2021

# Modeling Vegetation Effects on Barrier Island Evolution

Eric W. Schoen
*Virginia Commonwealth University*

# Modeling Vegetation Effects on Barrier Island Evolution With Sea Level Rise

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

by

**Eric Schoen**
Master of Science

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

Virginia Commonwealth University
Richmond, Virginia
April 2021

# Acknowledgements

I would like to thank my parents and brother, who have shown endless patience and support throughout my life while managing to imprint upon me an appreciation for the arts, a love of science, and a desire to see more and learn more about all things.

# Table of Contents

# List of Figures

# List of Tables

# Abstract

Barrier islands play a significant role in protecting coastlines and harboring coastal habitats. In an effort to study and better understand the evolution of barrier island systems, a cellular model capturing various meteorological and environmental processes is proposed. Erosion due to wind, gravity, and marine processes are coupled with plant population effects. We demonstrate the inhibition of plant cover on sediment mobility, island migration, and erosion in the presence of sea level rise.

# Chapter 1

# Introduction

Barrier islands are chains of land masses that form offshore from many coastal regions around the world, and often serve as a defensive formation against the impact of adverse weather systems [26]. These formations play a critical role in defending mainland shorelines and protecting inhabitants from storm surge and erosion, and additionally act as shelter protecting nearby habitats in the marshes and estuaries, many of which support fishing economies [15]. As home to diverse biological ecosystems, it is all the more critical that we understand the processes which govern their evolution. Barrier island geomorphology is affected by climate change, which may increase the duration and frequency of storms and escalate the rate of sea level rise [5]. A model for predicting the evolution of barrier islands in response to changing climatic conditions and local plant ecology is critical in aiding policy makers in regulating coastal regions.

The evolution of barrier island geography is largely dependent upon the erosive effects of wind and wave activity [12]. Stable barrier islands migrate landward as sea level rises [3]. Island migration is a response to long-term processes like regular tidal dynamics and wind erosion, coupled with more dramatic overwashing events where large amounts of sediment are moved from the foredune onto the backbarrier marsh portions of the island [16]. When keeping insufficient pace with sea level rise, overwashing and tidal deposition can result in the flattening and eventual drowning of the the island [21]. The likelihood of overwashing events is closely tied to island characteristics like transectional width and maximum dune height [19].

The interactions between vegetation and sediment transport are critical elements governing the dynamical evolution of coastal dune landscapes [1]. There is a positive correlation between plant density and sediment retention [11]. Plants are sensitive to changing island topography, physical impacts of waves, groundwater, nutrients and exposure to sea spray [30]. Furthermore, overwash events may reduce, or potentially eliminate, vegetative cover by exposing plants to lower elevations, or by destroying protective dune barriers, subjecting the plants to the stresses of saltwater flooding and sand burial [2]. Loss of dune cover, and subsequently the loss of dune-building plant life, have long term effects on the elevation of the island.

Multiple models have been created to demonstrate a variety of evolutionary behaviors exhibited by barrier islands. In 1995, B.T. Werner introduced a cellular model, where dunes are constructed with slabs of sediment and the elevation taken to be proportional to the number of slabs present at any location in the domain. Slabs of sediment are then transported about the domain, subject to natural erosive properties [29]. The algorithm in the model successfully recreates the effect of wind erosion and deposition, or "aeolian trasportation," of sediment. Werner's model omits any effects of vegetation of sediment transport, and deposition is dependent only on wind speed and the angle of repose between neighboring sediment slabs. The model also includes the gravitational response of sediment collapse from higher elevations to lower elevations, termed "avalanching".

The ISLAND model developed by E.B. Rastetter in 1991 incorporated plant populations with annual changes in vegetation, geomorphology, water table depth and groundwater salinity on cross-sectional transects of barrier islands [23]. Plant development is considered in life stages by repeatedly calculating the probability of successfully progressing to the next stage in order to determine overall survival. Plants are divided by classification as grasses, annuals, and perennials to establish life stage conditions and duration.

In 2002, Andres Baas proposed a cellular dune landscaping model, DECAL, which incorporated vegetation into the algorithm, capturing the richer dynamics brought about by the interaction between vegetation and sand transport processes [1]. This model reproduced many elements from Werner's work, including wind erosion and avalanching processes. The work's greatest achievement lies in successfully incorporating plant populations into a cellular domain and establishing dune formation dependence upon the existence of vegetation. The model does not incorporate any marine processes, sea level rise, or other beach profile dynamics.

Keijsers, DeGroot, and Riksen presented the DUBEVEG model in 2016. The cellular automaton incorporates three primary components: aeolian transportation, living plant populations, and the effects of regular marine processes. The DUBEVEG model did not encompass the entire island domain, only extending as far as the initial foredune. Additionally, the effect of wind transportation was of limited practical application as the model employed only unidirectional wind forces at constant rates [12].

Many two-dimensional models have sought to capture the dynamical behavior of shoreline slopes on a single transect of the island (see [13], [14], [10], [7], [6]). These models track cross-shore evolution and primarily focus on the active shoreface region. The basis of these models is typically some modification or extension upon the "Bruun Rule." In 1962, Bruun proposed a relationship between sea-level rise and shoreline recession based on the profile of the beach [4]. Maurice Schwartz tested this relationship is laboratory settings before giving it the eponymous moniker it is known by today in his 1964 publication [25].

Two dimensional barrier island models typically only consider the shore profile and do not extend into the subaerial portions of the island beyond the initial dune ridge. In 2010, Rosati et al. developed the 2DMCO model, a cross-shore model that that is situated over top of a compressible substrate, resulting in the eventual inundation of the island with the progression of landward migration [24]. The BRIE model given by Nienhuis et al. in 2019 details alongshore sediment transport, as opposed to limiting such transport to cross-shore fluxes, as well as inlet dynamics and flood tidal delta depositions [19]. An algorithm outlined by Lorenzo-Trueba et al. in 2014 presented four different responses to sediment fluxes: height drowning, width drowning, constant landward retreat, and periodic retreat [13]. These examples treat overwash of sediment as the primary driver of island migration, and assume little or no activity in the absence of storm events.

We present a comprehensive model capturing geomorphology via meteorological processes while accounting for living plant populations. The weathering processes in our model were partly inspired by the DECAL [1] and DUBEVEG [12] models proposed by Nield and Baas and Keijsers et al., respectively. We include four species of plants and examine their effect on island evolution under various sea level rise scenarios.

# Chapter 2

# The Model

The primary model framework is an array, $H$, composed of numbers of slabs of sediment. Each block of sediment has dimensions $\delta \times L \times L$, where $\delta$ is the thickness or height of the slab, and $L$ is the width and length of each slab. The island and surrounding region is discretized at locations $(i, j)$, for $i = 1, ..., n$ and $j = 1, ..., m$. The elevation of the landscape at $(i, j)$ is given by $\delta H(i, j)$, where $H(i, j)$ is the number of slabs of sediment at location $(i, j)$ relative to sea level. Additional arrays, $P_k(i, j)$ for $k \in \{1, 2, 3, 4\}$, represent the population density of plant species $k$ at cell $(i, j)$.

Sand and other sediments that make up the subaerial portion of the island shift in response to wind erosion and deposition of sediment, natural gravitational collapse, and landward migration due to marine processes and sea level rise. The presence of plants impede the movement of sediment. The likelihood of each slab of sediment shifting or eroding is inversely proportional to the vegetation cover.

A flow chart for the model procedure is given in Figure 2.1 and explained in further detail in the following sections.

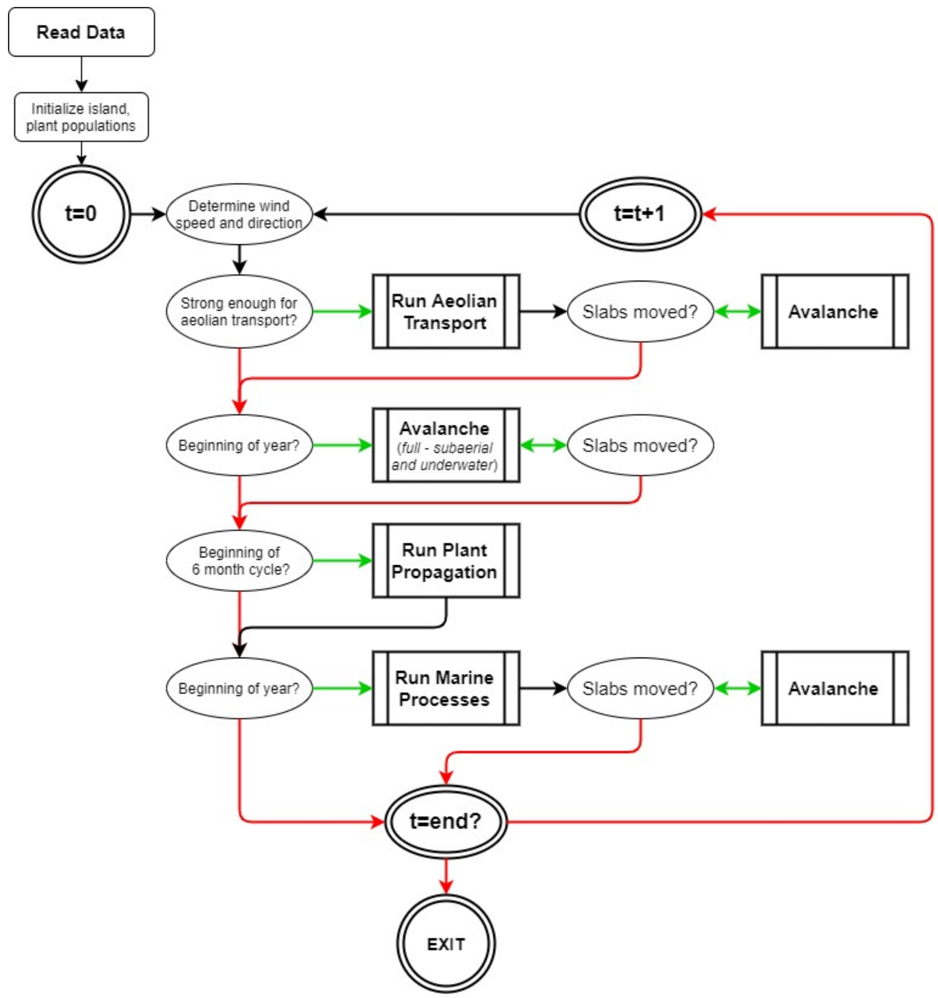



Figure 2.1: Flow chart of the model pathway.

## 2.1 Plant Populations

The model includes four species of plants native to Virginia's barrier islands: *Ammophila breviligulata*, *Spartina patens*, *Morella cerifera*, and *Spartina alterniflora*. *Ammophila breviligulata* is the primary dune-building grass. Both *Spartina* species are marsh grasses, but *Spartina alterniflora* grows exclusively in the marsh while *Spartina patens* can populate much higher elevations as well. *Morella cerifera* is a woody shrub. Plant populations inhabit elevation ranges given in Table (2.1) and are limited only by their respective growth and death rates, elevation tolerance, and ability to compete for space with neighboring plant species.

| **Species** | $P_k$ | $\lambda_{L,k}$ | $\lambda_{H,k}$ |
|---|---|---|---|
| Ammophila breviligulata | $P_1$ | 1 m | 5m |
| Spartina patens | $P_2$ | 0.75m | 3m |
| Morella cerifera | $P_3$ | 1.5m | 2.5m |
| Spartina alterniflora | $P_4$ | −0.5m | 1m |

Table 2.1: Elevation ranges of each plant species.

For a species $k$, with $k \in \{1, 2, 3, 4\}$, the population density of each cell, given by $P_k(i, j) \in [0, 1]$, can vary seasonally. Growth is determined for each species $k$ on cell $(i, j)$ by selecting a growth parameter, $\gamma_k(i, j, t)$, uniformly on the interval $[g, G]$, that may include both positive and negative values such that $\gamma_k(i, j, t) < 0$ indicates plant death, $\gamma_k(i, j, t) > 0$ indicates plant growth, and $\gamma_k(i, j, t) = 0$ has no effect on the current population density of the cell. Different ranges of parameter values capture the behaviors of growth and death associated with different seasons, as indicated by the time dependence $\gamma_k(i, j, t)$.

Propagation of plants into new cellular regions is handled by polling the plant populations of neighboring cells. For a species $k$ at location $(i, j)$ with population $P_k(i, j)$ we define $\Omega_{k,n}$ for $n = 1, 2, \ldots 8$, to be the Moore neighborhood (8-cell) adjacent to cell $(i, j)$. Propagation into the current cell is considered to be at the same growth rate, $\gamma_k(i, j, t)$, as is applied to growth within the cell. The new population density, $P'_k$, is given by

$$P'_k = P_k\big(1 + \gamma_k(i, j, t)\big) + \sum_{n=1}^{8} \gamma_k(i, j, t)\Omega_{k,n}. \tag{2.1}$$

Note that if there is no species population on the current cell (i.e. $P_k(i, j) = 0$), then equation (2.1) represents the spread of vegetation from populations of neighboring cells into new territory at cell $(i, j)$ when $\sum_{n=1,2,\ldots 8} \Omega_{k,n} \neq 0$. In the case that $P_k(i, j) \neq 0$, it is assumed that proximity to same-species populations creates stronger growth within a cell if $\gamma_k(i,j,t) > 0$. In effect, this represents better protection for communities of plants surrounded by the same species of plant. These populations would naturally experience higher rates of development than those that stood alone, or those with fewer neighboring same-species populations. The effect is similar in cases of $\gamma_k(i,j,t) < 0$, as proximity to dying plants is likely to effect same species plants. Provided at least one cell in the eight cell neighborhood currently hosts the species, equation (2.1) accounts for propagation into cells that were previously unpopulated, as well as growth within a cell that already has an established population. All cells that lie with the plants viable elevation range given in Table 2.1 are susceptible to growth.

