipud(H(:,:,1)),[-100:1:50],'Linestyle','none');
caxis([-1 50]);
hold on;
contour(1:997,1:1000,flipud(H(:,:,1)),[-1,-1],'Color','white');
% contourf(300:700,1:1000,flipud(H(:,300:700,1)),[-100:1:50],'Linestyle','none');
% caxis([-1 50]);
% hold on;
% contour(300:700,1:1000,flipud(H(:,300:700,1)),[-1,-1],'Color','white');
% set(gcf,'Position',[100,100,300,500]);
colorbar
title(strcat(['H, ',num2str(round(100*(t/26))/100),' years elapsed']));
set(gca,'Fontsize',figfontsize);
filenamestring=strcat(['imageCP',int2str(q),'.jpg']);
saveas(gcf,filenamestring);

matnamestring=strcat(['matrix',int2str(q)]);
save(strcat([matnamestring,'.mat']),'H');


%%%%%%%%%%%%%%%%%%%%
                                                                                  %
end
                                                                               %




end
toc %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
x=1:length(MarshAreaArray);
q=q+1;
plot(x,MarshAreaArray)
filenamestring=strcat(['image',int2str(q),'.jpg']);
saveas(gcf,filenamestring);

q=q+1;
plot(x,MarshVolumeArray)
filenamestring=strcat(['image',int2str(q),'.jpg']);
saveas(gcf,filenamestring);

q=q+1;
plot(x,DuneAreaArray)
filenamestring=strcat(['image',int2str(q),'.jpg']);
saveas(gcf,filenamestring);

q=q+1;
plot(x,DuneVolumeArray)
filenamestring=strcat(['image',int2str(q),'.jpg']);
saveas(gcf,filenamestring);

q=q+1;
plot(x,BeachAreaArray)
filenamestring=strcat(['image',int2str(q),'.jpg']);
saveas(gcf,filenamestring);

q=q+1;
plot(x,BeachVolumeArray)
filenamestring=strcat(['image',int2str(q),'.jpg']);
saveas(gcf,filenamestring);
```

```
function[P1,P2,P3,P4]=PlantPropagation(H,P1,P2,P3,P4,W,S,delta,P1Burial,P2Burial,
P3Burial,P4Burial)
%File PlantPropagation.m
%   This subroutine will first record the intial locations of each plant
%   species, so that we can keep track of individual plant burial. Next, it
%   will look at a number of different factors to determine, at each cell
%   location:
%       (a) will the plant die (from elevation or from competition)?
%       (b) will the plant (specifically shrubs) grow taller?
%       (c) will the plant propagate and how far will its seeds be
%           dispersed?
%
%   It is important to know that the shrubs (morella, P3) are more
%   competitive as a species than the grasses. We can determine if a plant
%   will die from competition based on the available "room to grow" and
%   also based on the "amount of growth" experienced by the shrubs, if the
%   cell is 100% covered. Each matrix Px(i,j) = [0,1] where P1(i,j) = 0 if
%   there is 0% coverage of plant 1 at location (i,j) and P2(i,j) = 1 if
%   there is a 100% coverage of plant 2 at location (i,j).
%
%   To run this file you will need to specify:
%       H     - elevation matrix that is continually used in main code
%       P1    - the plant matrix for Ammophila (GRASS)
%       P2    - the plant matrix for Spartina  (GRASS)
%       P3    - the plant matrix for Morella   (SHRUB)
%       W     - the elevation matrix for the water table
%       S     - the matrix which determines available salinity at each cell
%     delta   - the thickness of each slab
%
%   The routine will return the matrix for each of the plant species after
%   propagating.

fprintf('I am running PP! ')

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% CREATING 2nd LAYER TO RECORD ORIGINAL LOCATIONS OF PLANTS %             %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%              %
for i=1:size(H,1)                                                         %
    for j=1:size(H,2)                                                     %
        if P1(i,j,1)~=0                                                   %
            P1(i,j,2)=H(i,j);                                             %
        end                                                               %
                                                                          %
        %if P2(i,j,1)~=0                                                        %
        %    P2(i,j,2)=H(i,j);                                                  %
        %end                                                                    %
                                                                               %
        if P3(i,j,1)~=0                                                        %
            P3(i,j,2)=H(i,j);                                                  %
        end                                                                    %
                                                                               %
        %if P4(i,j,1)~=0                                                        %
        %    P4(i,j,2)=H(i,j);                                                  %
        %end                                                                    %
    end                                                                        %
end                                                                            %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% RESETTING NEGATED CELLS FOR CELLS WHICH WERE PREVIOUSLY SUBMERGED %     %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%     %
for i=1:size(H,1)                                                              %
    for j=1:size(H,2)                                                          %
        if P1(i,j,1)==-1 && (H(i,j)-W(i,j))>0                                  %
            P1(i,j,1)=0;                                                       %
        end                                                                    %
                                                                               %
        %if P2(i,j,1)==-1 && (H(i,j)-W(i,j))>0                                  %
        %    P2(i,j,1)=0;                                                       %
        %end                                                                    %
                                                                               %
        if P3(i,j,1)==-1 && (H(i,j)-W(i,j))>0                                  %
            P3(i,j,1)=0;                                                       %
        end                                                                    %
                                                                               %
        %if H(i,j,2)~=2%P4(i,j,1)==-1 && (H(i,j)-W(i,j))>=-0.5                  %
        %    P4(i,j,1)=0;    P4(i,j,2)=-1;                                      %
        %end                                                                    %
                                                                               %
        if (H(i,j)-W(i,j))<=0                                                  %
            P1(i,j,1)=-1;   P1(i,j,2)=-1;                                      %
            %P2(i,j,1)=-1;   P2(i,j,2)=-1;                                      %
            P3(i,j,1)=-1;   P3(i,j,2)=-1;                                      %
        end                                                                    %
                                                                               %
        %if (H(i,j)-W(i,j))<0.5                                                 %
        %    P4(i,j,1)=-1;   P4(i,j,2)=-1;                                      %
        %end                                                                    %
    end                                                                   %
end                                                                       %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% PLANT DEATH %                                                           %
%%%%%%%%%%%%%%%                                                           %
%                                                                         %
%            HAS BEEN MOVED TO THE BOTTOM OF THIS SUBROUTINE              %
%            %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%              %
%                                                                         %
%                      DEATH BY BURIAL                                    %
% currently, the burial is ONLY comparing the elevation of sand to the    %
% elevation of where plants first spawned. Eventually, we hope to have    %
% burial compare present year to previous year.                           %
%                                                                         %
%                                                                         %
% for i=1:size(H,1)                                                        %
%     for j=1:size(H,2)                                                    %
%         if ((H(i,j)-P1(i,j,2))*delta)>=P1Burial                          %
%             P1(i,j,1)=0;                                                 %
%             P1(i,j,2)=0;                                                 %
%         end                                                              %
%                                                                          %
%         if ((H(i,j)-P2(i,j,2))*delta)>=P2Burial                          %
%             P2(i,j,1)=0;                                                 %
%             P2(i,j,2)=0;                                                 %
%         end                                                              %
%                                                                          %
%         if ((H(i,j)-P3(i,j,2))*delta)>=P3Burial                          %
%             P3(i,j,1)=0;                                                 %
%             P3(i,j,2)=0;                                                 %
%         end                                                              %
%     end                                                                  %
% end                                                                      %
%                    DEATH BY COMPETITION                                  %
% for i=1:size(H,1)                                                        %
%     for j=1:size(H,2)                                                    %
%         if (P1(i,j,1)+P2(i,j,1)+P3(i,j,1))>1                             %
%             k=((P1(i,j,1)+P2(i,j,1)+P3(i,j,1))-1);                       %
%             if P1(i,j,1)<(k/2)                                           %
%                 P2(i,j,1)=P2(i,j,1)+P1(i,j,1)-k;                         %
%             elseif P2(i,j,1)<(k/2)                                       %
%                 P1(i,j,1)=P1(i,j,1)+P2(i,j,1)-k;                         %
%             else                                                         %
%                 P1(i,j,1)=P1(i,j,1)-(k/2);                                 %
%                 P2(i,j,1)=P2(i,j,1)-(k/2);                                 %
%             end                                                            %
%         end                                                                %
%     end                                                                    %
% end                                                                        %
%% DEATH BY COMPETITION OCCURS after PROPAGATION                            %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% PLANT PROPAGATION %                                                       %
%%%%%%%%%%%%%%%%%%%%%                                                       %
for i=1:size(H,1)                                                           %
    for j=1:size(H,2)                                                       %
        if P1(i,j,1)>0 && i>1 && j>1 && j<size(H,2) && i<size(H,1) %AMMOPHILA%
                                       %
            if 0.5<rand                        %CURRENT CELL                %
                P1(i,j,1)=((randi([1,5]))/100)+P1(i,j,1);                   %
            end                                                             %
            if 0.5<rand && P1(i+1,j,1)>=0   %NORTH                          %
                P1(i+1,j,1)=((randi([1,5]))/100)+P1(i+1,j,1);               %
            end                                                             %
            if 0.5<rand && P1(i+1,j+1,1)>=0 %NORTHEAST                      %
                P1(i+1,j+1,1)=((randi([1,5]))/100)+P1(i+1,j+1,1);           %
            end                                                             %
            if 0.5<rand && P1(i,j+1,1)>=0   %EAST                           %
                P1(i,j+1,1)=((randi([1,5]))/100)+P1(i,j+1,1);               %
            end                                                             %
            if 0.5<rand && P1(i-1,j+1,1)>=0 %SOUTHEAST                      %
                P1(i-1,j+1,1)=((randi([1,5]))/100)+P1(i-1,j+1,1);           %
            end                                                             %
            if 0.5<rand && P1(i-1,j,1)>=0   %SOUTH                          %
                P1(i-1,j,1)=((randi([1,5]))/100)+P1(i-1,j,1);               %
            end                                                             %
            if 0.5<rand && P1(i-1,j-1,1)>=0 %SOUTHWEST                      %
                P1(i-1,j-1,1)=((randi([1,5]))/100)+P1(i-1,j-1,1);           %
            end                                                             %
            if 0.5<rand && P1(i,j-1,1)>=0   %WEST                           %
                P1(i,j-1,1)=((randi([1,5]))/100)+P1(i,j-1,1);               %
            end                                                             %
            if 0.5<rand && P1(i+1,j-1,1)>=0 %NORTHWEST                      %
                P1(i+1,j-1,1)=((randi([1,5]))/100)+P1(i+1,j-1,1);           %
            end                                                             %
        end                                                                 %
```

```
                                                                  %
if P2(i,j,1)>0 && i>1 && j>1 && j<size(H,2) && i<size(H,1)%SPARTINA%
                              %
   if 0.5<rand                       %CURRENT CELL                %
      P2(i,j,1)=((randi([1,5]))/100)+P2(i,j,1);                   %
   end                                                            %
   if 0.5<rand && P2(i+1,j,1)>=0     %NORTH                       %
      P2(i+1,j,1)=((randi([1,5]))/100)+P2(i+1,j,1);               %
   end                                                            %
   if 0.5<rand && P2(i+1,j+1,1)>=0 %NORTHEAST                     %
      P2(i+1,j+1,1)=((randi([1,5]))/100)+P2(i+1,j+1,1);           %
   end                                                            %
   if 0.5<rand && P2(i,j+1,1)>=0     %EAST                        %
      P2(i,j+1,1)=((randi([1,5]))/100)+P2(i,j+1,1);               %
   end                                                            %
   if 0.5<rand && P2(i-1,j+1,1)>=0 %SOUTHEAST                     %
      P2(i-1,j+1,1)=((randi([1,5]))/100)+P2(i-1,j+1,1);           %
   end                                                            %
   if 0.5<rand && P2(i-1,j,1)>=0     %SOUTH                       %
      P2(i-1,j,1)=((randi([1,5]))/100)+P2(i-1,j,1);               %
   end                                                            %
   if 0.5<rand && P2(i-1,j-1,1)>=0 %SOUTHWEST                     %
      P2(i-1,j-1,1)=((randi([1,5]))/100)+P2(i-1,j-1,1);           %
   end                                                            %
   if 0.5<rand && P2(i,j-1,1)>=0     %WEST                        %
      P2(i,j-1,1)=((randi([1,5]))/100)+P2(i,j-1,1);               %
   end                                                            %
   if 0.5<rand && P2(i+1,j-1,1)>=0 %NORTHWEST                     %
      P2(i+1,j-1,1)=((randi([1,5]))/100)+P2(i+1,j-1,1);           %
   end                                                            %
end                                                               %
                                                                  %
if P4(i,j,1)>0 && i>1 && j>1 && j<size(H,2) && i<size(H,1) %SPARTINA (MARSH)%
                         %
   if 0.5<rand                       %CURRENT CELL                %
      P4(i,j,1)=((randi([1,5]))/100)+P4(i,j,1);                   %
   end                                                            %
   if 0.5<rand && P4(i+1,j,1)>=0     %NORTH                       %
      P4(i+1,j,1)=((randi([1,5]))/100)+P4(i+1,j,1);               %
   end                                                            %
   if 0.5<rand && P4(i+1,j+1,1)>=0 %NORTHEAST                     %
      P4(i+1,j+1,1)=((randi([1,5]))/100)+P4(i+1,j+1,1);           %
   end                                                            %
   if 0.5<rand && P4(i,j+1,1)>=0     %EAST                        %
      P4(i,j+1,1)=((randi([1,5]))/100)+P4(i,j+1,1);               %
```