The woody shrub *Morella* has an additional propagation condition. The seeds of this plant species are spread by birds locally. This is accounted for by allotting each cell within the viable growing range a small probability of a population being established at random, without requiring the existence of neighboring populations of *Morella*.

Each plant type is defined by a maximum percent cover value, $\eta_k$. We use these factors to modify the new population density, $P'_k$ as given in equation (2.1) to

$$P'_k = \min\left(\eta_k,\ P_k\big(1+\gamma_k(i,j,t)\big) + \sum_{n=1}^{8} \gamma_k(i,j,t)\Omega_{k,n}\right), \tag{2.2}$$

where we ensure growth does not exceed the maximum percent cover, $\eta_k$, for species $k$.

If shifting sediment causes a cell with a plant population to fall outside of the viable elevation range, (i.e. $\delta \cdot H(i,j) < \lambda_{L,k}$ or $\delta \cdot H(i,j) > \lambda_{H,k}$ for the values given in Table 2.1), then the plant population begins to die at some fixed rate. We define $\beta$ to be the death by elevation rate, and for every time step outside of the viable range for a population $P_k(i,j)$, the plant density is diminished by

$$P'_k = P_k - \beta P_k.$$

Multiple species of plants may cohabit the same cell. We define a global maximum percent coverage, $M_\infty$, as an upper bound for a cell's total population density. The total percent cover array, $T$, is managed for all cells within the island domain for this purpose. This array is defined as

$$T(i,j) = \sum_{k=1}^{4} P_k(i,j).$$

Plants may die due to competition or overcrowding. Death by competition occurs whenever $T(i,j) > M_\infty$. Our model is designed to favor *Morella* populations ($P_3$), hence when a cell become overpopulated, residing non-*Morella* plant species (i.e. the grass species) are reduced while leaving the *Morella* populations intact. In this event we let the excess coverage be given by $a = T(i,j) - M_\infty$ and define a new value for $k^{th}$ species of grass ($k = 1, 2,$ or $4$),

$$P'_k(i,j) = P_k(i,j) - \frac{a}{l}, \tag{2.3}$$

where $l \in \{1, 2, 3\}$ is total number of grass populations present on the cell $(i,j)$. Equation (2.3) reduces the percent cover of any plant species presently residing on the cell to an even proportion of the space that is not being occupied by $P_3(i,j)$. If $P_3(i,j) = 0$, then the entire space is evenly divided among the grasses.

The woody shrub *Morella* has two more unique characteristics. *Morella* demonstrates a clear tendency to grow on the nearshore side of the island where it is guarded from the salt water spray of the ocean [30]. To accommodate for this trend, *Morella* communities having population cells within the boundaries of the beach region are subjected to gradual decay.

## 2.2 Aeolian Transport

Aeolian transportation is the movement of sediment by the wind. Saltation occurs when sediment is lifted up by the wind and deposition occurs when the sediment resettles. For our model, the presence of sufficient wind speeds prompts the removal of a slab of sediment to from one cell which is then deposited one or more cells downwind. We adapt a wind table from existing research in order to consider a range of wind speeds which are likely to cause aeolian transport [8]. Wind measurements for the model environment are based on data taken from Hog Island, Virginia between 2007 and 2012 [22]. A wind speed, $\omega$, is sampled from the data set and the possible extent of a slab being transported is determined using Table 2.2.

| **Wind Speed** *(m/s)* | **Distance (*in cells*)** |
|---|---|
| $\omega < 6$ | 0 |
| $6 \leqslant \omega < 9$ | 1 |
| $9 \leqslant \omega < 13$ | 2 |
| $13 \leqslant \omega \leqslant 16$ | 3 |
| $16 < \omega$ | not considered |

Table 2.2: The number of cells that sediment may be moved downwind corresponding to different possible wind speed values. The presence of plants may reduce the distance moved by sediment. High wind events will be the focus of later research.

A wind direction is sampled from the same data set. The chosen direction is then associated with a even scalar multiple of $\pi/8$ ($22.5^{\circ}$) corresponding to a typical compass rose. Wind in the northern direction (blowing south to north) corresponds to wind angles between $\frac{-\pi}{8}$ and $\frac{\pi}{8}$, north-western wind corresponds to wind angles between $\frac{\pi}{8}$ and $\frac{3\pi}{8}$, and so on. Erosion is considered to be taken from the current cell as $H(i,j) = H(i,j) - 1$ and deposited in any of the directions associated with the downwind current, i.e. $H'(i,j) = H'(i,j) + 1$. With a westerly wind sediment can be deposited $d \in \{1,2,3\}$

cells away, to $H'(i,j) \in \{H(i,j-d), H(i-d,j-d), H(i+d,j-d)\}$ which are the western, north-western, or southwestern cells, respectively. Every step taken requires that the angle between the current and destination cells be no greater than $15^{\circ}$ [17].

Plants act as barriers to wind flow, effectively reducing local wind speed and allowing sediment to accumulate [9]. To approximate the inhibition of erosion due to each plant species we associate an erosion coefficient parameter, $\alpha_k \in [0,1]$. Known plant characteristics are taken into consideration. For instance dune building grasses and shrubs with large root systems have a higher erosion coefficient than normal grasses. The effective plant cover array, $PC$ is the weighted effect of all plant populations at a given location on the island given by

$$PC(i,j) = \sum_{k=1}^{4} \alpha_k P_k(i,j). \tag{2.4}$$

Note that the effective plant cover array satisfies $PC(i,j) \in [0,1]$ and represents the ability of all plants on the cell to impede erosion. The effective plant cover scales wind speed to determine the probability that conditions are sufficient to overcome the impeding effects of vegetation and permit saltation at a given cell.

If a slab moves more than one cell width, each step beyond the first requires a check for vegetation present in the cell to determine if the sediment stops or continues to move. The probability of erosion at the $d^{th}$ step, $\rho_{e_d}$ for $d \in \{1,2,3\}$, is based on the weighted densities of plant populations given in equation (2.4) and the chosen wind speed, calculated as

$$\rho_{e_d} = \big(1 - PC(i,j)\big) \cdot \frac{\omega - \omega_L}{\omega_H - \omega_L}, \tag{2.5}$$

where $\omega_L$ is the minimum wind speed required for aeolian transportation, and $\omega_H$ is the maximum wind speed considered by the model. Note that equation (2.5) satisfies $\rho_{e_d} \in [0,1]$, and as the effective plant cover $PC$ increases, the probability of erosion decreases. The likelihood of erosion is therefore inversely proportional to the density of vegetation while being directly proportional to the wind speed.

A cell being transported has a 50% chance of moving with the wind direction, and a 25% of moving in either of the off-directions. A more rigid probability is dependent upon the angle between the current slab and the destination slab, with preference going to the direction where the angle is steepest. We do not adjust for this probability here, as sediment will always shift in the direction of a sufficiently steep angle as will be explained in section 2.3.

## 2.3 Avalanche

As sediment moves around the island domain, it is possible that unrealistically steep mounds of sediment have formed. Gravitational forces effect these steep mounds by causing avalanches of sediment from areas of higher elevation into areas of lower elevation when certain conditions are met.

| | | |
|---|---|---|
| 1 H(i-1, j-1) | 2 H(i-1, j) | 3 H(i-1, j+1) |
| 4 H(i, j-1) | 5 H(i, j) | 6 H(i, j+1) |
| 7 H(i+1, j-1) | 8 H(i+1, j) | 9 H(i+1, j+1) |

Figure 2.2: 8 cell neighborhood

The angle of repose between two cell elevations is given by the angle measure

$$\theta' = \tan^{-1}\left(\frac{H(i,j) - H'(i,j)}{L} \cdot \delta\right), \tag{2.6}$$

where $\delta H'(i,j)$ is the elevation of any cell in the von Neumann (4-cell) neighborhood of cell $(i,j)$, labeled as 2, 4, 6, and 8 in Figure 2.2.

A critical angle of repose, $\theta_o$, is defined as the shallowest angle between neighboring stacks of sediment which prompts sediment to collapse [1]. If $\theta'$, as calculated from equation (2.6), satisfies $\theta' \geqslant \theta_o$, then the polled cell is in violation of the critical angle of repose and the probability of avalanching, $\rho_{av}$, is given by

$$\rho_{av}(i,j) = \min\left(1, \frac{\theta'}{\theta_o} \cdot (1 - PC(i,j))\right). \tag{2.7}$$

In the absence of plant populations, $PC(i,j) = 0$, and avalanching is guaranteed. For cells partially or entirely covered by vegetation the erosive quality of the surrounding sediment is hindered [30]. In this scenario the probability found by equation (2.7) the effective percent cover value satisfies, $PC(i,j) > 0$, as given in equation (2.4). Clearly when the effective plant cover is high, the probability of collapse is much reduced.

## 2.4 Marine Processes

Barrier Island shorelines are sculpted by a variety of processes, including tidal erosion and deposition, and aeolian transport of sediment. Each of these mechanisms involve a great deal of variation. For instance, water levels vary on a variety of timescales such as those during daily tidal cycles versus weekly neap-spring cycles, or during inclement weather events which bring storm surges and overwashing; any of which may become more or less prevalent over seasons or years [6].

The model focuses on long-term migratory behavior due to sea-level rise, and not on active shoreface profiling. The beach profile is assumed to be impacted only by aeolian processes. The sediment supply to the beach is otherwise abundant and sufficient to maintain a given rate of migration. For our purposes, the islands migrate at some rate that is consistent with observed shoreline changes as well as the assumptions required to employ the Bruun model as outlined in [25].

The rate of shoreline recession, $R$, varies widely from island to island, so the individual rates of migration are estimated from existing resources. For the Virginia barrier islands that are the focus of this study, these rates were established using the Virginia Coastal Resilience planning tool [27] which is compiled based on data sets provided by the Virginia Institute for Marine Science [28].

The Bruun model derives a basic relationship for predicting the shoreline recession, $R$, from an increase is seal level rise, $S$, as

$$R = \frac{A^*}{B + h^*} S. \tag{2.8}$$

$A^*$ is taken to be the cross-shore distance to the depth of closure, $h^*$. The depth of closure is the depth along the beach profile at which point sediment transport is minimal or non-existent. The height of the berm, $B$, is the uppermost potion of the beach face. As noted by Davidson-Arnott [6], for shallow shore angles, $\theta_b$, the equation 2.8 can be approximated as

$$R \approx \frac{1}{tan \theta_b} S,$$

where $\theta_b \approx (B + h^*)/A^*$ is the average slope of the nearshore. A common rate for sea level rise on the central eastern coast of United States is about 1/4 inch, or 6.35 millimeters, per year [20].

For $R < L$, where $L$ is the width of a cell in the island domain, accumulated migration distances for each vertical location, $i$, is stored in the vector $R_i$ until sufficient time has passed such that the island can be moved landward by one unit length with respect to that vertical coordinate. The entire subaerial portion of the island then migrates in unison with the shoreline, as defined as the easternmost horizontal cell location, $j_s$, of the elevation map for each vertical location $i$, for which $H(i, j_s) \geqslant 0$.