```
            end                                                         %
            if 0.5<rand && P4(i-1,j+1,1)>=0 %SOUTHEAST                  %
                P4(i-1,j+1,1)=((randi([1,5]))/100)+P4(i-1,j+1,1);       %
            end                                                         %
            if 0.5<rand && P4(i-1,j,1)>=0   %SOUTH                      %
                P4(i-1,j,1)=((randi([1,5]))/100)+P4(i-1,j,1);           %
            end                                                         %
            if 0.5<rand && P4(i-1,j-1,1)>=0 %SOUTHWEST                  %
                P4(i-1,j-1,1)=((randi([1,5]))/100)+P4(i-1,j-1,1);       %
            end                                                         %
            if 0.5<rand && P4(i,j-1,1)>=0   %WEST                       %
                P4(i,j-1,1)=((randi([1,5]))/100)+P4(i,j-1,1);           %
            end                                                         %
            if 0.5<rand && P4(i+1,j-1,1)>=0 %NORTHWEST                  %
                P4(i+1,j-1,1)=((randi([1,5]))/100)+P4(i+1,j-1,1);       %
            end                                                         %
        end                                                             %
                                                                        %
                %MORELLA%                                               %
         %it is given an opportunity to propagate anywhere on the island  %
         %since its seeds are dispersed by birds.                        %
                                                                        %
         %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
         %         %%NEWLY TURNED OFF%%                                    %
         if (delta*H(i,j))>=1.5 && (delta*H(i,j))<=2.5                  %
             prob=rand;                                                 %
             if 0.1<prob && P3(i,j,1)>=0                                %
                 P3(i,j,1)=(P3(i,j,1)+((randi([1,5])/100)));            %
             end                                                        %
         end                                                            %
         %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    end                                                                 %
end                                                                     %

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% ROUTINE DEATH  (brand new)   %                                        %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%                                         %
for i=1:size(H,1)                                                       %
    for j=1:size(H,2)                                                   %
        if P1(i,j,1)>0 && i>1 && j>1 && j<size(H,2) && i<size(H,1) %AMMOPHILA%
                                        %
            if 0.5<rand                         %CURRENT CELL           %
                P1(i,j,1)=P1(i,j,1)-((randi([0,4]))/100);               %
            end                                                         %
            if 0.5<rand && P1(i-1,j-1,1)>0  %North-West                  %
                P1(i-1,j-1,1)=P1(i-1,j-1,1)-((randi([0,4]))/100);          %
                if P1(i-1,j-1,1)<0                                         %
                    P1(i-1,j-1,1)=0;                                       %
                end                                                        %
            end                                                            %
            if 0.5<rand && P1(i-1,j,1)>0  %North                            %
                P1(i-1,j,1)=P1(i-1,j,1)-((randi([0,4]))/100);              %
                if P1(i-1,j,1)<0                                           %
                    P1(i-1,j,1)=0;                                         %
                end                                                        %
            end                                                            %
            if 0.5<rand && P1(i-1,j+1,1)>0  %North-East                     %
                P1(i-1,j+1,1)=P1(i-1,j+1,1)-((randi([0,4]))/100);          %
                if P1(i-1,j+1,1)<0                                         %
                    P1(i-1,j+1,1)=0;                                       %
                end                                                        %
            end                                                            %
            if 0.5<rand && P1(i,j+1,1)>0  %East                             %
                P1(i,j+1,1)=P1(i,j+1,1)-((randi([0,4]))/100);              %
                if P1(i,j+1,1)<0                                           %
                    P1(i,j+1,1)=0;                                         %
                end                                                        %
            end                                                            %
            if 0.5<rand && P1(i+1,j+1,1)>0  %South-East                     %
                P1(i+1,j+1,1)=P1(i+1,j+1,1)-((randi([0,4]))/100);          %
                if P1(i+1,j+1,1)<0                                         %
                    P1(i+1,j+1,1)=0;                                       %
                end                                                        %
            end                                                            %
            if 0.5<rand && P1(i+1,j,1)>0  %South                            %
                P1(i+1,j,1)=P1(i+1,j,1)-((randi([0,4]))/100);              %
                if P1(i+1,j,1)<0                                           %
                    P1(i+1,j,1)=0;                                         %
                end                                                        %
            end                                                            %
            if 0.5<rand && P1(i+1,j-1,1)>0  %South-West                     %
                P1(i+1,j-1,1)=P1(i+1,j-1,1)-((randi([0,4]))/100);          %
                if P1(i+1,j-1,1)<0                                         %
                    P1(i+1,j-1,1)=0;                                       %
                end                                                        %
            end                                                            %
            if 0.5<rand && P1(i,j-1,1)>0  %West                             %
                P1(i,j-1,1)=P1(i,j-1,1)-((randi([0,4]))/100);              %
                if P1(i,j-1,1)<0                                           %
                    P1(i,j-1,1)=0;                                         %
                end                                                         %
            end                                                             %
        end                                                                 %
                                                                            %
                                                                            %
        if P2(i,j,1)>0 && i>1 && j>1 && j<size(H,2) && i<size(H,1) %SPARTINA%
                                        %
            if 0.5<rand                     %CURRENT CELL                   %
                P2(i,j,1)=P2(i,j,1)-((randi([0,4]))/100);                   %
            end                                                             %
            if 0.5<rand && P2(i-1,j-1,1)>0  %North-West                      %
                P2(i-1,j-1,1)=P2(i-1,j-1,1)-((randi([0,4]))/100);           %
                if P2(i-1,j-1,1)<0                                          %
                    P2(i-1,j-1,1)=0;                                        %
                end                                                         %
            end                                                             %
            if 0.5<rand && P2(i-1,j,1)>0  %North                             %
                P2(i-1,j,1)=P2(i-1,j,1)-((randi([0,4]))/100);               %
                if P2(i-1,j,1)<0                                            %
                    P2(i-1,j,1)=0;                                          %
                end                                                         %
            end                                                             %
            if 0.5<rand && P2(i-1,j+1,1)>0  %North-East                      %
                P2(i-1,j+1,1)=P2(i-1,j+1,1)-((randi([0,4]))/100);           %
                if P2(i-1,j+1,1)<0                                          %
                    P2(i-1,j+1,1)=0;                                        %
                end                                                         %
            end                                                             %
            if 0.5<rand && P2(i,j+1,1)>0  %East                              %
                P2(i,j+1,1)=P2(i,j+1,1)-((randi([0,4]))/100);               %
                if P2(i,j+1,1)<0                                            %
                    P2(i,j+1,1)=0;                                          %
                end                                                         %
            end                                                             %
            if 0.5<rand && P2(i+1,j+1,1)>0  %South-East                      %
                P2(i+1,j+1,1)=P2(i+1,j+1,1)-((randi([0,4]))/100);           %
                if P2(i+1,j+1,1)<0                                          %
                    P2(i+1,j+1,1)=0;                                        %
                end                                                         %
            end                                                             %
            if 0.5<rand && P2(i+1,j,1)>0  %South                             %
                P2(i+1,j,1)=P2(i+1,j,1)-((randi([0,4]))/100);               %
                if P2(i+1,j,1)<0                                            %
                    P2(i+1,j,1)=0;                                          %
                end                                                         %
            end                                                             %
            if 0.5<rand && P2(i+1,j-1,1)>0  %South-West                      %
                P2(i+1,j-1,1)=P2(i+1,j-1,1)-((randi([0,4]))/100);           %
                if P2(i+1,j-1,1)<0                                          %
                    P2(i+1,j-1,1)=0;                                        %
                end                                                         %
            end                                                             %
            if 0.5<rand && P2(i,j-1,1)>0  %West                              %
                P2(i,j-1,1)=P2(i,j-1,1)-((randi([0,4]))/100);               %
                if P2(i,j-1,1)<0                                            %
                    P2(i,j-1,1)=0;                                          %
                end                                                         %
            end                                                             %
        end                                                                 %
                                                                            %
                                                                            %
        if P3(i,j,1)>0 && i>1 && j>1 && j<size(H,2) && i<size(H,1) %MORELLA%
                                        %
            if 0.5<rand                     %CURRENT CELL                   %
                P3(i,j,1)=P3(i,j,1)-((randi([0,4]))/100);                   %
            end                                                             %
            if 0.5<rand && P3(i-1,j-1,1)>0  %North-West                      %
                P3(i-1,j-1,1)=P3(i-1,j-1,1)-((randi([0,4]))/100);           %
                if P3(i-1,j-1,1)<0                                          %
                    P3(i-1,j-1,1)=0;                                        %
                end                                                         %
            end                                                             %
            if 0.5<rand && P3(i-1,j,1)>0  %North                             %
                P3(i-1,j,1)=P3(i-1,j,1)-((randi([0,4]))/100);               %
                if P3(i-1,j,1)<0                                            %
                    P3(i-1,j,1)=0;                                          %
                end                                                         %
            end                                                             %
            if 0.5<rand && P3(i-1,j+1,1)>0  %North-East                      %
                P3(i-1,j+1,1)=P3(i-1,j+1,1)-((randi([0,4]))/100);           %
                if P3(i-1,j+1,1)<0                                          %
                    P3(i-1,j+1,1)=0;                                        %
                end                                                         %
            end                                                             %
            if 0.5<rand && P3(i,j+1,1)>0  %East                              %
                P3(i,j+1,1)=P3(i,j+1,1)-((randi([0,4]))/100);               %
                if P3(i,j+1,1)<0                                            %
                    P3(i,j+1,1)=0;                                          %
                end                                                         %
            end                                                             %
```

```
        if 0.5<rand && P3(i+1,j+1,1)>0  %South-East                   %
            P3(i+1,j+1,1)=P3(i+1,j+1,1)-((randi([0,4]))/100);        %
            if P3(i+1,j+1,1)<0                                        %
                P3(i+1,j+1,1)=0;                                      %
            end                                                       %
        end                                                           %
        if 0.5<rand && P3(i+1,j,1)>0  %South                          %
            P3(i+1,j,1)=P3(i+1,j,1)-((randi([0,4]))/100);            %
            if P3(i+1,j,1)<0                                          %
                P3(i+1,j,1)=0;                                        %
            end                                                       %
        end                                                           %
        if 0.5<rand && P3(i+1,j-1,1)>0  %South-West                   %
            P3(i+1,j-1,1)=P3(i+1,j-1,1)-((randi([0,4]))/100);        %
            if P3(i+1,j-1,1)<0                                        %
                P3(i+1,j-1,1)=0;                                      %
            end                                                       %
        end                                                           %
        if 0.5<rand && P3(i,j-1,1)>0  %West                           %
            P3(i,j-1,1)=P3(i,j-1,1)-((randi([0,4]))/100);            %
            if P3(i,j-1,1)<0                                          %
                P3(i,j-1,1)=0;                                        %
            end                                                       %
        end                                                           %
    end                                                               %
                                                                      %
                                                                      %
    if P4(i,j,1)>0 && i>1 && j>1 && j<size(H,2) && i<size(H,1) %SPARTINA (marsh)%
                        %
        if 0.5<rand                     %CURRENT CELL                 %
            P4(i,j,1)=P4(i,j,1)-((randi([0,4]))/100);                %
        end                                                           %
        if 0.5<rand && P4(i-1,j-1,1)>0  %North-West                   %
            P4(i-1,j-1,1)=P4(i-1,j-1,1)-((randi([0,4]))/100);        %
            if P4(i-1,j-1,1)<0                                        %
                P4(i-1,j-1,1)=0;                                      %
            end                                                       %
        end                                                           %
        if 0.5<rand && P4(i-1,j,1)>0  %North                          %
            P4(i-1,j,1)=P4(i-1,j,1)-((randi([0,4]))/100);            %
            if P4(i-1,j,1)<0                                          %
                P4(i-1,j,1)=0;                                        %
            end                                                       %
        end                                                           %
        if 0.5<rand && P4(i-1,j+1,1)>0  %North-East                   %
            P4(i-1,j+1,1)=P4(i-1,j+1,1)-((randi([0,4]))/100);         %
            if P4(i-1,j+1,1)<0                                          %
                P4(i-1,j+1,1)=0;                                        %
            end                                                         %
        end                                                             %
        if 0.5<rand && P4(i,j+1,1)>0  %East                              %
            P4(i,j+1,1)=P4(i,j+1,1)-((randi([0,4]))/100);               %
            if P4(i,j+1,1)<0                                            %
                P4(i,j+1,1)=0;                                          %
            end                                                         %
        end                                                             %
        if 0.5<rand && P4(i+1,j+1,1)>0  %South-East                      %
            P4(i+1,j+1,1)=P4(i+1,j+1,1)-((randi([0,4]))/100);           %
            if P4(i+1,j+1,1)<0                                          %
                P4(i+1,j+1,1)=0;                                        %
            end                                                         %
        end                                                             %
        if 0.5<rand && P4(i+1,j,1)>0  %South                             %
            P4(i+1,j,1)=P4(i+1,j,1)-((randi([0,4]))/100);               %
            if P4(i+1,j,1)<0                                            %
                P4(i+1,j,1)=0;                                          %
            end                                                         %
        end                                                             %
        if 0.5<rand && P4(i+1,j-1,1)>0  %South-West                      %
            P4(i+1,j-1,1)=P4(i+1,j-1,1)-((randi([0,4]))/100);           %
            if P4(i+1,j-1,1)<0                                          %
                P4(i+1,j-1,1)=0;                                        %
            end                                                         %
        end                                                             %
        if 0.5<rand && P4(i,j-1,1)>0  %West                              %
            P4(i,j-1,1)=P4(i,j-1,1)-((randi([0,4]))/100);               %
            if P4(i,j-1,1)<0                                            %
                P4(i,j-1,1)=0;                                          %
            end                                                         %
        end                                                             %
    end                                                                 %
                                                                      %
                                                                      %