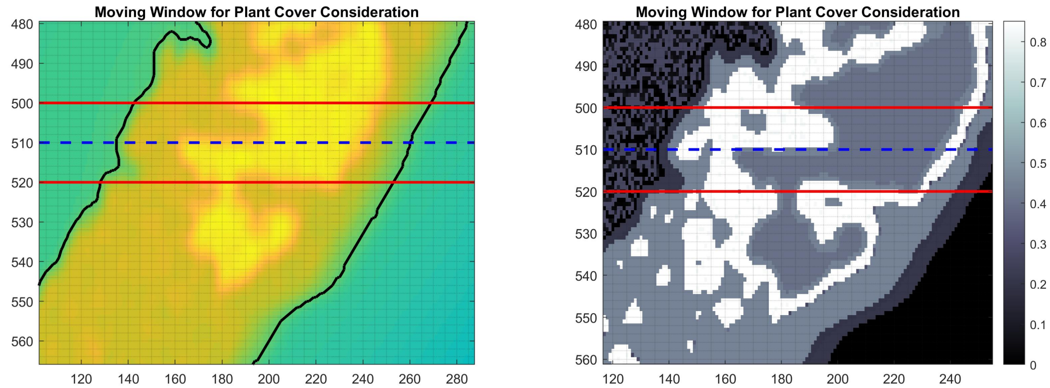



Figure 2.3: A moving window transect considering the impact of vegetation on the yearly migration of the shoreline for row $i$ (given by the dashed line). Equation (2.9 takes into consideration all of the values of $PC$ which fall within the red lines. The left image is the elevation map, while the right image is the total population density scaled by erosion inhibiting factors, $PC$)

The effect of vegetation on migration is measured using a moving transectional window of size $2w \times n$ as given in Figure 2.3. The value $w$ is number of rows above and below the current row, $i$, and $n$ is the total number of horizontal cells in the island domain. Using the effective plant cover, $PC$, as given in equation (2.4), the total weighted percent coverage values within the window are averaged such that cumulative impact of the nearby plant population at vertical location $i$ is given by

$$X_p = \frac{\sum_{r=-w}^{w} \sum_{j=1}^{n} PC(i+r, j)}{\psi}, \tag{2.9}$$

where $\psi$ is the area of the subaerial portion of the island within the transectional window. The resulting scalar, $X_p \in [0, 1]$, is used to calculate the extent of the migration. The migration landward in meters for the current row, $R_i$, is reduced each year by factors given in Table 2.3.

| **Reduction of** $R_i$ | $X_p$ **range** |
|---|---|
| $100\%$ | $0.5 \leqslant X_p$ |
| $70\%$ | $0.35 \leqslant X_p < .5$ |
| $50\%$ | $0.1 \leqslant X_p < .35$ |

Table 2.3: Reduction values for ranges of scaled total percent cover

We wish to examine different sea level rise scenarios. To accomplish this we define the following equation

$$R = \lfloor R_o (M_a)^{t/26} \rfloor, \tag{2.10}$$

where $R_o$ is the initial rate of the island migration, $t/26$ gives the current number of years passed (we take $t$ to be a two week time step), and $\lfloor \cdot \rfloor$ is the floor function. The parameter for migration acceleration, $M_a$, is varied to correspond with a desired rate of sea level rise. We wish to consider the absence of sea level rise, constant rates of sea level rise, and accelerating rates of sea level rise. For no sea level rise we take $M_a = 0$. Constant sea level rise is achieved by taking $M_a = 1$. For acceleration of sea level rise, we take $M_a > 1$ such that the desired rate of increase is observed.

# Chapter 3

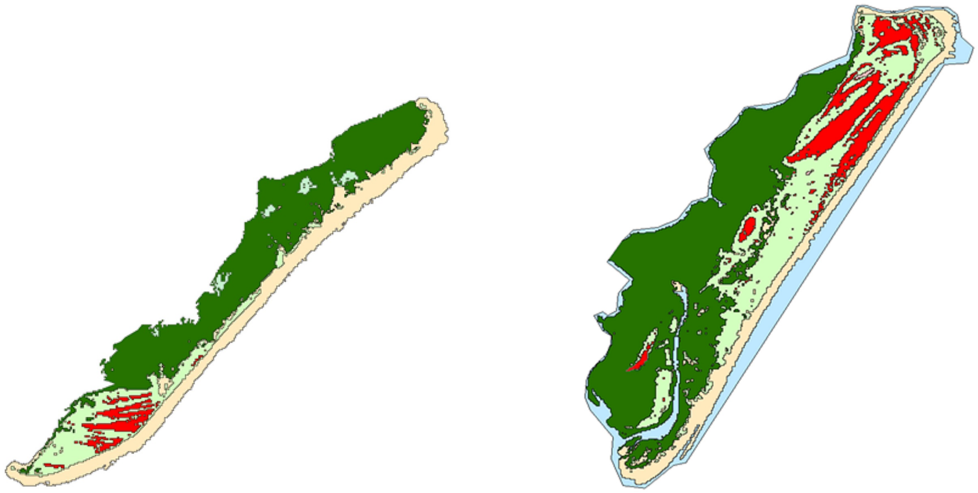
# Results

The goal of this study is to model the impact of vegetation on the evolution of a barrier island system. We assess variations in evolutionary behavior based on the percent cover of plant populations and three different sea level rise scenarios. Baseline parameter values are established using vegetation and shoreline data for existing islands. With these parameters established, we examine the effects of vegetation assuming no sea level rise, constant sea level rise, and accelerating sea level rise.

## 3.1 Experiment Design

Two unique barrier island elevation maps are utilized establish our parameter values. The maps are used to outline basic trends in evolution at an accelerating rate of sea level rise. Both are compared to existing geographical images to confirm model accuracy and thus confirm that the chosen parameters are reasonable. Either map will then be used for additional testing which will vary the initial plant percent cover conditions and rates of sea level rise.

The maps are created using the known distribution of plant species on two barrier islands from the Eastern Shore of Virginia: Smith Island and Parramore Island. Simple imagery compiled from field observations are color-coded by variety of vegetation

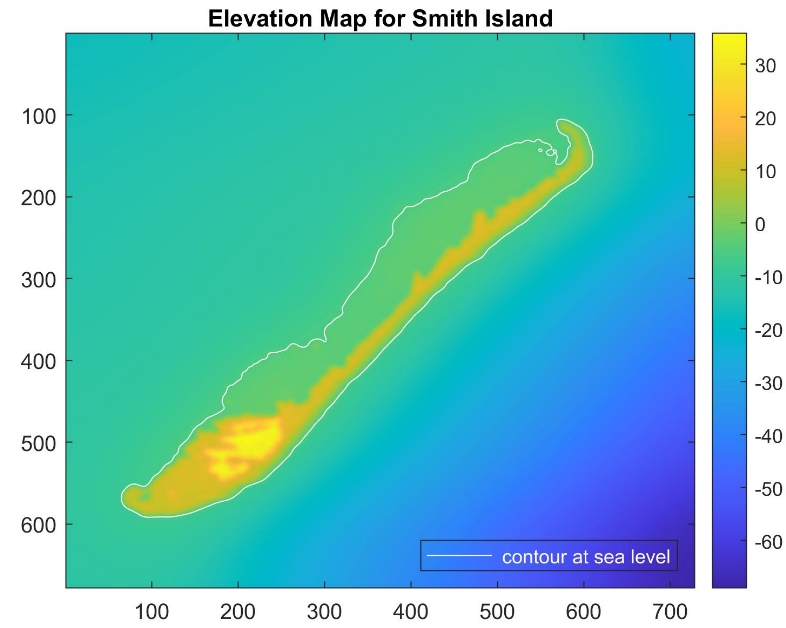


(a) Smith Island (b) Parramore Island

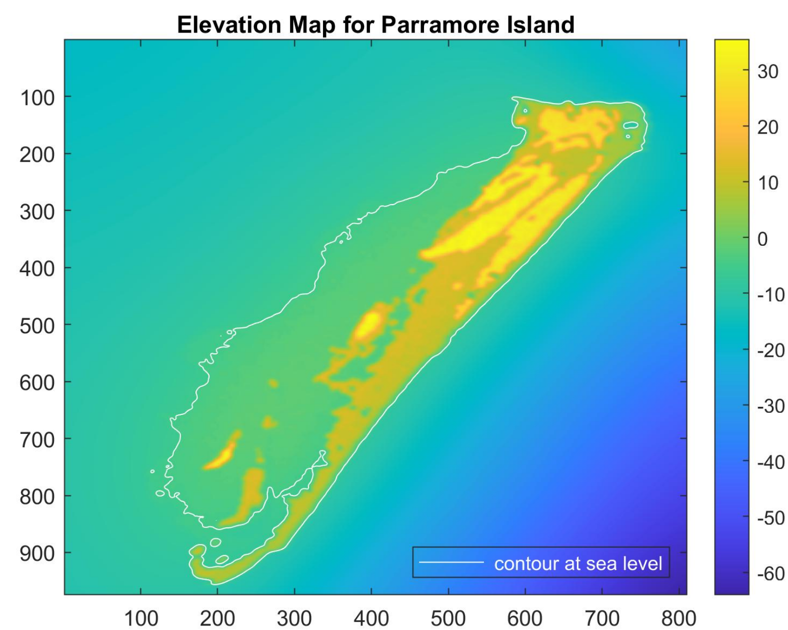
Figure 3.1: Original, color-coded images of Virginia barrier islands

present at each area. As seen in Figure 3.1, a dark green was associated with the marsh region, a lighter green for the dune grasses, red for the woody shrub, and a tan beach region with no vegetation.

Using these maps we approximated the elevation array, $H(i, j)$, based on Table 2.1. Areas below sea level were added by linear interpolation east and west from the boundary of island, with a shallower slope for the backbarrier marsh west of the island and a steeper slope for the ocean facing eastern side of the island. The resulting elevation maps used for our tests are given in Figure 3.2. These elevation maps are used to seed the island with appropriate plant life, generating our plant cover arrays, $P_k$.




(a) Smith Island



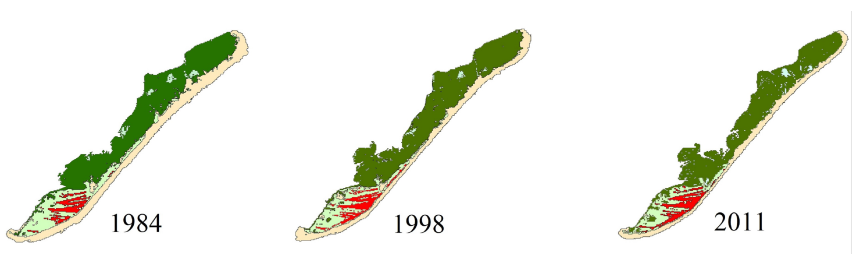

(b) Parramore Island

Figure 3.2: Elevation maps for Smith and Parramore Island

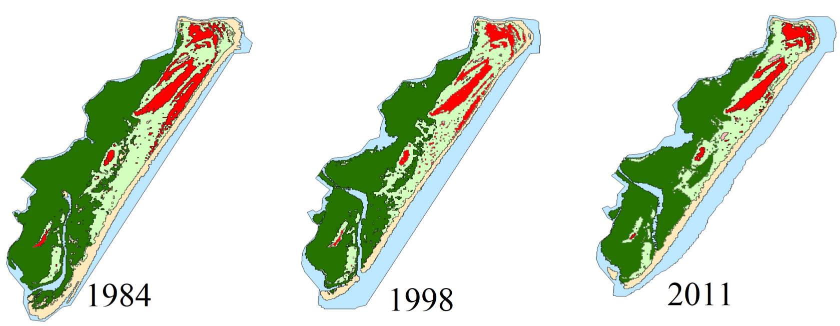
Note from Figure 3.1a that the northern portion of Smith island is long and narrow, with significant beach cover and little vegetation. Alternatively, the southern portion has much denser cover, and more plant variety, covering a larger surface area. In this case we would expect that vegetation would hinder the migration and sediment movement on the southern portion of the island and the lack of plant cover would encourage movement in the northern region. Parramore Island, seen in Figure 3.1b, shows the opposite trend: more vegetation in the north and less in the south. We would expect to see more movement in the south and less movement in the north.

Island evolution is simulated over a 27 year time period. The resulting figures are compared to illustrations taken from satellite imagery over the period of time from 1984-2011, seen in 3.3. These images are converted to overlaid contours, and compared with contour outputs of the model sampled at 9 year intervals.

## 3.2 Parameterization




(a) Smith Island evolution




(b) Parramore Island evolution

Figure 3.3: Illustration of island evolution spanning 27 years

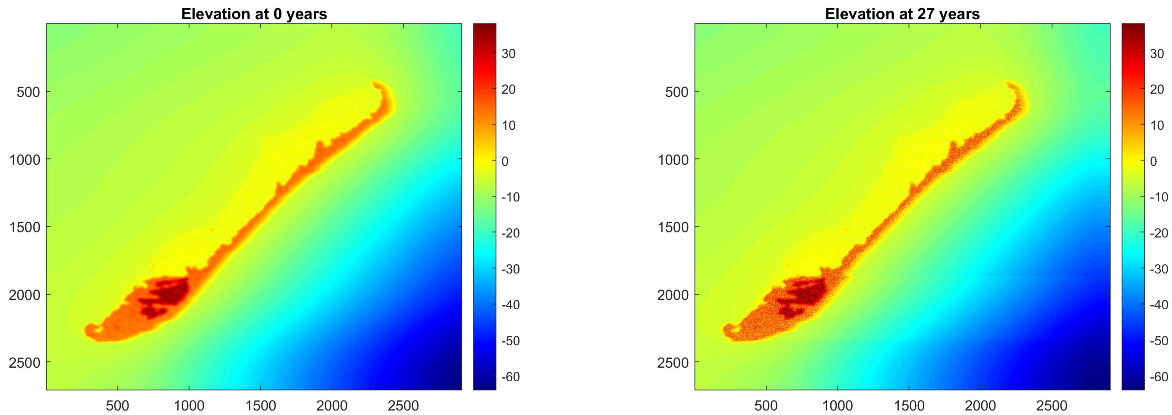
Parameterization for this model is difficult due to the large number of parameters, the paucity of good, long-term available data, and lack of sources with proper experimentation for these parameter values. Some of our parameters are established through educated guess followed by repeated model testing for optimization (for a full list see Table A.1 in appendix A). The parameters that require specific attention for these tests included those in Table 2.3 and those given in Table 3.1.

Note the coefficient $\alpha_1$ is slightly larger than $\alpha_2$, reflecting the status of $P_1$ as the primary dune building grass. The coefficient $\alpha_3$ is large because the woody shrub has larger root systems, and wider leaf cover, making it more capable of stopping sediment in motion. The marsh *Spartina* has a large erosion coefficient due to being primarily underwater. The growth range $\gamma_k$ along with the death by elevation was was established through repeated model tested to ensure plant life stayed abundant and thriving.