                                                              %
                                                                      %
            %MORELLA%                                                   %
     %it is given an opportunity to propagate anywhere on the island    %
     %since its seeds are dispersed by birds.                           %
```

```
                                                                      %
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        %        %%NEWLY TURNED OFF%%                                     %
        %        %% THIS WAS TURNED ON ABOVE... IT IS REDUNDANT HERE!     %
        %if (delta*H(i,j))>=1.5 && (delta*H(i,j))<=2.5                    %
        %    prob=rand;                                                   %
        %    if 0.1<prob && P3(i,j,1)>=0                                  %
        %        P3(i,j,1)=(P3(i,j,1)+((randi([1,5])/100)));              %
        %    end                                                          %
        %end                                                              %
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    end                                                               %
end

%
% DEATH BY COMPETITION (to rectify newly propagated plants)           %
%%%%%%%%%%  MOVED TO BOTTOM  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% for i=1:size(H,1)                                                    %
%     for j=1:size(H,2)                                                %
%         if (P1(i,j)+P2(i,j)+P3(i,j))>1                               %
%             k=((P1(i,j)+P2(i,j)+P3(i,j))-1);                         %
%
%             P1(i,j)=P1(i,j)-k;  %this is for the current simulation only!
%
%             if P1(i,j)<(k/2)                                         %
%                 P2(i,j)=P2(i,j)+P1(i,j)-k;                           %
%             elseif P2(i,j)<(k/2)                                     %
%                 P1(i,j)=P1(i,j)+P2(i,j)-k;                           %
%             else                                                     %
%                 P1(i,j)=P1(i,j)-(k/2);                               %
%                 P2(i,j)=P2(i,j)-(k/2);                               %
%             end                                                      %
%         end                                                          %
%     end                                                              %
% end                                                                  %
                                                                       %
                                                                      %
% DEATH BY ELEVATION (to rectify newly propagated plants) && CATEGORY %
%
%%% One might notice that Morella does not propagate on-cell until later in
%%% this program. That is OK since death by competition comes last.
%%% It is easier to simular plant growth within the propagation for the
%%% other grasses, since they propagate locally AND grow locally.
%%% Morella grows locally and propagates at random, so we simulate Routine
%%% Death immediately after routine growth, although Morella does not
%%% experience local growth (or propagation) until later.
%%% Order of operations would designate this organization negligible (it
%%% does not impact the procedure) but any semantic differences would only
%%% promote morella's invasiveness, which is OK.
for i=1:size(H,1)                                                      %
    for j=1:size(H,2)                                                  %
        if P1(i,j,1)>0 && (delta*H(i,j))<1                             %
            P1(i,j,1)=0;                                               %
            P1(i,j,2)=0;                                               %
        end                                                            %
        if P1(i,j,1)>0 && (delta*H(i,j))>5                             %
            P1(i,j,1)=0;                                               %
            P1(i,j,2)=0;                                               %
        end                                                            %
                                                                       %

        if P2(i,j,1)>0 && (delta*H(i,j))<0.75                          %
            P2(i,j,1)=0;                                               %
            P2(i,j,2)=0;                                               %
        end                                                            %
        if P2(i,j,1)>0 && (delta*H(i,j))>3                             %
            P2(i,j,1)=0;                                               %
             P2(i,j,2)=0;                                               %
        end                                                            %
                                                                       %
        if P3(i,j,1)>0 && (delta*H(i,j))<1.5                           %
            P3(i,j,1)=0;                                               %
            P3(i,j,2)=0;                                               %
        end                                                            %
        if P3(i,j,1)>0 && (delta*H(i,j))>2.5                           %
            P3(i,j,1)=0;                                               %
            P3(i,j,2)=0;                                               %
        end                                                            %
                                                                       %
        %if P4(i,j,1)>0 && (delta*H(i,j))<-0.5                          %
        %    P4(i,j,1)=0;                                               %
        %    P4(i,j,2)=0;                                               %
        %end                                                            %
        %if P4(i,j,1)>0 && (delta*H(i,j))>1                             %
        %    P4(i,j,1)=0;                                               %
        %    P4(i,j,2)=0;                                               %
        %end                                                            %
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        %%% Death by "land category"
        %%%
        if P1(i,j,1)>0 && H(i,j,2)==2
            P1(i,j,1)=0;
            P1(i,j,2)=0;
        end
        if P2(i,j,1)>0 && H(i,j,2)==2
            P2(i,j,1)=0;
            P2(i,j,2)=0;
        end
        if P3(i,j,1)>0 && H(i,j,2)==2
            P3(i,j,1)=0;
            P3(i,j,1)=0;
        end
        if P4(i,j,1)>0 && H(i,j,2)~=2
            P4(i,j,1)=0;
            P4(i,j,2)=0;
        end
    end                                                                       %
end                                                                           %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                           %
%   ALL OF GROWTH IS NEWLY TURNED OFF !!!!!!!!!!!!!!!!!!    %
%         (below look for lines with only one %)            %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% PLANT GROWTH %                                                              %
%%%%%%%%%%%%%%%%                                                              %
%
for i=1:size(H,1)                                                             %
    for j=1:size(H,2)                                                         %
        %%%prob=rand;                                                           %
        %%%if P1(i,j,1)>0 && 0.5<prob %AMMOPHILA%                               %
        %%%    P1(i,j,1)=(1+((randi([5,15]))/100))*P1(i,j,1);                   %
        %%%end                                                                  %
        %%%prob=rand;                                                           %
        %%%if P2(i,j,1)>0 && 0.5<prob %SPARTINA%                                %
        %%%    P2(i,j,1)=(1+((randi([5,15]))/100))*P2(i,j,1);                   %
        %%%end                                                                  %
        prob=rand;                                                            %
        if P3(i,j,1)>0 && 0.5<prob %MORELLA%                                  %
```

```
            P3(i,j,1)=(1+((randi([5,15]))/100))*P3(i,j,1);              %
        end                                                             %
    end                                                                 %
end                                                                     %
%
%%% FOR P1, P2, & P4 WE SIMULATE GROWTH DURING PROPAGATION ON CURRENT CELL %%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% PLANT DEATH %                                                         %
%%%%%%%%%%%%%%%                                                         %
%                      DEATH BY BURIAL                                  %
% currently, the burial is ONLY comparing the elevation of sand to the  %
% elevation of where plants first spawned. Eventually, we hope to have  %
% burial compare present year to previous year.                         %

%%%% CURRENT VALUES ARE MADE UP!!!!!!!!!!
%%% IF BURIAL VALUES NOT PROPERLY INITIATED IN MAIN CODE
%%%%
%%% DELETE ME! DELETE ME!! DELETE ME!!! DELETE ME!!!!
 for i=1:size(H,1)                                                      %
     for j=1:size(H,2)                                                  %
         if ((H(i,j)-P1(i,j,2))*delta)>=P1Burial                        %
             P1(i,j,1)=0;                                               %
             P1(i,j,2)=0;                                               %
         end                                                            %
                                                                        %
         if ((H(i,j)-P2(i,j,2))*delta)>=P2Burial                        %
             P2(i,j,1)=0;                                               %
             P2(i,j,2)=0;                                               %
         end                                                            %
                                                                        %
         if ((H(i,j)-P3(i,j,2))*delta)>=P3Burial                        %
             P3(i,j,1)=0;                                               %
             P3(i,j,2)=0;                                               %
         end                                                            %
                                                                        %
         if ((H(i,j)-P4(i,j,2))*delta)>=P4Burial                        %
             P4(i,j,1)=0;                                               %
             P4(i,j,2)=0;                                               %
         end                                                            %
     end                                                                %
 end                                                                    %
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
                  %DEATH BY COMPETITION                                   %
                  % only one plant in current simulations so no
                  % competition!!!

for i=1:size(H,1)                                                         %
    for j=1:size(H,2)                                                     %
        if (P1(i,j,1)+P2(i,j,1)+P3(i,j,1)+P4(i,j,1))>1                    %
            % k represents the amount that a cell is saturated
            % (i.e. how much above 1 the total concentration of plants is)
            k=((P1(i,j,1)+P2(i,j,1)+P3(i,j,1)+P4(i,j,1))-1);              %

            %P1(i,j,1)=P1(i,j,1)-k;    %This is only for current simulation!

            %All three are bigger than k/3                                %
            if P1(i,j,1)>=(k/3) && P2(i,j,1)>=(k/3) && P4(i,j,1)>=(k/3)  %
                P1(i,j,1)=P1(i,j,1)-(k/3);                                %
                P2(i,j,1)=P2(i,j,1)-(k/3);                                %
                P4(i,j,1)=P4(i,j,1)-(k/3);                                %
            %Only two are bigger than k/3                                 %
            %%1 2%%                                                       %
            elseif P1(i,j,1)>=(k/3) && P2(i,j,1)>=(k/3) && P4(i,j,1)<(k/3)
                P1(i,j,1)=P1(i,j,1)-((k/3)+(((k/3)-P4(i,j,1))/2));        %
                P2(i,j,1)=P2(i,j,1)-((k/3)+(((k/3)-P4(i,j,1))/2));        %
                P4(i,j,1)=0;                                              %
            %%1 4%%                                                       %
            elseif P1(i,j,1)>=(k/3) && P2(i,j,1)<(k/3) && P4(i,j,1)>=(k/3)
                P1(i,j,1)=P1(i,j,1)-((k/3)+(((k/3)-P4(i,j,1))/2));        %
                P2(i,j,1)=0;                                              %
                P4(i,j,1)=P4(i,j,1)-((k/3)+(((k/3)-P2(i,j,1))/2));        %
            %%2 4%%                                                       %
            elseif P1(i,j,1)<(k/3) && P2(i,j,1)>=(k/3) && P4(i,j,1)>=(k/3)
                P1(i,j,1)=0;                                              %
                P2(i,j,1)=P2(i,j,1)-((k/3)+(((k/3)-P1(i,j,1))/2));        %
                P4(i,j,1)=P4(i,j,1)-((k/3)+(((k/3)-P1(i,j,1))/2));        %
            %Only one is bigger than k/3%                                 %
            %%1%%                                                         %
            elseif P1(i,j,1)>=(k/3) && P2(i,j,1)<(k/3) && P4(i,j,1)<(k/3)%
                P1(i,j,1)=P1(i,j,1)-((k/3)+((k/3)-P2(i,j,1))+((k/3)-P4(i,j,1)));
                P2(i,j,1)=0;                                              %
                P4(i,j,1)=0;                                              %
            %%2%%                                                         %
            elseif P1(i,j,1)<(k/3) && P2(i,j,1)>=(k/3) && P4(i,j,1)<(k/3)%
                P1(i,j,1)=0;                                              %
```

```
                 P2(i,j,1)=P2(i,j,1)-((k/3)+((k/3)-P1(i,j,1))+((k/3)-P4(i,j,1)));
                 P4(i,j,1)=0;                                             %
             %%4%%                                                        %
             elseif P1(i,j,1)<(k/3) && P2(i,j,1)<(k/3) && P4(i,j,1)>=(k/3)%
                 P1(i,j,1)=0;                                             %
                 P2(i,j,1)=0;                                             %
                 P4(i,j,1)=P4(i,j,1)-((k/3)+((k/3)-P1(i,j,1))+((k/3)-P2(i,j,1)));
             end                                                          %
         end                                                              %
     end                                                                  %
 end                                                                      %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%% FOLLOWING THIS PROCEDURE.... %%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% RUN MARINE PROCESSES TO KILL PLANTS WHICH ARE TOO CLOSE TO THE SHORE %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
function [H,Hstar]=AeolianTransport(Hstar,delta,L,H,W,Windspeed,Windmin,StormThreshold,
WindDir,n,Pe,Pd,PC)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
fprintf('I am running AT! ')
for i=1:size(H,1) %for each row
                                                                       %
    for j=size(H,2):-1:1 %for each column
                                                                       %
        if H(i,j,2)~=2
        %
        numslabc=floor((((delta*H(i,j))-W(i,j))/delta));   %total number of moveable
slabs in the current cell                                       %
        slab=ceil(delta*L*(Pe/Pd)*n);    %calculated number of moveable slabs
in the current cell                                                    %

                                                                       %
        movedslabs=min(slab,numslabc);          %number of slabs to be moved in
the current cell                                                     %