The maximum percent cover values for all plant populations used during parameterization is taken to be 100%, ensuring an abundant plant population. Sea level rise is accelerating, resulting in an escalation of yearly migration. We take these conditions to be approximate to those of the island between 1984 and 2014.

| Notation | Definition | Value |
|---|---|---|
| $\alpha_1$ | erosion coefficient for $P_1$ | 2/3 |
| $\alpha_2$ | erosion coefficient for $P_2$ | 1/3 |
| $\alpha_3$ | erosion coefficient for $P_3$ | 1 |
| $\alpha_4$ | erosion coefficient for $P_4$ | 1 |
| $\gamma_k$ | growth range | [-0.02, 0.08] |
| $\beta$ | death by elevation percentage | -0.3 |
| R | rate of migration due to sea level rise | 15 m/year |

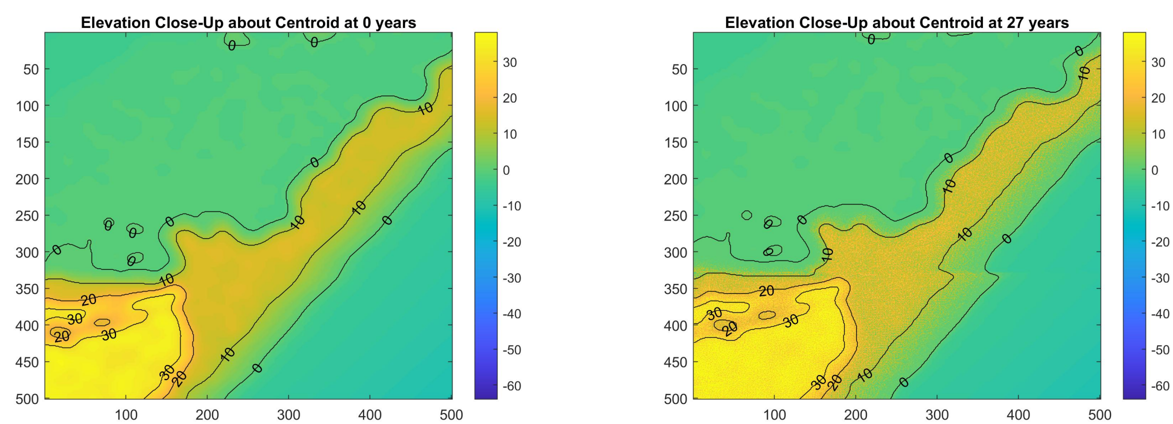
Table 3.1: Select parameters optimized during this study, along with the migration rate reduction values given in Table 2.3.




(a) (b)

Figure 3.4: Plots of the elevation map for Smith island before and after 27 years of evolution.

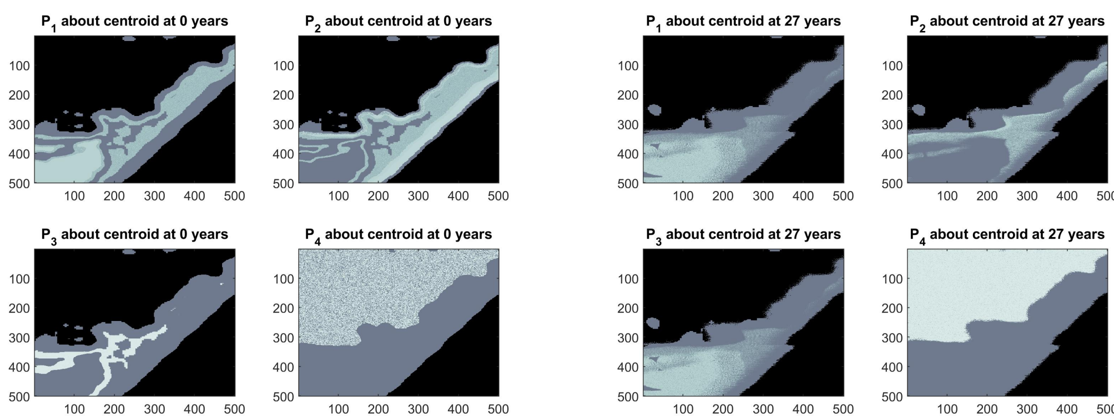
The plots in Figure 3.4 are taken from the initial elevation map and the map after 27 years of evolution. Very little change is evident, which is in keeping with expectations for high initial plant cover. Zooming in around the centroid of the map, given in Figure 3.5, allows us to see the effects of aeloian transport and avalanching in more granular detail. We can also see, around the middle of both figures.




(a) (b)

Figure 3.5: Close up of the elevation map with contours for areas around the centroid of Smith island before and after 27 years of evolution.

Plots of the vegetation cover are given in Figure 3.6. We can see in the Figure 3.6a how the plant populations are initially placed in very distinct locations with clear boundaries where the edges of their viable growth ranges occur. The black areas correspond to negative elevation areas; unviable for all plants except for the marsh Spartina, $P_4$ in the lower right. The darker shade of grey represents subaerial portions of the island which are outside of the plants elevation range. All varying lighter shades represent some population of the given species present in that cell, with the lighter shades being the highest percent cover possible. Grey areas are often shared by overlapping populations of different species, but the bottom left image shows the clear preference $P_3$ is given within it's comparatively minimal elevation range.

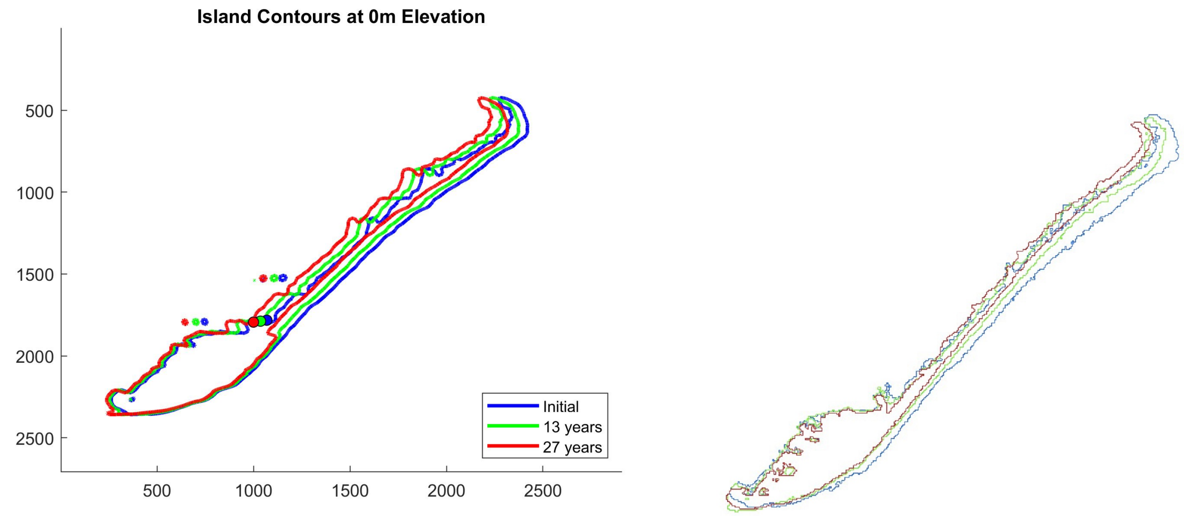



(a) Initial plant distribution

(b) Plant distribution after 27 years

Figure 3.6: Plots of percent plant cover for each $P_k$ at areas focused around the centroid of Smith island before and after 27 years of evolution.

The second Figure 3.6b shows significant spread of all plant species, indicating that the growth range, $\gamma_k \in [g, G]$, has been chosen to allow optimal vegetation growth. Morella has followed a pattern of migrating away from the beach, leaving bands of older populations to die as sediment shifting create conditions outside of the plants viable elevation range. In these areas, Spartina has moved in to take greater portions of the the vacated cells. The marsh Spartina population has enjoyed abundant growth in underwater areas where it is uncontested, and also dominates the larger part of the western shoreline.



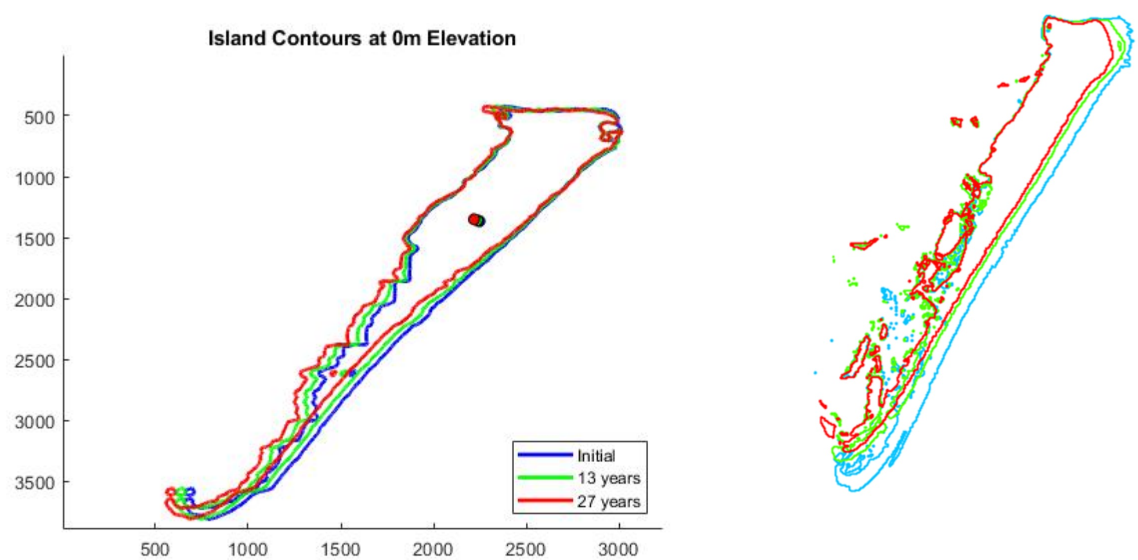

(a) (b)

Figure 3.7: A progression of contours plots of elevation map taken at 0, 12, and 27 years compared to similarly timed contours of Smith island. The dots represent the centroid of the island for that year.

Comparison of the contour lines taken at sea level around the shorelines of the island display the migratory trend of the island. The Figure 3.7 demonstrates clear landward progression of the island (the island is oriented such that the mainland is to the west, or left on the image). The centroid, defined as the center of mass for all positive elevation areas of the island, is plotted as a filled dot with color corresponding to the year. This point is a good tool for referring to trends around areas of the island with dense vegetation and multiple plant species. Note that the progression of the centroids taken during the same years as the contours follows a similar trend westward. In comparing our model to the contours taken from the true island, several positive relationships can be established.

The long, thin northern portion of the island is at lower elevations with a greater area of beach. This limits the area in which vegetation can grow. The lower elevations also prevent Morella from maintaining populations, leaving only the dune grasses which are less capable of hindering erosion. Alternatively, the southern area of the island is much wider and has a greater range of elevation values. This permits the full variety of plant species to occupy a greater area of the island. The impact of this can be seen in the image - the bottom, wider portion of the island migrates significantly less than the northern portions. Most importantly, this trend can be observed in both the model island evolution and the observed island evolution. This indicates that the values selected for migration rate reduction, $X_p$ given in Table 2.3, are appropriately chosen to return optimal results.

To confirm reasonable parameter estimates, we simulated the same sea level rise scenario with identical parameter values over the same time frame on Parramore Islands. This led to the results given in Figure 3.8.




(a) (b)

Figure 3.8: A progression of contour plots of elevation map taken at 0, 12, and 27 years compared to similarly timed contours of Parramore island. The dots represent the centroid of the island for that year.

Parramore island has denser land mass to the north, with a long narrow portion in the south. The contours of the island after 27 years of evolution are given in Figure 3.8. The migration in the North has clearly been stifled by the presence of vegetation.

Comparing the model behavior to the observed island contours, we see that the model evolutionary behavior largely follows empirical trends.

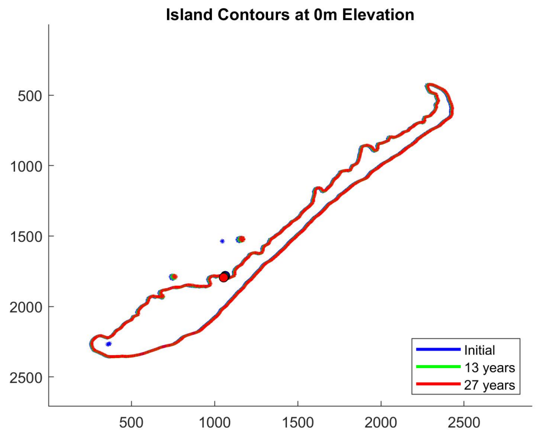
We note here that there are a few differences between the model behavior and the observed island behavior that can be pointed out in Figures 3.7 and 3.8. There is some lateral shrinking of the islands and additional regression in the southern part of Parramore Island, for instance. However, the overall drift of each island appears to be reasonably close and at this point in development, it is our goal to capture the migratory behavior of the island and dependence upon living plant populations. Many processes, like high wind events, storms, and overwashing, contribute to the geographical evolution of the island's topography and the shape of the shorelines. We are further restrained by the limits of atmospheric data available of this time frame, and the manner in which it is implemented is similarly restricted as a result. It is our intention to include many more atmospheric processes and improvements in future model development. The results we have shown are highly positive given our limitations.

## 3.3 Results - Variations on Initial Plant Conditions and Sea Level Rise

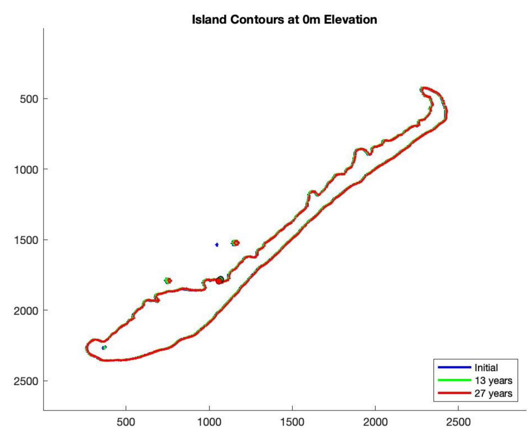
We simulate the island under varying conditions of sea level rise and plant percent coverage to demonstrate the dependence of barrier island geomorphology on sea level rise and the presence of vegetation. Sea level rise is taken to be either non existent, constant, or accelerating. Plant percent cover conditions are taken to be either non existent, at 50% of maximum coverage, or at 100% of maximum coverage. The island contours are used to establish key relationships and make observations.

### 3.3.1 No Sea Level Rise

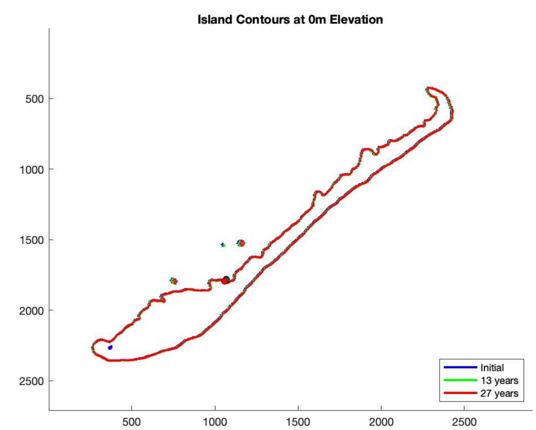
Island evolution in the absence of sea level rise is achieved by taking $M_a = 0$. It follows that the island migration rate, $R$, would be reduced to zero by the rate acceleration function given in equation (2.10). The contours are given for Smith island.




(a) 0% plant cover




(b) 50% plant cover



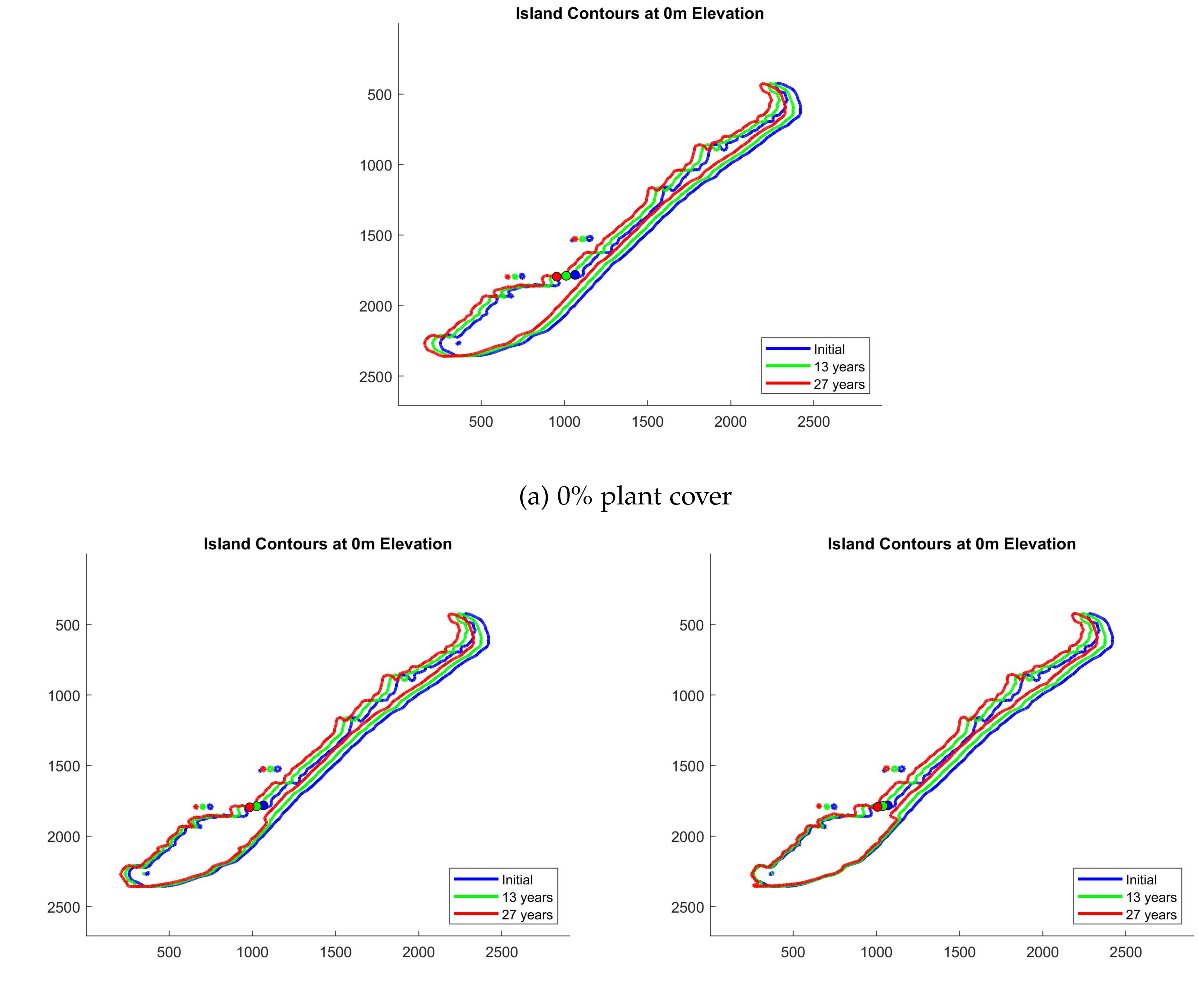

(c) 100% plant cover

Figure 3.9: Evolution of Smith island in the absence of sea level rise, $M_a = 0$, taken at varying percentages of initial and maximum plant cover.

As expected, there is virtually no movement of the island. Shorelines vary in the slightest degree, which is evidence that the aeolian transport and avalache processes are still being carried out. The same reasoning explains the slight change in centroid position.

### 3.3.2 Constant Sea Level Rise

Constant sea level rise is simulated by taking the migration acceleration growth factor to be $M_a = 1$.




(a) 0% plant cover

(b) 50% plant cover

(c) 100% plant cover

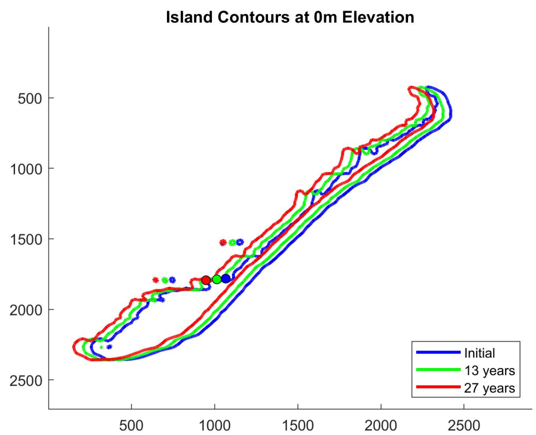
Figure 3.10: Evolution of Smith island with constant sea level rise, $M_a = 1$, taken at varying percentages of initial and maximum plant cover.

We can see in the Figure 3.10 that there is a notable change in the evolutionary behavior as the initial and maximum percent cover values are increased. The island observed in the absence of all plants, Figure 3.10a, can be seen to migrate uniformly. The curvature of the Eastern coastline remains generally unchanged, with small variations that can be attributed to aeolian transportation and avalanche. In Figure 3.10b we begin to see the effect of plants diminishing the migration, which particular hinderance occuring at the southern portion of the island where vegetation covers a greater surface area. The final Figure 3.10c displays full coverage bringing the southern area of the island to a near stand still. The centroids are grouped increasingly close together as the plant cover is increased, further illustrating this reduction of migratory behavior.

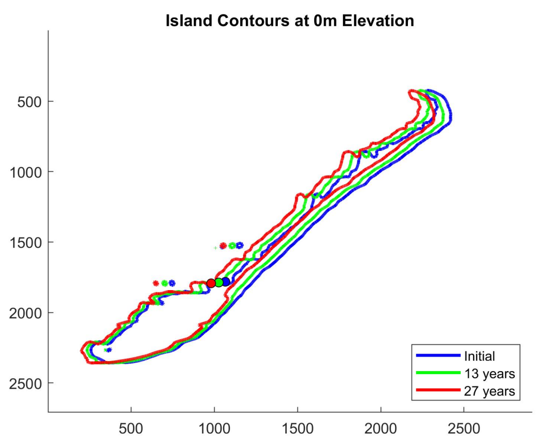
Note also the orientation of the centroids. The line connecting the centroid points becomes gradually more curved as the plant cover increases. We also see a small isthmus forming at the very southern tip of the island. When considered together, these observations demonstrate the island gradually curving, and more years of evolution may result in the pronounced overall bowed shape; a feature common to many barrier islands.

### 3.3.3 Accelerating Sea Level Rise

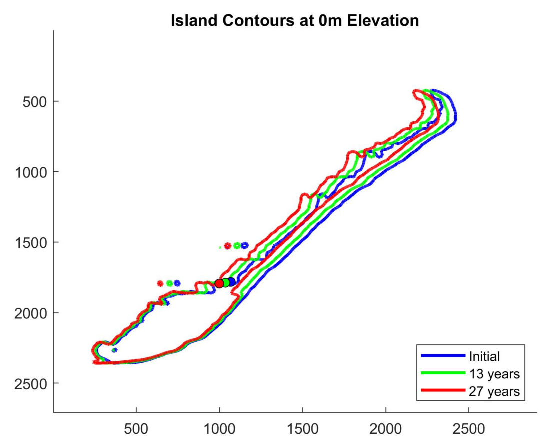
Accelerating sea level rise is achieved using a growth rate of 0.75%, as determined by the parameterization process. This gives us a growth factor $M_a = 1.0075$ which, when applied in equation (2.10), results in a gradual accelertion of landward migration. We would expect to see trends similar but more pronounced than those with constant see level rise.




(a) 0% plant cover




(b) 50% plant cover




(c) 100% plant cover

Figure 3.11: Evolution of Smith island with accelerating sea level rise, $M_a = 1.0075$, taken at varying percentages of initial and maximum plant cover.

We do indeed see the same trends as with a constant rate of sea level rise. The initial Figure 3.11a shows near uniform migration of both east and west shorelines, indicated that the island has the same migratory trends everywhere. Again we see in Figures 3.11b and 3.11c that the migration is significantly inhibited by increasing the initial and maximum percent cover values.

In comparing the vertical and horizontal location of the centroids we obtain more information. The initial, 13 year, and 27 year centroid coordinates for full plant cover and constant sea level rise given in Figure 3.10c are (1069,1782), (1040,1786), and (1007,1793), respectively. Considering a unit cell width of $\mathrm{L}$, this means that the centroid of the island migrated by $29 * \mathrm{L}$ meters in the first 13 years, and an additional $33\mathrm{L}$ meters in the following 14 years, for a total of $62\mathrm{L}$ meters over the whole 27 year span. The centroid coordinates in Figure 3.11c are (1069,1782), (1036,1788), and (999.5,1794). The same analysis yields migration values of $33\mathrm{L}$ meters in the first 13 years and $36.5$ meters in the last 14 years, for a total of $69.5\mathrm{L}$ meters in 27 years. The overall migration is then $7.5\mathrm{L}$ meters greater when sea level rise is accelerating. Since the centroid is necessarily located in the densest area of the island, this serves as a lower bound on the possible increase in migration for this scenario, and all other areas of the island would certainly experience even greater increases in landward migration.

# Chapter 4

# Conclusion

A four species of plants are included within the island domain. Plant populations were configured to allow for natural competition, growth, death and propagation in accordance with known species characteristics. Using educated estimates followed by repeated testing, the model was parameterized to approximate historical maps of island evolution. We optimize erosion coefficients which are used to define an effective total percent cover of vegetation for each location on the island. This effective percent cover is used when calculating the probability of sediment shifts by meteorological processes included in the model.

Aeolian transportation captures the effect of wind erosion via saltation and deposition of sediment slabs in respose to sufficient wind conditions sampled from real-world meteorological data. Slabs of sediment were permitted to move multiple steps, subject to wind conditions and the restrictions that capture the hindering effect of local vegetation. A probability of sediment transfer is defined by incorporating the effective percent plant cover.

Avalanching captures the effect of natural gravitational collapse. The critical angle of repose defines sufficient conditions for collapse in the absence of plant life. Dependence upon local vegetation defines the probability of collapse which is inversely proportional to the local effective percent coverage of plant species.