                                                                       %

                                                                       %
        %while movedslabs>0&&(delta*(H(i,j)-1))>=W(i,j)&&Windspeed>Windmin
                                                                        %
        %while (delta*(H(i,j)-1))>=W(i,j) && Windspeed>Windmin
        %hop=i+Salt;                            helps determine how far the slab
will move                                                       %
            %if hop>size(H,2)||hop<1
                                                                       %
            %    break
                                                                       %
            %end
                                                                       %
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
                                                    %

                                                                       %
            %probability for RNG use
                                                                       %
            prob=rand;
                                                                       %
            %P=1.0;
            P=0.5;%
            Rand=rand;
                                                                        %
            %R=rand;
                                                                        %
            MoveChance=(1-PC(i,j))*((max(Windspeed-Windmin,0))/(StormThreshold-Windmin))
            %MoveChance=((max(Windspeed-Windmin,0))/(StormThreshold-Windmin));

                                                                        %
            %%%%%%%%%%%%%%%%%%%%%%%
                                                                        %
            %% DEPOSITION RULES: %%
                                                                        %
            %%%%%%%%%%%%%%%%%%%%%%%
                                                                        %

                                                                        %
            %Wind Direction
                                                                        %
        %1=N  2=NE  3=E  4=SE  5=S  6=SW  7=W  8=NW
                                                                        %



        %
            if WindDir==1 && rand<MoveChance       %North
                                                                        %
                if Rand<((1-P)/2) && (i>1) && (j>1) %move to the port slab

            %
                    if atan(((H(i-1,j-1)-H(i,j))*delta)/L)<(pi/12)      %slope
    between slabs must be sufficiently shallow (15 degrees)            %
                        Hstar(i,j)=Hstar(i,j)-1;
                                                                        %
                        Hstar(i-1,j-1)=Hstar(i-1,j-1)+1;
                                                                        %
                        fprintf('N L \n')
                                                                        %
                    end
                elseif Rand<((1-P)/2) && ((i==1) || (j==1))
                    Hstar(i,j)=Hstar(i,j)-1;    %
                elseif Rand>=((1-P)/2) && Rand<((1+P)/2) && (i>1)   %move with
    the wind
       %
                    if atan(((H(i-1,j)-H(i,j))*delta)/L)<(pi/12)
```

```
                                                                          %
                        Hstar(i,j)=Hstar(i,j)-1;
                                                                          %
                        Hstar(i-1,j)=Hstar(i-1,j)+1;
                                                                          %
                        fprintf('N M \n')
                                                                          %
                    end
                elseif Rand>=((1-P)/2) && Rand<((1+P)/2) && (i==1)
                    Hstar(i,j)=Hstar(i,j)-1;        %
                elseif Rand>=((1+P)/2) && (i>1) && (j<size(Hstar,2))       %move
to the starboard slab
                                %
                    if atan(((H(i-1,j+1)-H(i,j))*delta)/L)<(pi/12)
                                                                          %
                        Hstar(i,j)=Hstar(i,j)-1;
                                                                          %
                        Hstar(i-1,j+1)=Hstar(i-1,j+1)+1;
                                                                          %
                        fprintf('N R \n')
                                                                          %
                    end
                elseif Rand>=((1+P)/2) && ((i==1) || (j==size(Hstar,2)))
                    Hstar(i,j)=Hstar(i,j)-1;        %
                end



                %
            elseif WindDir==2 && rand<MoveChance      %North-East
                %MOVE N
                if Rand<((1-P)/2) && (i>1)

   %
                    if atan(((H(i-1,j)-H(i,j))*delta)/L)<(pi/12)
                                                                          %
                        Hstar(i,j)=Hstar(i,j)-1;
                                                                          %
                        Hstar(i-1,j)=Hstar(i-1,j)+1;
                                                                          %
                        fprintf('NE L \n')
                                                                          %
                    end
                elseif Rand<((1-P)/2) && (i==1)
                    Hstar(i,j)=Hstar(i,j)-1;
                    %MOVE NE
                elseif Rand>=((1-P)/2) && Rand<((1+P)/2) && (i>1) && (j<size(Hstar,2))

                  %
                    if atan(((H(i-1,j+1)-H(i,j))*delta)/L)<(pi/12)
                                                                                    %
                        Hstar(i,j)=Hstar(i,j)-1;
                                                                                    %
                        Hstar(i-1,j+1)=Hstar(i-1,j+1)+1;
                                                                                    %
                        fprintf('NE M \n')
                                                                                    %
                    end
                elseif Rand>=((1-P)/2) && Rand<((1+P)/2) && ((i==1) ||
    (j==size(Hstar,2)))
                    Hstar(i,j)=Hstar(i,j)-1;
                    %MOVE E
                elseif Rand>=((1+P)/2) && (j<size(Hstar,2))

                                %
                    if atan(((H(i,j+1)-H(i,j))*delta)/L)<(pi/12)
                                                                                    %
                        Hstar(i,j)=Hstar(i,j)-1;
                                                                                    %
                        Hstar(i,j+1)=Hstar(i,j+1)+1;
                                                                                    %
                        fprintf('NE R \n')
                                                                                    %
                    end
                elseif Rand>=((1+P)/2) && (j==size(Hstar,2))
                    Hstar(i,j)=Hstar(i,j)-1;        %
                end



                %
            elseif WindDir==3 && rand<MoveChance      %East
                %MOVES NE
                if Rand<((1-P)/2) && (i>1) && (j<size(Hstar,2))

                        %
                    if atan(((H(i-1,j+1)-H(i,j))*delta)/L)<(pi/12)
                                                                                    %
                        Hstar(i,j)=Hstar(i,j)-1;
                                                                                    %
                        Hstar(i-1,j+1)=Hstar(i-1,j+1)+1;
                                                                                    %
                        fprintf('E L \n')
                                                                                    %
                    end
                elseif Rand<((1-P)/2) && ((i==1) || (j==size(Hstar,2)))
                    Hstar(i,j)=Hstar(i,j)-1;
                    %MOVES E
                elseif Rand>=((1-P)/2) && Rand<((1+P)/2) && (j<size(Hstar,2))

                  %
                    if atan(((H(i,j+1)-H(i,j))*delta)/L)<(pi/12)
                                                                                    %
                        Hstar(i,j)=Hstar(i,j)-1;
                                                                                    %
                        Hstar(i,j+1)=Hstar(i,j+1)+1;
                                                                                    %
                        fprintf('E M \n')
                                                                                    %
                    end
                elseif Rand>=((1-P)/2) && Rand<((1+P)/2) && (j==size(Hstar,2))
                    Hstar(i,j)=Hstar(i,j)-1;
                    %MOVES SE
                elseif Rand>=((1+P)/2) && (i<size(Hstar,1)) && (j<size(Hstar,2))

                                                                 %
                    if atan(((H(i+1,j+1)-H(i,j))*delta)/L)<(pi/12)
                                                                                    %
                        Hstar(i,j)=Hstar(i,j)-1;
                                                                                    %
                        Hstar(i+1,j+1)=Hstar(i+1,j+1)+1;
                                                                                    %
                        fprintf('E R \n')
                                                                                    %
                    end
                elseif Rand>=((1+P)/2) && ((i==size(Hstar,1)) || (j==size(Hstar,2)))
                    Hstar(i,j)=Hstar(i,j)-1;            %
                end


                %
            elseif WindDir==4 && rand<MoveChance     %South-East
                %MOVES E
                if Rand<((1-P)/2) && (j<size(Hstar,2))
```

```
            %
                if atan(((H(i,j+1)-H(i,j))*delta)/L)<(pi/12)
                                                                          %
                    Hstar(i,j)=Hstar(i,j)-1;
                                                                          %
                    Hstar(i,j+1)=Hstar(i,j+1)+1;
                                                                          %
                    fprintf('SE L \n')
                                                                          %
                end
            elseif Rand<((1-P)/2) && (j==size(Hstar,2))
                Hstar(i,j)=Hstar(i,j)-1;
                %MOVES SE
            elseif Rand>=((1-P)/2) && Rand<((1+P)/2) && (i<size(Hstar,1))
&& (j<size(Hstar,2))
                                    %
                if atan(((H(i+1,j+1)-H(i,j))*delta)/L)<(pi/12)
                                                                          %
                    Hstar(i,j)=Hstar(i,j)-1;
                                                                          %
                    Hstar(i+1,j+1)=Hstar(i+1,j+1)+1;
                                                                          %
                    fprintf('SE M \n')
                                                                          %
                end
            elseif Rand>=((1-P)/2) && Rand<((1+P)/2) && ((i==size(Hstar,1))
|| (j==size(Hstar,2)))
                Hstar(i,j)=Hstar(i,j)-1;
                %MOVES S
            elseif Rand>=((1+P)/2) && (i<size(Hstar,1))

                                 %
                if atan(((H(i+1,j)-H(i,j))*delta)/L)<(pi/12)
                                                                          %
                    Hstar(i,j)=Hstar(i,j)-1;
                                                                          %
                    Hstar(i+1,j)=Hstar(i+1,j)+1;
                                                                          %
                    fprintf('SE R \n')
                                                                          %
                end
            elseif Rand>=((1+P)/2) && (i==size(Hstar,1))
                Hstar(i,j)=Hstar(i,j)-1;           %
            end
```

```
    %
elseif WindDir==5 && rand<MoveChance     %South
    %MOVES SE
    if Rand<((1-P)/2) && (i<size(Hstar,1)) && (j<size(Hstar,2))

                         %
        if atan(((H(i+1,j+1)-H(i,j))*delta)/L)<(pi/12)
                                                              %
            Hstar(i,j)=Hstar(i,j)-1;
                                                              %
            Hstar(i+1,j+1)=Hstar(i+1,j+1)+1;
                                                              %
            fprintf('S L \n')
                                                              %
        end
    elseif Rand<((1-P)/2) && ((i==size(Hstar,1)) || (j==size(Hstar,2)))
        Hstar(i,j)=Hstar(i,j)-1;
        %MOVES S
    elseif Rand>=((1-P)/2) && Rand<((1+P)/2) && (i<size(Hstar,1))

       %
        if atan(((H(i+1,j)-H(i,j))*delta)/L)<(pi/12)
                                                              %
            Hstar(i,j)=Hstar(i,j)-1;
                                                              %
            Hstar(i+1,j)=Hstar(i+1,j)+1;
                                                              %
            fprintf('S M \n')
                                                              %
        end
    elseif Rand>=((1-P)/2) && Rand<((1+P)/2) && (i==size(Hstar,1))
        Hstar(i,j)=Hstar(i,j)-1;
        %MOVES SW
    elseif Rand>=((1+P)/2) && (i<size(Hstar,1)) && (j>1)

                               %
        if atan(((H(i+1,j-1)-H(i,j))*delta)/L)<(pi/12)
                                                              %
            Hstar(i,j)=Hstar(i,j)-1;
                                                              %
            Hstar(i+1,j-1)=Hstar(i+1,j-1)+1;
                                                              %
            fprintf('S R \n')
                                                              %
```

```
                end
            elseif Rand>=((1+P)/2) && ((i==size(Hstar,1)) || (j==1))
                Hstar(i,j)=Hstar(i,j)-1;            %
            end


            %
        elseif WindDir==6 && rand<MoveChance       %South-West
            %MOVES S
            if Rand<((1-P)/2) && (i<size(Hstar,1))

            %
                if atan(((H(i+1,j)-H(i,j))*delta)/L)<(pi/12)
                                                                        %
                    Hstar(i,j)=Hstar(i,j)-1;
                                                                        %
                    Hstar(i+1,j)=Hstar(i+1,j)+1;
                                                                        %
                    fprintf('SW L \n')
                                                                        %
                end
            elseif Rand<((1-P)/2) && (i==size(Hstar,1))
                Hstar(i,j)=Hstar(i,j)-1;
                %MOVES SW
            elseif Rand>=((1-P)/2) && Rand<((1+P)/2) && (i<size(Hstar,1))
&& (j>1)
                        %
                if atan(((H(i+1,j-1)-H(i,j))*delta)/L)<(pi/12)
                                                                        %
                    Hstar(i,j)=Hstar(i,j)-1;
                                                                        %
                    Hstar(i+1,j-1)=Hstar(i+1,j-1)+1;
                                                                        %
                    fprintf('SW M \n')
                                                                        %
                end
            elseif Rand>=((1-P)/2) && Rand<((1+P)/2) && ((i==size(Hstar,1))
|| (j==1))
                Hstar(i,j)=Hstar(i,j)-1;
                %MOVES W
            elseif Rand>=((1+P)/2) && (j>1)

                  %
                if atan(((H(i,j-1)-H(i,j))*delta)/L)<(pi/12)
                                                                        %
                    Hstar(i,j)=Hstar(i,j)-1;
                                                                                    %
                    Hstar(i,j-1)=Hstar(i,j-1)+1;
                                                                                    %
                    fprintf('SW R \n')
                                                                                    %
                end
            elseif Rand>=((1+P)/2) && (j==1)
                Hstar(i,j)=Hstar(i,j)-1;                                   %
            end


            %
        elseif WindDir==7 && rand<MoveChance        %West
            %MOVES SW
            if Rand<((1-P)/2) && (i<size(Hstar,1)) && (j>1)