Marine process captures the long-term effect of sea level rise on island migration in accordance with the Bruun rule for shoreline recession. This landward motion is reduced by a scaling factor defined using the effect plant cover and island subaerial surface area. The values of this scaling factor are associated with migration reduction percentages. We employed a moving window to calculate the migration reduction to ensure realistic island movement. Additionally, we define the acceleration of sea level rise based on a growth parameter which was similarly optimized to demonstrate island evolutionary behavior consistent with historical trends.

The barrier island evolution model that we developed combines four processes to demonstrate how the presence of vegetation effects the evolution of the entire barrier island on the decadal time scales. Using two of Virginia's Eastern Shore barrier islands, we simulated 27 years of geographic development for sea level rise scenarios including the absence of sea level rise, constant rates of sea level rise, and accelerating rates of sea level rise. We further demonstrated the model's global dependence on plant populations by varying the maximum percent cover between 0%, 50% and, 100% for all three sea level rise scenarios. We presented and analyzed the evolutionary results using island contours taken at 0, 13, and 27 years.

The model demonstrated a dramatic increase of landward migration as the rate of sea level rise escalated. This rate of landward migration was large in the absence of plant life, and shown to be appropriately hindered when the percent cover of vegetation was increased. Wider, more densely vegetated areas of the island experienced less shoreline migration, while narrower and sparsely populated areas saw significantly more landward movement. Reducing plant cover to zero showed the shorelines moving uniformly. The absence of sea level rise dramatically decreased migration, and demonstrated near uniform movement of the shoreline regardless of plant cover.

Many barrier island models have been developed which include one or more of the aeolian, avalanche, and marine processes, or a plant population sub-model. Ours is the first to combine all four elements and employ them on a whole island domain which includes ocean, beach, central dune field, backbarrier marsh, and all shorelines from the northern to the southern tip of the island.

There is much promise in the model, and further development will increase model reliability and application beyond the scope we have presented here. Future versions of the model will include more detailed shoreline development, which would require more robust development marine processes to include sediment fluxes in response to tidal activity and evolution of the beach profile. Further research is also needed to incorporate the known currents around barrier islands in order to capture the subtle but significant southern drift of Virginia's barrier islands.

Additional field observation and laboratory testing of plant species, specifically with respect to their erosion inhibiting capabilities, will continue to inform our parameter selection. With more information about how successfully plant species impede sediment collapse, saltation, and transfer on the wind, we can ensure that the model is performing with the greatest accuracy. Similarly, as more data becomes available regarding species responses to salt exposure and burial by sediment, we can ensure that the parameters governing plant growth and death cycles are best selected to reflect the empirical conditions.

Much of a barrier island's migration is attributed to the process of sediment movement from the beach face into the backbarrier region of the island [19]. Wave run-up and water level surges during storms create overwashing flows [18]. Further development of the model will include storm events and subsequent overwashing while accounting for the effect of increased wind speeds. Furthermore, the increase of occurrence and intensity of storms is closely linked to climate change. For this reason, incorporating storm events into our current model is essential to the goal of understanding the impact of climate change on barrier island evolution.

# Bibliography

[1] Baas, A. C. Chaos, fractals and self-organization in coastal geomorphology: simulating dune landscapes in vegetated environments. *Geomorphology (Amsterdam, Netherlands) 48*, 1-3 (2002), 309–328.

[2] Brantley, S. T., Bissett, S. N., Young, D. R., Wolner, C. W. V., and Moore, L. J. Barrier island morphology and sediment characteristics affect the recovery of dune building grasses following storm-induced overwash. *PloS one 9*, 8 (2014), e104747–e104747.

[3] Brenner, O. T., Moore, L. J., and Murray, A. B. The complex influences of back-barrier deposition, substrate slope and underlying stratigraphy in barrier island response to sea-level rise: Insights from the virginia barrier islands, mid-atlantic bight, u.s.a. *Geomorphology (Amsterdam, Netherlands) 246* (2015), 334–350.

[4] Bruun, P. Sea-level rise as a cause of shore erosion. *Journal of Waterways and Harbors Division 88*, 1 (1962), 117–132.

[5] Dai, H., Ye, M., and Niedoroda, A. W. A model for simulating barrier island geomorphologic responses to future storm and sea-level rise impacts. *Journal of coastal research 31*, 5 (2015), 1091–1102.

[6] Davidson-Arnott, R. G. D. Conceptual model of the effects of sea level rise on sandy coasts. *Journal of Coastal Research 21*, 6 (2005), 1166–1172.

[7] Dean, R. G. Equilibrium beach profiles: Characteristics and applications. *Journal of coastal research 7*, 1 (1991), 53–84.

[8] Delgado-Fernandez, I., and Davidson-Arnott, R. Meso-scale aeolian sediment input to coastal dunes: The nature of aeolian transport events. *Geomorphology 126*, 1-2 (2011), 217–232.

[9] Feagin, R. A., Figlus, J., Zinnert, J. C., Sigren, J., Martínez, M. L., Silva, R., Smith, W. K., Cox, D., Young, D. R., and Carter, G. Going with the flow or against the grain? the promise of vegetation for protecting beaches, dunes, and barrier islands from erosion. *Frontiers in ecology and the environment 13*, 4 (2015), 203–210.

[10] Fiedler, J. W., Smit, P. B., Brodie, K. L., McNinch, J., and Guza, R. Numerical modeling of wave runup on steep and mildly sloping natural beaches. *Coastal engineering (Amsterdam) 131* (2018), 106–113.

[11] Keijsers, J., De Groot, A., and Riksen, M. Vegetation and sedimentation on coastal foredunes. *Geomorphology (Amsterdam, Netherlands) 228* (2015), 723–734.

[12] Keijsers, J. G. S., De Groot, A. V., and Riksen, M. J. P. M. Modeling the biogeomorphic evolution of coastal dunes in response to climate change: Modeling coastal dunes. *Journal of geophysical research. Earth surface 121*, 6 (2016), 1161–1181.

[13] Lorenzo-Trueba, J., and Ashton, A. Rollover, drowning, and discontinuous retreat: Distinct modes of barrier response to sea-level rise arising from a simple morphodynamic model. *Journal of Geophysical Research: Earth Surface 119* (04 2014).

[14] Masetti, R., Fagherazzi, S., and Montanari, A. Application of a barrier island translation model to the millennial-scale evolution of sand key, florida. *Continental shelf research 28*, 9 (2008), 1116–1126.

[15] Masterson, J. P., Fienen, M. N., Thieler, E. R., Gesch, D. B., Gutierrez, B. T., and Plant, N. G. Effects of sea-level rise on barrier island groundwater system dynamics - ecohydrological implications. *Ecohydrology 7*, 3 (2014), 1064–1071.

[16] Nettleton, Benjamin P. (Benjamin Peter), .-a. *The role of vegetation-topographic interactions in a barrier island system : island migration in a changing climate*. 2018.

[17] Nield, J. M., and Baas, A. C. W. Investigating parabolic and nebkha dune formation using a cellular automaton modelling approach. *Earth surface processes and landforms 33*, 5 (2008), 724–740.

[18] Nienhuis, J. H., Heijkers, L. G., and Ruessink, G. Barrier breaching versus overwash deposition: parameterizing the morphologic impact of storms on coastal barriers. *Earth and Space Science Open Archive ESSOAr* (2021).

[19] Nienhuis, J. H., and Lorenzo-Trueba, J. Can barrier islands survive sea-level rise? quantifying the relative role of tidal inlets and overwash deposition. *Geophysical research letters 46*, 24 (2019), 14613–14621.

[20] Parris, A. S. a. *Global sea level rise scenarios for the United States National Climate Assessment*. NOAA Technical Report OAR. CPO ; 1. 2012.

[21] Passeri, D. L., Dalyander, P. S., Long, J. W., Mickey, R. C., Jenkins, R. L., Thompson, D. M., Plant, N. G., Godsey, E. S., and Gonzalez, V. M. The roles of storminess and sea level rise in decadal barrier island evolution. *Geophysical research letters 47*, 18 (2020), n/a.

[22] Porter, J., Krovetz, D., Nuttle, W., and Spitler, J. Hourly meteorological data for the virginia coast reserve lter 1989-present ver 36. https://doi.org/10.6073/pasta/c5538bb29f26c6099cb7d4ea0500e7b5, 2018.

[23] Rastetter, E. B. A Spatially Explicit Model of Vegetation-Habitat Interactions on Barrier Islands. In *Quantitative Methods in Landscape Ecology*, R. H. Gardner, Ed., vol. 82. Springer-Verlag, New York, 1991.

[24] Rosati, J. D., Dean, R. G., and Stone, G. W. A cross-shore model of barrier island migration over a compressible substrate. *Marine geology 271*, 1 (2010), 1–16.

[25] Schwartz, M. L. The bruun theory of sea-level rise as a cause of shore erosion. *The Journal of geology 75*, 1 (1967), 76–92.

[26] Seabloom, E. W., Ruggiero, P., Hacker, S. D., Mull, J., and Zarnetske, P. Invasive grasses, climate change, and exposure to storm-wave overtopping in coastal dune ecosystems. *Global change biology 19*, 3 (2013), 824–832.

[27] The Nature Conservancy. Coastal resilience. `https://maps.coastalresilience.org/virginia/#`, 2016. Last accessed 17 April 2021.

[28] Virginia Institute of Maine Science. Sea level rise - accomack county. `http://cmap2.vims.edu/SeaLevelRise/Accomack_SLR.html`, 2016. Last accessed 17 April 2021.

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

[30] Zinnert, J. C., Shiflett, S. A., Via, S., Bissett, S., Dows, B., Manley, P., and Young, D. R. Spatial–temporal dynamics in barrier island upland vegetation: The overlooked coastal landscape. *Ecosystems (New York) 19*, 4 (2016), 685–697.

# Appendix A

# Notation Index

| Notation | Definition | Value |
|---|---|---|
| $n$ | number of cell widths in island domain | 678, 974 * |
| $m$ | number of cell lengths in island domain | 728, 810 * |
| $\delta$ | height of each slab | 0.1 meters |
| $L$ | width and length of each slab | 4 meters |
| $\lambda_{L,k}$ | minimum viable elevation for $P_k$, $k = 1, 2, 3, 4$ | *see Table 1* |
| $\lambda_{H,k}$ | maximum viable elevation for $P_k$, $k = 1, 2, 3, 4$ | *see Table 1* |
| $\beta$ | death by elevation percentage | 30% |
| $\gamma_k$ | growth death percentange ($k = 1, 2, 3, 4,\ \gamma_k \in [g, G]$) | -2%-8% $yr^{-1}$ |
| $g$ | plant growth minimum | -0.02 |
| $G$ | plant growth maximum | 0.08 |
| $\eta_k$ | plant percent cover maximum for each $P_k$, $k = 1, 2, 3, 4$ | 80%, 80%, 60%, 60% |
| $M_\infty$ | global percent cover maximum (for all $P_k$ on a given cell) | 80% |
| $\omega_L$ | minimum wind speed required for sediment transport | 6 meters/second |
| $\omega_H$ | threshold for storm event | 16 meters/second |
| $\alpha_k$ | erosion coefficient for each $P_k$, $k = 1, 2, 3, 4$ | 0.667, 0.333, 1, 1 |
| $\theta_o$ | angle of repose for avalanche | $\pi/6$ |
| $R_o$ | initial rate of shoreline retreat | $15\ m/yr$ |
| $\theta_o$ | angle of repose for avalanche | $\pi/6$ |
| $2w$ | width of transectional window to calculate $X_p$ | 10 cells |
| $M_a$ | migration rate growth factor | 0, 1, 1.0075 |

Table A.1: Model constant and parameter notion and values,
* values are for Smith, Parramore respectively.

| Notation | Variable definition |
|---|---|
| $H$ | array containing island elevation values in number of slabs |
| $P_k$ | array containing percent cover values for plant populations $(k = 1, 2, 3, 4)$ |
| $PC$ | total percent cover for all $P_k$ weighted by erosion coefficients, $\alpha_k$ |
| $a$ | percent cover on cell in excess of $M_\infty$ |
| $l$ | number of grass species on cell ($l \in \{1, 2, 3\}$) |
| $\theta'$ | current cell angle of elevation with respect to a neighboring cell |
| $\rho_{av}$ | probability of avalanching |
| $\rho_{e_d}$ | probability of aeolian transport at step $d$ |
| $d$ | aeolian transport distance in cell widths ($d \in \{1, 2, 3\}$) |
| $\omega$ | wind speed |
| $R$ | rate of shoreline retreat (Bruun Rule) |
| $R'$ | accumulated shoreline retreat vector |
| $A^*$ | cross-shore distance to depth of closure (Bruun Rule) |
| $h^*$ | depth of closure (Bruun Rule) |
| $B$ | berm height (Bruun Rule) |
| $\theta_b$ | approximate angle of foreshore slope (Bruun Rule) |
| $j_s$ | horizontal location of the shoreline for row $i$ |
| $X_p$ | migration rate reduction factor |
| $\Psi$ | area of subaerial portion of island within transectional window |

Table A.2: Model arrays and variables

# Appendix B

# MATLAB code

## B.1 Main code

```
%% MainCode092820
% MPswitch=1;
% ATcntr=0;
% SLRswitch=0;
% is the updated main code for barrier island evolution.
% this version of the code is build upon the original code by Greg Robson

% off-hand key notes (NOT EVEN REMOTELY COMPREHENSIVE)
%   **ARRAYS
%       H - main elevation matrix
%           (only updated in initialization and at end of the main loop)
%       Hstar - dummy elevation array for changes in subroutines
%           *third dimension unused - was for land categorization
%       P_i~Plant perent coverage matrices,
%       P3d - tracks dead morella
%           ~Pi(:,:,1)  current cell percent cover
%           ~Pi(:,:,2)  initial elevation of current cell
%               (currently unused, will be needed to check for burial)
%       PC - Combine Pi's for a total percent cover array - (no cell >1)
%             (Pi stores value -999 in cells without that plant - i.e.
%             water, so PC is the array with only cells greater than 0
%       Ptot= Sun of all PCi
%       W - water table data (currently unused)
%       S - salinity (?) data (currently unused)
%   **General PARAMETERS
%       time = number of iterations (two week time steps)
%       delta - slab heigh (meters)
%       L - slab length and width (meters)

%   **Routines and Governing Parameters
%     Main Code
%       t - current time step
```

```
%       (2) AeolianTransport
%          Pe - prob. of erosion
%          Pd - prob. deposition
%          n - number of slabs which can move due to wind
%      (3) Plant Propogation
%          currently runs twice per year
%          first time ignores morella and is meant to simulate springtime
%              growth for grasses (P1,P2,P4 - allow them more chance to grow)
%          PiIC - Plant initial percent cover for P1,P2,P3,P4
%          PiErosionCoefficient - factors into AT
%          P-Burial - number of slabs until death

%      (4) Marine Processes
%          currently runs once every 3 months
%
%       avalanche:
%              whole domain (underwater and island) ever 5th year time step
%              once per year runs THREE times per step over only subaerial (ground)
%                  after SwampProcess, AeolianTransport, and before loop end
%                  otherwise runs once per time step, before loop end
%
%    This version contains NO use of Land Categorization
tic

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                            %%%%%LOAD ELEVATION MATRIX%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% PARRAMORE ISLAND DATA:%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% IslandArea=14400000;
% 12km x 1200 m = 14400000 - Relative Influence Antecendent... paper, 12.4 m/yr
filename='Parramore03312021.mat'
IslandArea=19070000; %for parramore
data=importdata(filename);
H=data;                              %H will be the main elevation matrix
TranChk=1;



% filename='EricsIsland.txt'
% IslandArea=41000;% EricsIsland

%%%%%%%%%%%%%%%%%%%%%%%%
% SMITH ISLAND DATA:%
%%%%%%%%%%%%%%%%%%%%%%%%
% filename='Smith03312021.mat'
% IslandArea=9065000; %for Smith
% data=importdata(filename);
% H=data;                              %H will be the main elevation matrix
% TranChk=2;

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


ScaleFactor=floor(sqrt(IslandArea/(sum(sum(H(:,:)>0)))));
```

```
% InterpFactor=.75*sqrt(ScaleFactor);   %ScaleFactor 2
 InterpFactor=0.5*sqrt(ScaleFactor);     %ScaleFactor 4

H=interp2(H,InterpFactor);
 ScaleFactor=floor(sqrt(IslandArea/(sum(sum(H(:,:)>0)))));
 if TranChk==1
     TranRow=floor(.25*size(H,1));   %choose row for transect for parramore
 elseif TranChk==2
     TranRow=floor(0.75*size(H,1));    %choose row for transect for smith
 elseif exists(TranChk)==0
     TranRow=floor(0.5*size(H,1));
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%END - LOAD ELEVATION MATRIX%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                    %%%%%     PARAMETERS      %%%%%
                                                                                    %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
                                                                                    %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 fprintf('\n')
 fprintf('Initializing...')
           %cheat sheet:
 time=702;  %2600~100yrs;%1300~50yrs;%780~30yrs;%520~20yrs;260~10yrs;
 delta=0.1;
% L=1;%<----------ScaleFactor
 L=ScaleFactor;
 BchMax =1.5;   %greatest elevation that the beach spreads inland to
OW=0; %switch for overwashing - might use when we get storms incorporated (if windspeed>16m/s OW=1 for on, etc)
 BchW=10;    %unused - maximum beach width
 MPswitch=1;     %changes marine processes:    --(0 or any ~=1,2,3) for OFF
%                                              --(1) for ON {new version - update using same equil. slope}
     MarineProcesses03312021
%                                              --(2)for ON {old version - update equil. slope every 12 wks}
     OLDMarineProcesses03312021
 DepositSupply=3;     %number of slabs of sediment supplied to beach each time MarinePRocesses is run (3 months
 SLRswitch=0;    %turns sea level rise ON(1)/OFF(0 or any ~=1)
 SLRyrs=[ 8 7 6 5]; %years at which to perform SLR at rate SLR and/or migration...
 Ma= 1;%1.0075;       %shoreline migration growth factor, use 1 for no accel/noSLR
 MigAccel=0;     %will need to track magnitude of migration change
 MigYr=15; %initial yearly migration of the island in meters/year
 ScaleFactor=floor(sqrt(IslandArea/(sum(sum(H(:,:,1)>0)))));
% R=zeros(size(H,1),1);
 MaxSwampWidth=1000; %how far the swamp should go out into the water - just have to make something up for now

 WindData=1; %0~steady windspeed/direction ,1~windspeed and direction from empirical data

 if WindData==0
     Windspeed=16;
     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
     WindDir=1;  %wind direction           %  1=N  2=NE  3=E  4=SE  5=S  6=SW  7=W  8=NW  %
                                                                          %
     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
end
Windmin=6;
StormThreshold=16;
ATcntr=0; %testing variable - to count number of times AT is called

%%%%%%%%%%%%%%%%%%%%%%%%    PLANT PARAMETERS    %%%%%%%%%%%%%%%%%%%%%%%%%%%
%Plant initial conditions:
PlantRangeArray=[1 5;0.75 3;1.5 2.5;-0.5 0]; %all of the elevation ranges  for p1-p4
P1IC=1.0;
P2IC=1.0;
P3IC=1.0;
P4IC=1.0;
P1PctMax=.6;                    %largest percentage we will allow any plant population on a given cell to attain
P2PctMax=.6;
P3PctMax=.8;
P4PctMax=.8;
PctMax=[P1PctMax P2PctMax P3PctMax P4PctMax];
MasterMax=1.0;% The most any cell can permit - 80% plant coverage
KillSwitch=0;   %this will kill plants on the bottom half of the island if set to 1, set to 0 (or anything else) to turn off
alpha=.01; %propagation rate for each populated cell
DBE=.3;      %death by elevation rate for each populated cell outside of plant's elevation range
gdrange1=[-.02:.01:.08]; %range of percent values for growth/death for plant populations at (0, 50)% cover
gdrange2=[-.02:.01:.08];%[-.04:.01:.04]; %range of percent values for growth/death for plant pops greater than 50% cover

%Elevation Matrix dependent parameters:
ROW=size(H,1);
COLUMN=size(H,2);
H(:,:,2)=zeros(ROW,COLUMN);       %Creating extra dimension for H
Hstar=H;            %initializing dummy matrix
MigCnt=zeros(ROW,1);   %used to store how many meters the shoreline should have receded by, shoreline moves when MigCnt>
    ScalingFactor
M=max(max(H(:,:,1)));
m=min(min(H(:,:,1)));
W=zeros(ROW,COLUMN);  %do not comment out - needed as input
S=zeros(ROW,COLUMN);  %do not comment out - needed as input

%parameters unused so far (check with Greg)
% numslabg=0;
% q=0;
% Salt=1;
% ps=0.6;
AV_ARRAY=zeros(time,1000,3);          %tracks cell movement in AV. routine
%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%% Imaging Options %%%%%

%  ***note that we do not currently have a running land categorization
%      subroutine, so leave those options commented or at 0 - I think this
%      will eventually come back into use so I am leaving the lines there***

DurImFreq=78;        %how often to output "during" elevation images (26~yearly, 130~every 5 years)
    MElevCont=0;
    SElevCont=0;
PlantDurImFreq=234; %same, just for plant images
DurCUImFreq=78;