                    %
                if atan(((H(i+1,j-1)-H(i,j))*delta)/L)<(pi/12)
                                                                                    %
                    Hstar(i,j)=Hstar(i,j)-1;
                                                                                    %
                    Hstar(i+1,j-1)=Hstar(i+1,j-1)+1;
                                                                                    %
                    %fprintf('W L \n')
                                                                                     %
                end
            elseif Rand<((1-P)/2) && ((i==size(Hstar,1)) || (j==1))
                Hstar(i,j)=Hstar(i,j)-1;
                %MOVES W
            elseif Rand>=((1-P)/2) && Rand<((1+P)/2) && (j>1)

    %
                if atan(((H(i,j-1)-H(i,j))*delta)/L)<(pi/12)
                                                                                    %
                    Hstar(i,j)=Hstar(i,j)-1;
                                                                                    %
                    Hstar(i,j-1)=Hstar(i,j-1)+1;
                                                                                    %
                    %fprintf('W M \n')
                                                                                     %
                end
            elseif Rand>=((1-P)/2) && Rand<((1+P)/2) && (j==1)
                Hstar(i,j)=Hstar(i,j)-1;
                %MOVES NW
            elseif Rand>=((1+P)/2) && (i>1) && (j>1)

                           %
                if atan(((H(i-1,j-1)-H(i,j))*delta)/L)<(pi/12)
                                                                                    %
                    Hstar(i,j)=Hstar(i,j)-1;
                                                                                    %
                    Hstar(i-1,j-1)=Hstar(i-1,j-1)+1;
                                                                                    %
                    %fprintf('W R \n')
                                                                                     %
                end
            elseif Rand>=((1+P)/2) && ((i==1) || (j==1))
                Hstar(i,j)=Hstar(i,j)-1;            %
            end


            %
        elseif WindDir==8 && rand<MoveChance      %North-West
            %MOVES W
            if Rand<((1-P)/2) && (j>1)

    %
                if atan(((H(i,j-1)-H(i,j))*delta)/L)<(pi/12)
                                                                                    %
                    Hstar(i,j)=Hstar(i,j)-1;
                                                                                    %
                    Hstar(i,j-1)=Hstar(i,j-1)+1;
                                                                                    %
                    fprintf('NW L \n')
                                                                                    %
                end
            elseif Rand<((1-P)/2) && (j==1)
                Hstar(i,j)=Hstar(i,j)-1;
                %MOVES NW
            elseif Rand>=((1-P)/2) && Rand<((1+P)/2) && (i>1) && (j>1)

            %
                if atan(((H(i-1,j-1)-H(i,j))*delta)/L)<(pi/12)
                                                                                    %
                    Hstar(i,j)=Hstar(i,j)-1;
                                                                                    %
                    Hstar(i-1,j-1)=Hstar(i-1,j-1)+1;
                                                                                    %
                    fprintf('NW M \n')
```

```
                                                                              %
                    end
                elseif Rand>=((1-P)/2) && Rand<((1+P)/2) && ((i==1) || (j==1))
                    Hstar(i,j)=Hstar(i,j)-1;
                    %MOVES N
                elseif Rand>=((1+P)/2) && (i>1)

                      %
                    if atan(((H(i-1,j)-H(i,j))*delta)/L)<(pi/12)
                                                                              %
                        Hstar(i,j)=Hstar(i,j)-1;
                                                                              %
                        Hstar(i-1,j)=Hstar(i-1,j)+1;
                                                                              %
                        fprintf('NW R \n')
                                                                              %
                    end
                elseif Rand>=((1+P)/2) && (i==1)
                    Hstar(i,j)=Hstar(i,j)-1;           %
                end
                                                                              %
            end
                                                                              %
            movedslabs=movedslabs-1;
                                                                              %
        end
                                                                              %
        end
                                                                                 %
    end
                                                                              %
%end
                                                                               %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%replace H with Hstar and repeat for another timestep
                                                                              %
H=Hstar;
                                                                              %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
end
```

```
function [Hstar,flag]=AvalancheUPDATEflagonly(Hstar,delta,L,flag,PC)
%File AvalancheUPDATE.m
%   This subroutine will FIRST check if the difference in elevation of
%   neighboring cells is sufficiently large to prompt the cel of interest
%   (R,C) to avalanche into a nearby neighbor. SECONDLY, after determining
%   all cells which are viable to avalanche, it will compare the amount of
%   vegetation (with a parameter for "root strength") on the AVALANCHING
%   CELL to a fixed random percentage, [0,1], to determine whether or not
%   there is sufficient plant coverage to prohibit erosion (avalanching).
%   It is important that this random percentage stay fixed so that a cell
%   which had been decided as having "too much" vegetation (i.e. decided
%   that it will not avalanche) does not become viable to avalanche while
%   iterating. This routine will continue to check if the avalanching cell
%   has high enough elevation to continue avalanching and will continue
%   until there are no cells which are available to avalanche.
%
%   To run this file you will need to specify:
%     Hstar   - elevation matrix that is continually updated in main code
%       R     - the variable row location that we are considering
%       C     - the variable column location that we are considering
%       i     - the consant row location that we will refer to
%       j     - the constant column location that we will refer to
%     delta   - the thickness of each slab
%       L     - the length of each slab
%      flag   - parameter to control avalanche sequence in main code
%       PC    - weighted "percent cover" of plants WRT erosion
%
%   The routine will return Hstar after it has been updated to reflect
%   avalanching.
%

tic

flag=0;
FLAG=1;
alpha=1;
fprintf('I am running AV! ')
for R=1:size(Hstar,1)
    for C=1:size(Hstar,2)
        Erosioncheck=1;
        FLAG=1;
while FLAG==1
    RAND=rand;
    %FLAG=1;
    if Hstar(R,C)>=0
        if R<=(size(Hstar,1)-1)
            R=floor(R);
            Rp=R+1;
        else
            Rp=floor(R);
        end
        if R>=2
            R=floor(R);
            Rm=R-1;
        else
            Rm=floor(R);
        end
        if C<=(size(Hstar,2)-1)
            C=floor(C);
            Cp=C+1;
        else
            Cp=floor(C);
        end
        if C>=2
            C=floor(C);
            Cm=C-1;
        else
            Cm=floor(C);
        end
        %Rp=R+1;
        %Rm=R-1;
        %Cp=C+1;
        %Cm=C-1;
        N=[Hstar(Rp,C),Hstar(Rm,C),Hstar(R,Cp),Hstar(R,Cm)];
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %%% FOUR (4) EQUALLY SIZED NEIGHBORS THAT PROMPT AVALANCHE %%%
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        if atan(((Hstar(R,C)-N(1))*delta)/L)>=(pi/6) &&
atan(((Hstar(R,C)-N(2))*delta)/L)>=(pi/6) && atan(((Hstar(R,C)-N(3))*delta)/L)>=(pi/6)
&& atan(((Hstar(R,C)-N(4))*delta)/L)>=(pi/6)
            if (RAND<alpha*(((atan(((Hstar(R,C)-N(1))*delta)/L))/(pi/6)))*(1-PC(R,C)))
|| (RAND<alpha*(((atan(((Hstar(R,C)-N(2))*delta)/L))/(pi/6)))*(1-PC(R,C))) ||
(RAND<alpha*(((atan(((Hstar(R,C)-N(3))*delta)/L))/(pi/6)))*(1-PC(R,C))) ||
(RAND<alpha*(((atan(((Hstar(R,C)-N(4))*delta)/L))/(pi/6)))*(1-PC(R,C)))
                while (FLAG==1) && (atan(((Hstar(R,C)-N(1))*delta)/L)>=(pi/6)
&&
atan(((Hstar(R,C)-N(2))*delta)/L)>=(pi/6) && atan(((Hstar(R,C)-N(3))*delta)/L)>=(pi/6)
&& atan(((Hstar(R,C)-N(4))*delta)/L)>=(pi/6))
                    prob=rand;
                    %fprintf('4 ')
                    %%Avalanche 1%%
                    if (prob<0.25) &&
(RAND<alpha*(((atan(((Hstar(R,C)-N(1))*delta)/L))/(pi/6)))*(1-PC(R,C)))
                        Hstar(R,C)=Hstar(R,C)-1;
                        Hstar(Rp,C)=Hstar(Rp,C)+1;
                        flag=1;
                        %fprintf('Move ')
                    %%Avalanche 2%%
                    elseif (prob>=0.25 && prob<0.5) &&
(RAND<alpha*(((atan(((Hstar(R,C)-N(1))*delta)/L))/(pi/6)))*(1-PC(R,C)))
                        Hstar(R,C)=Hstar(R,C)-1;
                        Hstar(Rm,C)=Hstar(Rm,C)+1;
                        flag=1;
                        %fprintf('Move ')
                    %%Avalanche 3%%
                    elseif (prob>=0.5 && prob<0.75) &&
(RAND<alpha*(((atan(((Hstar(R,C)-N(1))*delta)/L))/(pi/6)))*(1-PC(R,C)))
                        Hstar(R,C)=Hstar(R,C)-1;
                        Hstar(R,Cp)=Hstar(R,Cp)+1;
                        flag=1;
                        %fprintf('Move ')
                    %%Avalanche 4%%
                    elseif (prob>=0.75) &&
(RAND<alpha*(((atan(((Hstar(R,C)-N(1))*delta)/L))/(pi/6)))*(1-PC(R,C)))
                        Hstar(R,C)=Hstar(R,C)-1;
                        Hstar(R,Cm)=Hstar(R,Cm)+1;
                        flag=1;
                        %fprintf('Move ')
                    else
                        FLAG=0;
                        %fprintf('No Move ')
                        break
                    end
                end
            else
                Erosioncheck=0;
                FLAG=0;
            end
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        %%% THREE (3) EQUALLY SIZED NEIGHBORS THAT PROMPT AVALANCHE %%%
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
            %%%%%%%%%
            % 1 2 3 %
            %%%%%%%%%
        elseif atan(((Hstar(R,C)-N(1))*delta)/L)>=(pi/6) &&
 atan(((Hstar(R,C)-N(2))*delta)/L)>=(pi/6) && atan(((Hstar(R,C)-N(3))*delta)/L)>=(pi/6)
            if (RAND<alpha*(((atan(((Hstar(R,C)-N(1))*delta)/L))/(pi/6)))*(1-PC(R,C)))
|| (RAND<alpha*(((atan(((Hstar(R,C)-N(1))*delta)/L))/(pi/6)))*(1-PC(R,C))) ||
(RAND<alpha*(((atan(((Hstar(R,C)-N(1))*delta)/L))/(pi/6)))*(1-PC(R,C)))
                while (FLAG==1) && (atan(((Hstar(R,C)-N(1))*delta)/L)>=(pi/6)
&&
atan(((Hstar(R,C)-N(2))*delta)/L)>=(pi/6) && atan(((Hstar(R,C)-N(3))*delta)/L)>=(pi/6))
                    prob=rand;
                    %fprintf('3.1 ')
                    %%Avalanche 1%%
                    if (prob<(1/3)) &&
(RAND<alpha*(((atan(((Hstar(R,C)-N(1))*delta)/L))/(pi/6)))*(1-PC(R,C)))
                        Hstar(R,C)=Hstar(R,C)-1;
                        Hstar(Rp,C)=Hstar(Rp,C)+1;
                        flag=1;
                        %fprintf('Move ')
                    %%Avalanche 2%%
                    elseif (prob>=(1/3) && prob<(2/3)) &&
(RAND<alpha*(((atan(((Hstar(R,C)-N(1))*delta)/L))/(pi/6)))*(1-PC(R,C)))
                        Hstar(R,C)=Hstar(R,C)-1;
                        Hstar(Rm,C)=Hstar(Rm,C)+1;
                        flag=1;
                        %fprintf('Move ')
                    %%Avalanche 3%%
                    elseif (prob>=(2/3)) &&
(RAND<alpha*(((atan(((Hstar(R,C)-N(1))*delta)/L))/(pi/6)))*(1-PC(R,C)))
                        Hstar(R,C)=Hstar(R,C)-1;
                        Hstar(R,Cp)=Hstar(R,Cp)+1;
                        flag=1;
                        %fprintf('Move ')
                    else
                        FLAG=0;
                        %fprintf('No Move ')
                        break
                    end
                end
            else
                Erosioncheck=0;
                FLAG=0;
            end
        %%%%%%%%
        % 1 2 4 %
        %%%%%%%%
        elseif atan(((Hstar(R,C)-N(1))*delta)/L)>=(pi/6) &&
atan(((Hstar(R,C)-N(2))*delta)/L)>=(pi/6) && atan(((Hstar(R,C)-N(4))*delta)/L)>=(pi/6)
            if (RAND<alpha*(((atan(((Hstar(R,C)-N(1))*delta)/L))/(pi/6)))*(1-PC(R,C)))
|| (RAND<alpha*(((atan(((Hstar(R,C)-N(1))*delta)/L))/(pi/6)))*(1-PC(R,C))) ||
(RAND<alpha*(((atan(((Hstar(R,C)-N(1))*delta)/L))/(pi/6)))*(1-PC(R,C)))
                while (FLAG==1) && (atan(((Hstar(R,C)-N(1))*delta)/L)>=(pi/6)
&&
atan(((Hstar(R,C)-N(2))*delta)/L)>=(pi/6) && atan(((Hstar(R,C)-N(4))*delta)/L)>=(pi/6))
                    prob=rand;
                    %fprintf('3.2 ')
                    %%Avalanche 1%%
                    if (prob<(1/3)) &&
(RAND<alpha*(((atan(((Hstar(R,C)-N(1))*delta)/L))/(pi/6)))*(1-PC(R,C)))
                        Hstar(R,C)=Hstar(R,C)-1;
                        Hstar(Rp,C)=Hstar(Rp,C)+1;
                        flag=1;
                        %fprintf('Move ')
                    %%Avalanche 2%%
                    elseif (prob>=(1/3) && prob<(2/3)) &&
(RAND<alpha*(((atan(((Hstar(R,C)-N(1))*delta)/L))/(pi/6)))*(1-PC(R,C)))
                        Hstar(R,C)=Hstar(R,C)-1;
                        Hstar(Rm,C)=Hstar(Rm,C)+1;
                        flag=1;
                        %fprintf('Move ')
                    %%Avalanche 4%%
                    elseif (prob>=(2/3)) &&
(RAND<alpha*(((atan(((Hstar(R,C)-N(1))*delta)/L))/(pi/6)))*(1-PC(R,C)))
                        Hstar(R,C)=Hstar(R,C)-1;
                        Hstar(R,Cm)=Hstar(R,Cm)+1;
                        flag=1;
                        %fprintf('Move ')
                    else
                        FLAG=0;
                        %fprintf('No Move ')
                        break
                    end
                end
            else
                Erosioncheck=0;
                FLAG=0;
            end
        %%%%%%%%%
        % 1 3 4 %
        %%%%%%%%%
        elseif atan(((Hstar(R,C)-N(1))*delta)/L)>=(pi/6) &&
atan(((Hstar(R,C)-N(3))*delta)/L)>=(pi/6) && atan(((Hstar(R,C)-N(4))*delta)/L)>=(pi/6)
```

```
            if (RAND<alpha*(((atan(((Hstar(R,C)-N(1))*delta)/L))/(pi/6)))*(1-PC(R,C)))
|| (RAND<alpha*(((atan(((Hstar(R,C)-N(1))*delta)/L))/(pi/6)))*(1-PC(R,C))) ||
(RAND<alpha*(((atan(((Hstar(R,C)-N(1))*delta)/L))/(pi/6)))*(1-PC(R,C)))
                while (FLAG==1) && (atan(((Hstar(R,C)-N(1))*delta)/L)>=(pi/6)
&&
atan(((Hstar(R,C)-N(3))*delta)/L)>=(pi/6) && atan(((Hstar(R,C)-N(4))*delta)/L)>=(pi/6))
                    prob=rand;
                    %fprintf('3.3 ')
                    %%Avalanche 1%%
                    if (prob<(1/3)) &&
(RAND<alpha*(((atan(((Hstar(R,C)-N(1))*delta)/L))/(pi/6)))*(1-PC(R,C)))
                        Hstar(R,C)=Hstar(R,C)-1;
                        Hstar(Rp,C)=Hstar(Rp,C)+1;
                        flag=1;
                        %fprintf('Move ')
                    %%Avalanche 3%%
                    elseif (prob>=(1/3) && prob<(2/3)) &&
(RAND<alpha*(((atan(((Hstar(R,C)-N(1))*delta)/L))/(pi/6)))*(1-PC(R,C)))
                        Hstar(R,C)=Hstar(R,C)-1;
                        Hstar(R,Cp)=Hstar(R,Cp)+1;
                        flag=1;
                        %fprintf('Move ')
                    %%Avalanche 4%%
                    elseif (prob>=(2/3)) &&
(RAND<alpha*(((atan(((Hstar(R,C)-N(1))*delta)/L))/(pi/6)))*(1-PC(R,C)))
                        Hstar(R,C)=Hstar(R,C)-1;
                        Hstar(R,Cm)=Hstar(R,Cm)+1;
                        flag=1;
                        %fprintf('Move ')
                    else
                        FLAG=0;
                        %fprintf('No Move ')
                        break
                    end
                end
            else
                Erosioncheck=0;
                FLAG=0;
            end
        %%%%%%%%
        % 2 3 4 %
        %%%%%%%%
        elseif atan(((Hstar(R,C)-N(2))*delta)/L)>=(pi/6) &&
atan(((Hstar(R,C)-N(3))*delta)/L)>=(pi/6) && atan(((Hstar(R,C)-N(4))*delta)/L)>=(pi/6)
            if (RAND<alpha*(((atan(((Hstar(R,C)-N(1))*delta)/L))/(pi/6)))*(1-PC(R,C)))
```

```
|| (RAND<alpha*(((atan(((Hstar(R,C)-N(1))*delta)/L))/(pi/6)))*(1-PC(R,C))) ||
(RAND<alpha*(((atan(((Hstar(R,C)-N(1))*delta)/L))/(pi/6)))*(1-PC(R,C)))
                while (FLAG==1) && (atan(((Hstar(R,C)-N(2))*delta)/L)>=(pi/6)
&&
atan(((Hstar(R,C)-N(3))*delta)/L)>=(pi/6) && atan(((Hstar(R,C)-N(4))*delta)/L)>=(pi/6))
                    prob=rand;
                    %fprintf('3.4 ')
                    %%Avalanche 2%%
                    if (prob<(1/3)) &&
(RAND<alpha*(((atan(((Hstar(R,C)-N(1))*delta)/L))/(pi/6)))*(1-PC(R,C)))
                        Hstar(R,C)=Hstar(R,C)-1;
                        Hstar(Rm,C)=Hstar(Rm,C)+1;
                        flag=1;
                        %fprintf('Move ')
                    %%Avalanche 3%%
                    elseif (prob>=(1/3) && prob<(2/3)) &&
(RAND<alpha*(((atan(((Hstar(R,C)-N(1))*delta)/L))/(pi/6)))*(1-PC(R,C)))
                        Hstar(R,C)=Hstar(R,C)-1;
                        Hstar(R,Cp)=Hstar(R,Cp)+1;
                        flag=1;
                        %fprintf('Move ')
                    %%Avalanche 4%%
                    elseif (prob>=(2/3)) &&
(RAND<alpha*(((atan(((Hstar(R,C)-N(1))*delta)/L))/(pi/6)))*(1-PC(R,C)))
                        Hstar(R,C)=Hstar(R,C)-1;
                        Hstar(R,Cm)=Hstar(R,Cm)+1;
                        flag=1;
                        %fprintf('Move ')
                    else
                        FLAG=0;
                        %fprintf('No Move ')
                        break
                    end
                end
            else
                Erosioncheck=0;
                FLAG=0;
            end
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %%% TWO (2) EQUALLY SIZED NEIGHBORS THAT PROMPT AVALANCHE %%%
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        %%%%%%%
        % 1 2 %
        %%%%%%%
        elseif atan(((Hstar(R,C)-N(1))*delta)/L)>=(pi/6) &&
atan(((Hstar(R,C)-N(2))*delta)/L)>=(pi/6)
            if (RAND<alpha*(((atan(((Hstar(R,C)-N(1))*delta)/L))/(pi/6)))*(1-PC(R,C)))
|| (RAND<alpha*(((atan(((Hstar(R,C)-N(1))*delta)/L))/(pi/6)))*(1-PC(R,C)))
                while (FLAG==1) && (atan(((Hstar(R,C)-N(1))*delta)/L)>=(pi/6)
&& atan(((Hstar(R,C)-N(2))*delta)/L)>=(pi/6))

                    prob=rand;
                    %fprintf('2.1 ')
                    %%Avalanche 1%%
                    if (prob<0.5) &&
(RAND<alpha*(((atan(((Hstar(R,C)-N(1))*delta)/L))/(pi/6)))*(1-PC(R,C)))
                        Hstar(R,C)=Hstar(R,C)-1;
                        Hstar(Rp,C)=Hstar(Rp,C)+1;
                        flag=1;
                        %fprintf('Move ')
                    %%Avalanche 2%%
                    elseif (prob>=0.5) &&
(RAND<alpha*(((atan(((Hstar(R,C)-N(1))*delta)/L))/(pi/6)))*(1-PC(R,C)))
                        Hstar(R,C)=Hstar(R,C)-1;
                        Hstar(Rm,C)=Hstar(Rm,C)+1;
                        flag=1;
                        %fprintf('Move ')
                    else
                        FLAG=0;
                        %fprintf('No Move ')
                        break
                    end
                end
            else
                Erosioncheck=0;
                FLAG=0;
            end
        %%%%%%
        % 1 3 %
        %%%%%%
        elseif atan(((Hstar(R,C)-N(1))*delta)/L)>=(pi/6) &&
atan(((Hstar(R,C)-N(3))*delta)/L)>=(pi/6)
            if (RAND<alpha*(((atan(((Hstar(R,C)-N(1))*delta)/L))/(pi/6)))*(1-PC(R,C)))
|| (RAND<alpha*(((atan(((Hstar(R,C)-N(1))*delta)/L))/(pi/6)))*(1-PC(R,C)))
                while (FLAG==1) && (atan(((Hstar(R,C)-N(1))*delta)/L)>=(pi/6)
&& atan(((Hstar(R,C)-N(3))*delta)/L)>=(pi/6))
                    prob=rand;
                    %fprintf('2.2 ')
                    %%Avalanche 1%%
                    if (prob<0.5) &&
(RAND<alpha*(((atan(((Hstar(R,C)-N(1))*delta)/L))/(pi/6)))*(1-PC(R,C)))
                        Hstar(R,C)=Hstar(R,C)-1;
                        Hstar(Rp,C)=Hstar(Rp,C)+1;
                        flag=1;
                        %fprintf('Move ')
                    %%Avalanche 3%%
                    elseif (prob>=0.5) &&
(RAND<alpha*(((atan(((Hstar(R,C)-N(1))*delta)/L))/(pi/6)))*(1-PC(R,C)))
                        Hstar(R,C)=Hstar(R,C)-1;
                        Hstar(R,Cp)=Hstar(R,Cp)+1;
                        flag=1;
                        %fprintf('Move ')
                    else
                        FLAG=0;
                        %fprintf('No Move ')
                        break
                    end
                end
            else
                Erosioncheck=0;
                FLAG=0;
            end
        %%%%%%
        % 1 4 %
        %%%%%%
        elseif atan(((Hstar(R,C)-N(1))*delta)/L)>=(pi/6) &&
atan(((Hstar(R,C)-N(4))*delta)/L)>=(pi/6)
            if (RAND<alpha*(((atan(((Hstar(R,C)-N(1))*delta)/L))/(pi/6)))*(1-PC(R,C)))
|| (RAND<alpha*(((atan(((Hstar(R,C)-N(1))*delta)/L))/(pi/6)))*(1-PC(R,C)))
                while (FLAG==1) && (atan(((Hstar(R,C)-N(1))*delta)/L)>=(pi/6)
&& atan(((Hstar(R,C)-N(4))*delta)/L)>=(pi/6))
                    prob=rand;
                    %fprintf('2.3 ')
                    %%Avalanche 1%%
                    if (prob<0.5) &&
(RAND<alpha*(((atan(((Hstar(R,C)-N(1))*delta)/L))/(pi/6)))*(1-PC(R,C)))
                        Hstar(R,C)=Hstar(R,C)-1;
                        Hstar(Rp,C)=Hstar(Rp,C)+1;
                        flag=1;
                        %fprintf('Move ')
                    %%Avalanche 4%%
                    elseif (prob>=0.5) &&
(RAND<alpha*(((atan(((Hstar(R,C)-N(1))*delta)/L))/(pi/6)))*(1-PC(R,C)))
                        Hstar(R,C)=Hstar(R,C)-1;
                        Hstar(R,Cm)=Hstar(R,Cm)+1;
                        flag=1;
                        %fprintf('Move ')
                    else
                        FLAG=0;
                        %fprintf('No Move ')
                        break
                    end
                end
            else
                Erosioncheck=0;
                FLAG=0;
            end
        %%%%%%%
        % 2 3 %
        %%%%%%%
        elseif atan(((Hstar(R,C)-N(2))*delta)/L)>=(pi/6) &&
atan(((Hstar(R,C)-N(3))*delta)/L)>=(pi/6)
            if (RAND<alpha*(((atan(((Hstar(R,C)-N(1))*delta)/L))/(pi/6)))*(1-PC(R,C)))
|| (RAND<alpha*(((atan(((Hstar(R,C)-N(1))*delta)/L))/(pi/6)))*(1-PC(R,C)))
                while (FLAG==1) && (atan(((Hstar(R,C)-N(2))*delta)/L)>=(pi/6)
&& atan(((Hstar(R,C)-N(3))*delta)/L)>=(pi/6))
                    prob=rand;
                    %fprintf('2.4 ')
                    %%Avalanche 2%%
                    if (prob<0.5) &&
(RAND<alpha*(((atan(((Hstar(R,C)-N(1))*delta)/L))/(pi/6)))*(1-PC(R,C)))
                        Hstar(R,C)=Hstar(R,C)-1;
                        Hstar(Rm,C)=Hstar(Rm,C)+1;
                        flag=1;
                        %fprintf('Move ')
                    %%Avalanche 3%%
                    elseif (prob>=0.5) &&
(RAND<alpha*(((atan(((Hstar(R,C)-N(1))*delta)/L))/(pi/6)))*(1-PC(R,C)))
                        Hstar(R,C)=Hstar(R,C)-1;
                        Hstar(R,Cp)=Hstar(R,Cp)+1;
                        flag=1;
                        %fprintf('Move ')
                    else
                        FLAG=0;
                        %fprintf('No Move ')
                        break
                    end
                end
            else
                Erosioncheck=0;
                FLAG=0;
            end
        %%%%%%%
        % 2 4 %
        %%%%%%%
        elseif atan(((Hstar(R,C)-N(2))*delta)/L)>=(pi/6) &&
atan(((Hstar(R,C)-N(4))*delta)/L)>=(pi/6)
            if (RAND<alpha*(((atan(((Hstar(R,C)-N(1))*delta)/L))/(pi/6)))*(1-PC(R,C)))
|| (RAND<alpha*(((atan(((Hstar(R,C)-N(1))*delta)/L))/(pi/6)))*(1-PC(R,C)))
                while (FLAG==1) && (atan(((Hstar(R,C)-N(2))*delta)/L)>=(pi/6)
&& atan(((Hstar(R,C)-N(4))*delta)/L)>=(pi/6))
                    prob=rand;
                    %fprintf('2.5 ')
                    %%Avalanche 2%%
                    if (prob<0.5) &&
(RAND<alpha*(((atan(((Hstar(R,C)-N(1))*delta)/L))/(pi/6)))*(1-PC(R,C)))
                        Hstar(R,C)=Hstar(R,C)-1;
                        Hstar(Rm,C)=Hstar(Rm,C)+1;
                        flag=1;
                        %fprintf('Move ')
                    %%Avalanche 4%%
                    elseif (prob>=0.5) &&
(RAND<alpha*(((atan(((Hstar(R,C)-N(1))*delta)/L))/(pi/6)))*(1-PC(R,C)))
                        Hstar(R,C)=Hstar(R,C)-1;
                        Hstar(R,Cm)=Hstar(R,Cm)+1;
                        flag=1;
                        %fprintf('Move ')
                    else
                        FLAG=0;
                        %fprintf('No Move ')
                        break
                    end
                end
            else
                Erosioncheck=0;
                FLAG=0;
            end
        %%%%%%%
        % 3 4 %
        %%%%%%%
        elseif atan(((Hstar(R,C)-N(3))*delta)/L)>=(pi/6) &&
atan(((Hstar(R,C)-N(4))*delta)/L)>=(pi/6)
            if (RAND<alpha*(((atan(((Hstar(R,C)-N(1))*delta)/L))/(pi/6)))*(1-PC(R,C)))
|| (RAND<alpha*(((atan(((Hstar(R,C)-N(1))*delta)/L))/(pi/6)))*(1-PC(R,C)))
                while (FLAG==1) && (atan(((Hstar(R,C)-N(3))*delta)/L)>=(pi/6)
&& atan(((Hstar(R,C)-N(4))*delta)/L)>=(pi/6))
                    prob=rand;
                    %fprintf('2.6 ')
                    %%Avalanche 3%%
                    if (prob<0.5) &&
(RAND<alpha*(((atan(((Hstar(R,C)-N(1))*delta)/L))/(pi/6)))*(1-PC(R,C)))
                        Hstar(R,C)=Hstar(R,C)-1;
                        Hstar(R,Cp)=Hstar(R,Cp)+1;
                        flag=1;
                        %fprintf('Move ')
                    %%Avalanche 4%%
                    elseif (prob>=0.5) &&
(RAND<alpha*(((atan(((Hstar(R,C)-N(1))*delta)/L))/(pi/6)))*(1-PC(R,C)))
                        Hstar(R,C)=Hstar(R,C)-1;
                        Hstar(R,Cm)=Hstar(R,Cm)+1;
                        flag=1;
                        %fprintf('Move ')
                    else
                        FLAG=0;
                        %fprintf('No Move ')
                        break
                    end
                end
            else
                Erosioncheck=0;
                FLAG=0;
            end
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %%% ONE (1) NEIGHBOR THAT PROMPTS AVALANCHE %%%
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        %%%%%
        % 1 %
        %%%%%
        elseif atan(((Hstar(R,C)-N(1))*delta)/L)>=(pi/6)
            if RAND<alpha*(((atan(((Hstar(R,C)-N(1))*delta)/L))/(pi/6)))*(1-PC(R,C))
                while atan(((Hstar(R,C)-N(1))*delta)/L)>=(pi/6)
                    %fprintf('1.1 ')
                    Hstar(R,C)=Hstar(R,C)-1;
                    Hstar(Rp,C)=Hstar(Rp,C)+1;
                    flag=1;
                    %fprintf('Move ')
                end
            else
                Erosioncheck=0;
                FLAG=0;
            end
        %%%%%
        % 2 %
        %%%%%
        elseif atan(((Hstar(R,C)-N(2))*delta)/L)>=(pi/6)
            if RAND<alpha*(((atan(((Hstar(R,C)-N(1))*delta)/L))/(pi/6)))*(1-PC(R,C))
                while atan(((Hstar(R,C)-N(2))*delta)/L)>=(pi/6)
                    %fprintf(’1.2 ’)
                    Hstar(R,C)=Hstar(R,C)-1;
                    Hstar(Rm,C)=Hstar(Rm,C)+1;
                    flag=1;
                    %fprintf(’Move ’)
                end
            else
                Erosioncheck=0;
                FLAG=0;
            end
        %%%%%
        % 3 %
        %%%%%
        elseif atan(((Hstar(R,C)-N(3))*delta)/L)>=(pi/6)
            if RAND<alpha*(((atan(((Hstar(R,C)-N(1))*delta)/L))/(pi/6)))*(1-PC(R,C))
                while atan(((Hstar(R,C)-N(3))*delta)/L)>=(pi/6)
                    %fprintf(’1.3 ’)
                    Hstar(R,C)=Hstar(R,C)-1;
                    Hstar(R,Cp)=Hstar(R,Cp)+1;
                    flag=1;
                    %fprintf(’Move ’)
                end
            else
                Erosioncheck=0;
                FLAG=0;
            end
        %%%%%
        % 4 %
        %%%%%
        elseif atan(((Hstar(R,C)-N(4))*delta)/L)>=(pi/6)
            if RAND<alpha*(((atan(((Hstar(R,C)-N(1))*delta)/L))/(pi/6)))*(1-PC(R,C))
                while atan(((Hstar(R,C)-N(4))*delta)/L)>=(pi/6)
                    %fprintf(’1.4 ’)
                    Hstar(R,C)=Hstar(R,C)-1;
                    Hstar(R,Cm)=Hstar(R,Cm)+1;
                    flag=1;
                    %fprintf(’Move ’)
                end
```

```
            else
                Erosioncheck=0;
                FLAG=0;
            end
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %%% ZERO (0) NEIGHBORS THAT PROMPT AVALANCHE %%%
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        else
            Erosioncheck=0;
            FLAG=0;
        end
    else
        Erosioncheck=0;
        FLAG=0;
    end
end
    end
end


toc
end
```

```
function [Hstar,PlantColumnArray,PlantColumnArray2,P3,flag]=
MarineProcessesSLOPE(Hstar,delta,L,flag,PC,P3)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%% MARINE PROCESSES %%%%%
                                                                          %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%
                                                                          %

                                                                          %
%   This subroutine will consider the periodicity of oceanic waves and how
                                                                          %
%   they can change the landscape of coastal islands. We will construct an
                                                                          %
%   "Equilibrium profile" which is a smooth surface maintained by the runup
                                                                          %
%   and swash of these waves.
                                                                          %
%
                                                                          %
%   To run this file, you will need to specify:
                                                                          %
%      Hstar  - elevation matrix that will be updated
                                                                          %
%      delta  - the thickness of each slab
                                                                          %
%        L    - the length of each slab
                                                                          %
%
                                                                          %
%   The subroutine will then return Hstar after it has been updated
                                                                          %
%   according to the marine processes.
                                                                          %
%
                                                                          %

                                                                          %
Rv=0;       %resistance to erosion due to vegetation
                                                                          %
We=1;       %maximum erosive strength of waves
                                                                          %
Wdiss=0;    %cumulative dissipation factor (pg. 888 in Silva)
                                                                          %
```

```
Pinun=0;    %probability of bed level change regardless of waves
                                                                    %
S=1;        %stochasticity term representing unaccounted conditions (e.g., grain
size variability)                                                %

                                                                    %
%H0=1;       %wave height    USABLE DEFINITIONS IN KEISJERS PAPER
                                                                    %
%L0=1;       %wave length
                                                                    %
%Fenergy=1;      %factor for hydrodynamic energy
                                                                    %
Li=size(Hstar,1);
Wi=size(Hstar,2);%
AdjascentLength=zeros([Li 1]);
                                                                    %
OppositeLocation=zeros([Li 1]);
                                                                    %
ColumnArray=zeros([Li 1]);
PlantColumnArray=zeros([Li 1]);%
PlantColumnArray2=zeros([Li 1]);
LongTransect=zeros([Wi 1]);
beta=zeros([Li 1]);
FLAG=0;
                                                                   %

                                                                    %
fprintf('I am running MP! ')


                                                                    %%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Find the slope and R2% to help determine the Equilibrium Profile %
                                                                    %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
                                                                    %

                                                                    %
% First, we find all of the rows which have shoreline and record the column
                                                                    %
% in which we find the shoreline. We store this is ColumnArray.
                                                                    %
for i=1:size(Hstar,1)
                                                                    %
```

```
    for j=size(Hstar,2):-1:1
                                                                          %
        if Hstar(i,j)>=0 && ColumnArray(i)==0 %&& Hstar(i,j-1)>=0
                                                                          %
            ColumnArray(i)=j;
                                                                          %
            break;
                                                                          %
        end
                                                                          %
    end
                                                                          %
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Now we will force a new Array to keep track of Morella!                 %
%for i=1:size(Hstar,1)                                                     %
%    for j=size(Hstar,2):-1:1                                              %
%        if P3(i,j,1)>0 && PlantColumnArray(i)==0 && Hstar(i,j)>=0         %
%            PlantColumnArray(i)=j;                                        %
%        end                                                               %
%    end                                                                   %
%end                                                                       %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%% I AM ATTEMPTING TO LET PLANTCOLUMNARRAY = COLUMNARRAY
%%% THEN I WILL TRY TO FORCE PLANTCOLUMNARRAY2 NEVER TO EXTEND ABOVE 2.5M

for i=1:size(Hstar,1)
    PlantColumnArray(i)=ColumnArray(i);
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% NEW AND ALSO PROBLEMATIC AREA FOR CODE %%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%Create a 2nd column Array for plants!
FLAG=0;
for i=1:size(Hstar,1)
    LongTransect=Hstar(i,:,1);
    if PlantColumnArray(i)>0
        while PlantColumnArray2(i)==0
            for j=(PlantColumnArray(i)):-1:1
        if PlantColumnArray2(i)>0
            break
        end
    %if PlantColumnArray2(i)==0 && PlantColumnArray(i)>0
%%%while P3(i,j,1)>0 && FLAG==0
    %for j=PlantColumnArray(i):-1:1
    %%%while P3(i,j,1)>0 && FLAG==0
    %while PlantColumnArray2(i)==0



        % (1)Case where max elevation for all columns in a fixed
        % row is LESS than 2.5m

        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        %NEED TO KNOW:
        % does max return index or does max return actual entry?
        % Should find a 'workaround' to ensure this case does not
        % look beyond the foredune
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        if ((max(max(LongTransect)))*delta)<2.5
            for k=(PlantColumnArray(i)-1):-1:1
                if P3(i,k,1)>0 && P3(i,k-1,1)<=0 && PlantColumnArray2(i)==0
                    PlantColumnArray2(i)=k;
                    break
                end
            end
            if PlantColumnArray2(i)==0
                PlantColumnArray2(i)=PlantColumnArray(i);
            end


            %NEWEST ADDITION!!
        % (2) Case where max elevation for all columns in a fixed
        % row EQUALS 2.5m
        elseif ((max(max(LongTransect)))*delta)==2.5
            for k=(PlantColumnArray(i)-1):-1:1
                if (Hstar(i,k,1)*delta)==2.5 && (PlantColumnArray2(i)==0)
                    PlantColumnArray2(i)=k;
                end
            end
            %%%%%%%%%%%%%%%%%%%%%

        % (3) Case where max elevation for all columns in a fixed
        % row is greater than 2.5m
```

```
                elseif ((max(max(LongTransect)))*delta)>2.5
                    for k=(PlantColumnArray(i)-1):-1:1
                        if (Hstar(i,j,1)*delta)>2.5 && PlantColumnArray2(i)==0
                            PlantColumnArray2(i)=j;
                        end
                    end
                end
            end
        end
    end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


                                                                        %
%Now, we reset our column variable, j=0, and then for each row, i, we fix j
                                                                        %
%to equal the corresponding column in the matrix, stored in ColumnArray
                                                                        %
%above. Then, we find the value, c, for which H(i,j-c) is above 2.5 meters.
                                                                        %
for i=1:size(Hstar,1)
                                                                        %
    if ColumnArray(i)>0
                                                                        %
        j=ColumnArray(i);
                                                                        %
    else
                                                                        %
        j=-1;
                                                                        %
    end
                                                                        %
    if ColumnArray(i)>0 && j>0 %If there exists a coastline%
                                                                        %
        %While the height is BELOW (or equal to) the threshold...
                                                                        %
        c=j;
                                                                        %
        while ((Hstar(i,c+1))*delta)<=2.5 && (c)>0    %(should be 2.5)%
                                                                        %
            %increment the value of c and continue to update our new arrays
                                                                        %
            for c=j-1:-1:0
```

```
                                                                             %
                 AdjascentLength(i)=(j-c)*L; %distance from j to c is j-c
                                                                             %
                 OppositeLocation(i)=c; %column c is what we want
                                                                             %
                     %Seems to be messing up here (mostly 0s or 1s)
                                                                             %
                     %might need to make delta bigger
                                                                             %
                     %(currently seeking difference of 10 slabs)
                     if ((Hstar(i,c+1))*delta)>2.5
                         break
                     end
             end
         end
         %for j=ColumnArray(i):-1:1
         %    if ((Hstar(i,j))*delta)>2.5 && NewArray(i)==0
         %        NewArray(i)=j;
         %        %break
         %    end
         %    %break
         %end
     end
                                                                             %
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Repeat the loops above, but for Morella                                       %
%                                                                               %
%for i=1:size(Hstar,1)                                                          %
                                                                                %
%    if PlantColumnArray(i)>0                                                   %

  %
%        j=PlantColumnArray(i);                                                 %

  %
%    else                                                                       %
                                                                                %
%        j=-1;                                                                  %
                                                                                %
%    end                                                                        %
%    while PlantColumnArray(i)>0 && j>0 %if Morella exists in this row...       %
%        for c=j:-1:1                                                           %
%            if P3(i,c,1)>0                                                     %
%               P3(i,c,1)=0;                                            %
%               P3(i,c,2)=0;                                            %
%           elseif P3(i,c,1)==0
%               j=0;
%               break
%           end                                                         %
%       end                                                             %
%   end                                                                 %
%end                                                                    %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%for i=size(Hstar,1


for i=1:size(Hstar,1)
    for j=PlantColumnArray(i):-1:PlantColumnArray2(i)
        if j>0
            P3(i,j,1)=0;
            P3(i,j,2)=0;
        end
    end
end


                                                                      %
for i=1:size(Hstar,1)
                                                                      %
    if OppositeLocation(i)>0
                                                                      %
        %equilibrium foreshore slope
                                                                      %
        beta(i)=atan((Hstar(i,OppositeLocation(i)))/AdjascentLength(i));
                                                                      %
    else
                                                                      %
        beta(i)=0;
                                                                      %
        OppositeLocation(i)=0;
                                                                      %
        %c=0;
                                                                      %
    end
                                                                      %
end
                                                                      %
```

```
                                                                          %

                                                                          %


                                                                          %
for i=1:size(Hstar,1)
                                                                          %
    if OppositeLocation(i)>0 && ColumnArray(i)>0
                                                                          %
        for k=AdjascentLength(i):-1:-5  % <-- it moves 20 cells into the ocean!
                                                                         %
            Hstar(i,ColumnArray(i)-k)=
round(Hstar(i,OppositeLocation(i))
-((Hstar(i,OppositeLocation(i))/AdjascentLength(i))*(AdjascentLength(i)-k)));
        end
                                                                          %
    end
                                                                          %
end
                                                                          %
%[Hstar,flag]=AvalancheUPDATE(Hstar,delta,L,flag,PC);
                                                                           %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
end
```

```
function [Hstar,ColumnArraySwamp1,ColumnArraySwamp2,AdjascentLengthSwamp,
OppositeLocationSwamp]=SwampProcesses(Hstar,delta,L,flag,PC)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%% SWAMP PROCESSES %%%%%
                                                              %
%%%%%%%%%%%%%%%%%%%%%%%%
                                                              %

                                                             %
%   This subroutine will consider the process by which the marshy region of
%   the island is formed and maintained. The goal is to "declare" that
%   low elevation portions of the inland side of the island as the marsh.
%   We will make an additional layer (k=0 or k=1) in our elevation matrix
%   where H(i,j,2)=0 indicates that there is NO marsh present on the ith
%   row and jth column (i.e. H(i,j,1) is still viable to be polled to move
%   during Aeolian Transport) and H(i,j,2)=2 indicates that H(i,j) is
%   considered to be "in" the marsh (i.e. H(i,j,1) is no longer viable to
%   relinquish any slabs during Aeolian Transport). The process performs by
%   polling each row and storing the western-most cell of positive
%   elevation into ColumnArraySwamp1 to create a western "boundary" on the
%   swamp. Next, the process will poll each row
%   (beginning with the corresponding column stored in ColumnArraySwamp1)to
%   find the easternmost cell whose elevation indicates that it is still a
%   part of our marsh (i.e. H(i,j,1)*delta<=1). Let us say that this
%   determined cell occures at (i,j). With a small probability (roughly 20%
%   probability), cell H(i,j+1,1) will take its topmost slab and deposit it
%   at H(i,ColumnArraySwamp1(i)-1). The goal is to create a gradual western
%   migration of the island as well as a gradual growth of the marsh
%   (especially in proportion to the rest of the island).
%
%   What's more, the northern and southern "tips" of the island have
%   irregular behavior, when compared to the coastline and the interior of
%   the island. We hope to replicate the procedure described above on the
%   "tips" of the island, but without declaring these regions as marsh. The
%   difficulty will be that the tips are not necessarily excluded from
%   being consumed by the marsh region but similarly, the goal is not to
%   necessarily replace them with the marsh.
%
                                                             %
%   To run this file, you will need to specify:
                                                             %
```

```
%       Hstar  - elevation matrix that will be updated
                                                                         %
%       delta  - the thickness of each slab
                                                                         %
%         L    - the length of each slab
%        flag  - boolean variable to indicate the need to avalanche
%
                                                                         %
%    The subroutine will then return Hstar after it has been updated
                                                                         %
%    according to the marine processes.
                                                                         %
%
                                                                         %

                                                                         %
Rv=0;       %resistance to erosion due to vegetation
                                                                         %
We=1;       %maximum erosive strength of waves
                                                                         %
Wdiss=0;    %cumulative dissipation factor (pg. 888 in Silva)
                                                                         %
Pinun=0;    %probability of bed level change regardless of waves
                                                                         %
S=1;        %stochasticity term representing unaccounted conditions (e.g., grain
size variability)                                                     %

                                                                         %
%H0=1;       %wave height    USABLE DEFINITIONS IN KEISJERS PAPER
                                                                         %
%L0=1;       %wave length
                                                                         %
%Fenergy=1;      %factor for hydrodynamic energy
                                                                         %
Li=size(Hstar,1);
                                                                         %
AdjascentLengthSwamp=zeros([Li 1]);
                                                                         %
OppositeLocationSwamp=zeros([Li 1]);
                                                                         %
ColumnArraySwamp1=zeros([Li 1]);
                                                                         %
ColumnArraySwamp2=zeros([Li 1]);
                                                                         %
%FLAG=0;
```

```
                                                                           %
fprintf('I am running SP ')
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Declare Marsh Region & Poll Eastern Boundary Cells to be Deposited at Western
Boundary %                                                                %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
                                                              %

                                                                           %
% First, we find all of the rows which have shoreline and record the column
                                                                           %
% in which we find the shoreline. We store this is ColumnArraySwamp1.
                                                                           %
for i=1:size(Hstar,1)
                                                                           %
    for j=1:size(Hstar,2)
                                                                           %
        if Hstar(i,j)>=0 && ColumnArraySwamp1(i)==0 %&& Hstar(i,j-1)>=0
                                                                           %
            ColumnArraySwamp1(i)=j;
                                                                           %
            break;
                                                                           %
        end
                                                                           %
    end
                                                                           %
end
                                                                           %

%Next, we find the first cell, which is to the east of
%H(i,ColumArraySwamp1(i)) for each relevant value for i. We store this in
%ColumArraySwamp2.

for i=1:size(Hstar,1)
    if ColumnArraySwamp1(i)~=0
        for j=ColumnArraySwamp1(i):size(Hstar,2)
            if (Hstar(i,j)*delta)>=1 && ColumnArraySwamp2(i)==0
                ColumnArraySwamp2(i)=j;
                break;
            end
        end
    end
end

%Now, we have an eastern boundary and a western boundary for our marsh for
%each row on the island. We must be careful in the next step! Because of
%the nature of the tips of the islands, we must acknowledge that is it
%possible that for some value of i, ColumnArraySwamp1(i)~=0 but
%ColumnArraySwamp2(i)=0! We must be careful to avoid computation error.

%Next, we will declare which region is our "initial" swamp region.

for i=1:size(Hstar,1)
    if ColumnArraySwamp2(i)~=0
        for j=ColumnArraySwamp1(i):ColumnArraySwamp2(i)
            Hstar(i,j,2)=2;
        end
    end
end

%Next, we will look at Hstar(i,ColumnArraySwamp2(i)+1) (the portion of the
%island which is NOT in the marsh but is directly adjacent to the marsh)
%and give it a 20% probability to be deposited at
%Hstar(i,ColumnArraySwamp1(i)-1) (the first portion of the island which has
%a non-positive elevation that is directly adjascent to the marsh).
%This will produce a "flattening" of the marsh which will intentionally
%migrate westward. It will also potentially produce eastward migration of
%the marsh (due to local elevation changes).

for i=1:size(Hstar,1)
    if ColumnArraySwamp2(i)~=0 && ColumnArraySwamp1(i)>1
        if (ColumnArraySwamp1(i)-1)>=1
            if rand<=0.002
                Hstar(i,ColumnArraySwamp1(i)-1)=Hstar(i,ColumnArraySwamp1(i)-1)+1;
                Hstar(i,ColumnArraySwamp2(i)+1)=Hstar(i,ColumnArraySwamp2(i)+1)-1;
                flag=1;
            end
        elseif (ColumnArraySwamp(i)-1)<=0
            Hstar(i,ColumnArraySwamp2(i)+1)=Hstar(i,ColumnArraySwamp2(i)+1)-1;
        end
    end
end

%Lastly, we should redeclare which cells are considered to be "in" the
%swamp, provided that flag==1, since the topography has changed.
%Should we run Avalanche first???? (I think yes)
```

```
if flag==1
    [Hstar,flag]=AvalancheUPDATEflagonly(Hstar,delta,L,flag,PC);
end

for i=1:size(Hstar,1)
    if ColumnArraySwamp2(i)~=0 && flag==1
        for j=ColumnArraySwamp1(i)-1:ColumnArraySwamp2(i)+1
            if j>0 && (Hstar(i,j,1)*delta)>=-0.5 && (Hstar(i,j,1)*delta)<=1
                Hstar(i,j,2)=2;
            end
        end
    end
end

%%%%% IT REMAINS TO PROGRAM %%%%

%Behavior of the north/south tips of island!
%Incorporating the presence of the marsh in Aeolian Transport!
%Other things???


                                                                      %
%[Hstar,flag]=AvalancheUPDATE(Hstar,delta,L,flag,PC);
                                                                           %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
end
```

```
function [Hstar,P1,P2,P3,P4]=LandCategorization(Hstar,P1,P2,P3,P4,
PlantColumnArray,PlantColumnArray2,ColumnArraySwamp1,ColumnArraySwamp2)
    ROW=size(Hstar,1);
    COLUMN=size(Hstar,2);
    %%% First, we will reset all of Hstar(:,:,2)=0, since it is much more
    %%% difficult to "forcibly" identify the dune fields.
    Hstar(:,:,2)=zeros(ROW,COLUMN);

    %%% Next, we will use portions of MarineProcesses to identify which
    %%% portions of the island will be categorized as "beach"
    %%% Hstar(i,j,2)=1
    %%% and portions of SwampProcesses to identify which portions of the
    %%% island will be categorized as "marsh" Hstar(i,j,2)=2.


    %%%% MARINE PROCESSES %%%%
    % Here, I am making sure that Marine Processes returns PlantColumnArray
    % and PlantColumnArray2, hopefully allowing us to avoid relocating the
    % important cells.
    for i=1:ROW
        for j=PlantColumnArray(i):-1:PlantColumnArray2(i)
            if j>0
                Hstar(i,j,2)=1;
            end
        end
    end

    %%%% SWAMP PROCESSES %%%%
    % Here, I am making sure that SwampProcesses returns ColumnArraySwamp1
    % and ColumnArraySwamp2, hopefully allowing us to avoid relocating the
    % important cells (similar to above).
    for i=1:ROW
        if ColumnArraySwamp2(i)~=0
            for j=ColumnArraySwamp1(i):ColumnArraySwamp2(i)
                Hstar(i,j,2)=2;
            end
        end
    end


    %%% Lastly, we redeclare which portions of the island are underwater.
   for i=1:ROW
       for j=1:COLUMN
```

```
            if Hstar(i,j,1)<0
                Hstar(i,j,2)=-1;
            end
        end
    end

end
```