```

```
ElevImCU_during=1;
Contour27=1;        %contours at 0, 13, and 27 years
DurContour=0;       %images of the island outline

TransectIm=1;       %images of 2D island transect
    TranImFreq=234; %every 9 years
    if TransectIm==1 && exist('TranRow') == 0
        TranRow=floor(.5*size(H,1));   %choose row for transect
    end

MnPlantCvrIm=1;      %mean percent cover for each plant, displayed at end of routine 1 to turn on, any other number to turn off
ElevIm_before=1;        ElevIm_during=1;        ElevIm_after=0;
LandCatIm_before=0;     LandCatIm_during=0;     LandCatIm_after=0;          %leave these at 0 until we have a working LandCategorization routine
PlantPCIm_before=0;     PlantPCIm_during=0;     PlantPCIm_after=0;
SingleStepICIm=0;   %plots from initialization process :
%includes elevation and Plaint initial conditions for
%one pass through Swamp, PlantPropagation and
%AeolionTransport
climsH=[min(min(H(:,:,1))) max(max(H(:,:,1)))];
climsP=[-1 1];
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%              %%%%%%Initializing Plant Arrays%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%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

%GRASS #2  (Spartina)
P2=zeros(ROW,COLUMN,2);
P2Burial=0.5;           %meters of burial until death
P2ErosionCoefficient=(1/3);

%SHRUB #1  (Morella)(bird-dispersed seeds)
P3=zeros(ROW,COLUMN,2);
P3Burial=1.5;           %meters of burial until death
P3ErosionCoefficient=1;

%dead morella will track when morella reaches death and store dead
%debris data in P3d(:,:,1) for some number of years which will be counted in
%P3d(:,:,2) - this is only updated inside of PlantProp
P3d=zeros(ROW,COLUMN,2);
P3dErosionCoefficient=1;

%SECOND TYPE OF SPARTINA!!!
P4=zeros(ROW,COLUMN,2);
P4Burial=0.5;
P4ErosionCoefficient=(1/3);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% 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 some% 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.
 fprintf('\n')
 fprintf('Seeding the island...')
PC1=zeros(size(H,1),size(H,2));
PC2=zeros(size(H,1),size(H,2));
PC3=zeros(size(H,1),size(H,2));
PC4=zeros(size(H,1),size(H,2));
 [Hstar,MeanBeachWidth,ESL,WSL,MESL,MWSL,OL]=Shoreline03312021(Hstar,delta,L,BchMax);%function to return shoreline(s)
 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)=P1IC;
             P1(i,j,2)=H(i,j,1);
         elseif (H(i,j,1)-W(i,j))<0
             P1(i,j,1)=-999;
             P1(i,j,2)=0;
         end
         PC1(i,j)=P1(i,j,1)*(P1(i,j,1)>0);
         if (delta*H(i,j,1))>=0.25 && (delta*H(i,j,1))<=3
             P2(i,j,1)=P2IC;
             P2(i,j,2)=H(i,j,1);
         elseif (H(i,j,1)-W(i,j))<0
             P2(i,j,1)=-999;
             P2(i,j,2)=0;%
         end
         PC2(i,j)=P2(i,j,1)*(P2(i,j,1)>0);
         if (delta*H(i,j,1))>=1.5 && (delta*H(i,j,1))<=2.5
             P3(i,j,1)=P3IC;
             P3(i,j,2)=H(i,j,1);
         elseif (H(i,j,1)-W(i,j))<0
             P3(i,j,1)=-999;
             P3(i,j,2)=0;
         end
         PC3(i,j)=P3(i,j,1)*(P3(i,j,1)>0);
         if (delta*H(i,j,1))>=-0.5 && (delta*H(i,j,1))<=1
             if ESL(i)~=0
                 if MWSL(i)<j && MESL(i)>=j
                     if rand<0.5
                         P4(i,j,1)=P4IC;
                         P4(i,j,2)=H(i,j,1);
                     end
                 else
                     P4(i,j,1)=0;
                     P4(i,j,2)=0;
                 end
             end
%                WesternCells=zeros(1,MaxSwampWidth);
%                EasternCells=zeros(1,MaxSwampWidth);
%                for mm=1:min(j-1,size(WesternCells,2))
%                    WesternCells(mm)=delta*H(i,j-mm,1);
```

```
%               end
%               for mm=1:min(size(EasternCells,2),size(H,2)-j)
%                   EasternCells(mm)=delta*H(i,j+mm,1);
%               end
%               CheckWesternCells=WesternCells<=-.5;%must be within MaxSwampWidth cells to the west of a cell which is outside of marsh range
%               CheckEasternCells=EasternCells>0;
%               if sum(CheckWesternCells)==0 || sum(CheckEasternCells)==0
%                   P4(i,j,1)=-999;
%                   P4(i,j,2)=0;
%               end%
%               if sum(CheckWesternCells)~=0 &&.5<rand
%                   P4(i,j,1)=P4IC;
%                   P4(i,j,2)=H(i,j,1);%
%               end
        elseif (delta*H(i,j,1)-W(i,j))<-0.5 || (delta*H(i,j,1)-W(i,j))>1
                if (delta*H(i,j,1)<PlantRangeArray(4,1)) %if less than min height (-0.5)
                    P4(i,j,1)=-999;
                    P4(i,j,2)=0;
                elseif (delta*H(i,j,1)>PlantRangeArray(4,2)) %elseif greater than max height, make 0 (no death by elev. just kill)
                    P4(i,j,1)=0;
                    P4(i,j,2)=0;
                end
        end
        PC4(i,j)=P4(i,j,1)*(P4(i,j,1)>0);
    end
end
t=NaN;
[P1,P2,P3,P4,P3d]=PlantPropagation03312021(Hstar,t,P1,P2,P3,P4,W,S,delta,P3d,MaxSwampWidth,PlantRangeArray,alpha,DBE,gdrange1,gdrange2,PctMax,MasterMax,MWSL,MESL,ESL);
    for i=1:size(H,1)
        for j=1:size(H,2)
            PC1(i,j)=P1(i,j,1)*(P1(i,j,1)>0);
            PC2(i,j)=P2(i,j,1)*(P2(i,j,1)>0);                                    %
            PC3(i,j)=P3(i,j,1)*(P3(i,j,1)>0);
            PC4(i,j)=P4(i,j,1)*(P4(i,j,1)>0);
        end
    end
PC=(P1ErosionCoefficient.*PC1(:,:))+(P2ErosionCoefficient.*PC2(:,:))+(P3ErosionCoefficient.*PC3(:,:))+(P4ErosionCoefficient.*PC4(:,:))+(P3dErosionCoefficient.*P3d(:,:,1));
Ptot=P1(:,:,1)+P2(:,:,1)+P3(:,:,1)+P4(:,:,1)+P3d(:,:,1);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%END - INITIALIZING PLANT ARRAYS%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%killing plants on half of island test%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if KillSwitch==1;
    for i=floor(0.5*(size(H,1))):size(H,1)    %killing bottom half plants
        for j=1:size(H,2)
            if P1(i,j,1)>0
                P1(i,j,1)=0;
                PC1(i,j)=0;
            end
```

```
                if P2(i,j,1)>0
                    P2(i,j,1)=0;
                    PC2(i,j)=0;
                end
                if P3(i,j,1)>0
                    P3(i,j,1)=0;
                    PC3(i,j)=0;
                end
                if P4(i,j,1)>0
                    P4(i,j,1)=0;
                    PC4(i,j)=0;
                end
            end
        end
    PC=(P1ErosionCoefficient.*PC1(:,:))+(P2ErosionCoefficient.*PC2(:,:))+(P3ErosionCoefficient.*PC3(:,:))+(P4ErosionCoefficient.*
        PC4(:,:))+(P3dErosionCoefficient.*P3d(:,:,1));
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%END of half island plant test conditions%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%                      %%%%%%%%%BEFORE IMAGES%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

if ElevIm_before==1
    Hfilt=round(imgaussfilt(H(:,:,1),2),0);
    %%%%%%%%%Elevation%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    figure                                              %
    colormap(jet())                                     %
    imagesc(H(:,:,1),climsH)                            %
    hold on
    if MElevCont==1
    contour(H(:,:,1),[-5 -5],'color','m','linewidth',1.1)
    end
    if SElevCont==1
    contour(Hfilt(:,:,1),[-0 -0],'color','k','linewidth',1.1)
    end
    if MElevCont==1 && SElevCont==1
        lgnd=legend('\color{white} marsh contour (-0.5m)','\color{white} sea level contour (0m)');
        set(lgnd,'color','none','location','southeast');
    elseif MElevCont==1 && SElevCont==0
        lgnd=legend('\color{white} marsh contour (-0.5m)');
        set(lgnd,'color','none','location','southeast');
    elseif SElevCont==1 && MElevCont==0
        lgnd=legend('\color{white} sea level contour (0m)');
        set(lgnd,'color','none','location','southeast');
    end
    colorbar                                            %
    title('Before Image');                              %
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
end

if PlantPCIm_before==1
    %%%%%%%PLANTS%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %% AMMOPHILA
    figure
```

```
    colormap gray
    imagesc(P1(:,:,1),climsP)
    colorbar
    title('P1 - Ammophila IC')
    %%% SPARTINA
    figure
    colormap gray
    imagesc(P2(:,:,1),climsP)
    colorbar
    title('P2 - Spartina patens IC')
    %%% MORELLA
    figure
    colormap gray
    imagesc(P3(:,:,1),climsP)
    colorbar
    title('P3 - Morella IC')
    %%% SPARTINA (marsh)
    figure
    colormap gray
    imagesc(P4(:,:,1),climsP)
    colorbar
    title('P4 - Spartina alterniflora IC')
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%Single Step Subroutine Images%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
fprintf('\n')
fprintf('Running startup sequences:')
fprintf('\n')
fprintf('Running Swamp Processes...')
%Initial - Run Swamp
    [Hstar,ColumnArraySwamp1,ColumnArraySwamp2,AdjascentLengthSwamp,OppositeLocationSwamp,flag,INum]=SwampProcesses03312021(Hstar,
        delta,L,flag,PC);
fprintf('done')
fprintf('\n')
fprintf('Running Avalanche...')
% (1.5) post swamp avalache
PassCount=1;
CellsMoved=zeros(1,1000);
[Hstar,flag,CellCt]=AVALANCHE03312021(Hstar,delta,L,flag,PC);
CellsMoved(PassCount)=CellCt;
while flag==1
    PassCount=PassCount+1;
    [Hstar,flag,CellCt]=AVALANCHE03312021(Hstar,delta,L,flag,PC);
    CellsMoved(PassCount)=CellCt;%
end
CellsMoved=CellsMoved(1,1:PassCount);
SwampAvArray=[NaN PassCount CellsMoved];     %tracking number of passes and number of cells moved per pass
AvIC_Array1=SwampAvArray;                     %AvIC_Array1 tracks avalanche data for initialization routines
H=Hstar;                                          %update H
fprintf('done')


%****************** NOT USING LAND CATEGORIZATION - BUT MAY AGAIN LATER - DO NOT DELETE!!
for i=1:size(Hstar,1)
    for Icount=1:INum
```

```
            if ColumnArraySwamp2(i,Icount)~=0
%                   for j=ColumnArraySwamp1(i,Icount):ColumnArraySwamp2(i,Icount)
%                       Hstar(i,j,2)=2;
%                   end
            end
        end
end
%****************** NOT USING LAND CATEGORIZATION - BUT MAY AGAIN LATER - DO NOT DELETE!!

%post swamp processes/avalanche1 images
if SingleStepICIm==1
    figure('NumberTitle','off','Name','After Swamp');
    colormap(jet())          %
    %     clims=[min(min(Hstar(:,:,1))) max(max(Hstar(:,:,1)))];            %
    imagesc(Hstar(:,:,1),climsH)%
    title('ICs After Swamp')
    colorbar
end
fprintf('\n')
fprintf('Running Marine Processes...')
%Initial - Marine Processes
MeanBeachWidthVec=zeros(1,time+2);  %for tracking beach width evolution
if MPswitch==1
    % need version of PC with erosion coefficients AND with negative values (zeros only where plants COULD grow) to calculate migration
    PCmp=-999*ones(ROW,COLUMN);
    for i=1:ROW
        for j=1:COLUMN
            if P1(i,j,1)>=0 || P2(i,j,1)>=0 ||P3(i,j,1)>=0 ||P4(i,j,1)>=0
                PCmp(i,j)=(P1ErosionCoefficient*P1(i,j,1)*(P1(i,j,1)>0))+(P2ErosionCoefficient*P2(i,j,1)*(P2(i,j,1)>0))+(P3ErosionCoefficient*P3(i,j,1)*(P3(i,j,1)>0));%+(P4ErosionCoefficient*P4(i,j,1)*(P4(i,j,1)>0));
            end
        end
    end
%     Pt1=max(P1(:,:,1),0);
%     Pt2=max(P2(:,:,1),0);
%     Pt3=max(P3(:,:,1),0);
%     Pt4=max(P4(:,:,1),0);
%     PCmp=(P1ErosionCoefficient.*Pt1)+(P2ErosionCoefficient.*Pt2)+(P3ErosionCoefficient.*Pt3)+(P4ErosionCoefficient.*Pt4);
    [Hstar,P3,MeanBeachWidth,ESL,WSL,MESL,MWSL,OppositeLocation,SLRyrs,MigCnt,MigAccel]=MarineProcesses03312021(Hstar,delta,L,P1,P2,P3,P4,BchMax,OW,t,SLRyrs,PCmp,IslandArea,MigCnt,ScaleFactor,MigYr,Ma,MigAccel,MasterMax);
    MeanBeachWidthVec(1)=MeanBeachWidth;
    H=Hstar;
elseif MPswitch==2
    [Hstar,PlantColumnArray,PlantColumnArray2,P3,flag,MeanBeachWidth]=OLDMarineProcesses03312021(Hstar,delta,L,flag,PC,P3,BchMax,BchW);
    MeanBeachWidthVec(1)=MeanBeachWidth;
    H=Hstar;
end
fprintf('done')

if SingleStepICIm==1
    figure('NumberTitle','off','Name','After Marine Processes');
    colormap(jet())          %
    imagesc(Hstar(:,:,1),climsH)%
    title('ICs after Swamp, Marine Processes')
```

```
    colorbar
end

%Redeclare PC since P3 has been updated
%PC=(P1ErosionCoefficient.*P1(:,:,1))+(P2ErosionCoefficient.*P2(:,:,1))+(P3ErosionCoefficient.*P3(:,:,1))+(P4ErosionCoefficient.*
    P4(:,:,1));                                                    %
for i=1:size(H,1)
    for j=1:size(H,2)
        PC1(i,j)=P1(i,j,1)*(P1(i,j,1)>0);
        PC2(i,j)=P2(i,j,1)*(P2(i,j,1)>0);                                                                   %
        PC3(i,j)=P3(i,j,1)*(P3(i,j,1)>0);
        PC4(i,j)=P4(i,j,1)*(P4(i,j,1)>0);
    end
end
PC=(P1ErosionCoefficient.*PC1(:,:))+(P2ErosionCoefficient.*PC2(:,:))+(P3ErosionCoefficient.*PC3(:,:))+(P4ErosionCoefficient.*PC4
    (:,:))+(P3dErosionCoefficient.*P3d(:,:,1));

%post marine processes avalanche
PassCount=1;
CellsMoved=zeros(1,1000);
[Hstar,flag,CellCt]=AVALANCHE03312021(Hstar,delta,L,flag,PC);
CellsMoved(PassCount)=CellCt;
while flag==1
    PassCount=PassCount+1;
    [Hstar,flag,CellCt]=AVALANCHE03312021(Hstar,delta,L,flag,PC);
    CellsMoved(PassCount)=CellCt;%
end
CellsMoved=CellsMoved(1,1:PassCount);
AvIC_Array2=[NaN PassCount CellsMoved];
H=Hstar;

%****************** NOT USING LAND CATEGORIZATION - BUT MAY AGAIN LATER - DO NOT DELETE!!
%     %Initial Declare all categories in H
PlantColumnArray=zeros([size(H,1) 1]); %P3 isn't allowed to grow on the beach - this is to check for that
PlantColumnArray2=zeros([size(H,1) 1]);
[Hstar,P1,P2,P3,P4]=LandCategorizationUNUSED03312021(Hstar,P1,P2,P3,P4,PlantColumnArray,PlantColumnArray2,ColumnArraySwamp1,
    ColumnArraySwamp2,INum);
%****************** NOT USING LAND CATEGORIZATION - BUT MAY AGAIN LATER - DO NOT DELETE!!

if SingleStepICIm==1
    figure
    colormap(jet())
    imagesc(Hstar(:,:,2),climsH)
    colorbar
    ('ICs LandCats - After Swamp, Marine Processes and Avalanche');
end
fprintf('\n')
fprintf('Initialization complete')

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%END INITIAL PARAMETERIZATION%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
tic
T=zeros(time,1);
%########################################################################%
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%!!@@##$$                %%%%%%%%MAIN LOOP%%%%%%%%                $$##@@!!%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%########################################################################
tStart=cputime;
for t=0:time
    TimeStep=t
    Ptot=P1(:,:,1)+P2(:,:,1)+P3(:,:,1)+P4(:,:,1)+P3d(:,:,1);
    if 0==mod(t,26)
        if t==0
            fprintf('\n')
            fprintf('Here we go...')
        elseif t==26
            fprintf('\n')
            fprintf('%d year has passed',t/26)
        else
            fprintf('\n')
            fprintf('%d years have passed',t/26)
        end
    end
    %     TimeStep=t
    %%    %PC=(P1ErosionCoefficient.*P1(:,:,1))+(P2ErosionCoefficient.*P2(:,:,1))+(P3ErosionCoefficient.*P3(:,:,1))+(P4ErosionCoefficient.*P4(:,:,1));
    %need to work this into the inside of PlantProcesses
    for i=1:size(H,1)
        for j=1:size(H,2)
            PC1(i,j)=P1(i,j,1)*(P1(i,j,1)>0);
            PC2(i,j)=P2(i,j,1)*(P2(i,j,1)>0);                                                     %
            PC3(i,j)=P3(i,j,1)*(P3(i,j,1)>0);
            PC4(i,j)=P4(i,j,1)*(P4(i,j,1)>0);
        end
    end
    PC=(P1ErosionCoefficient.*PC1(:,:))+(P2ErosionCoefficient.*PC2(:,:))+(P3ErosionCoefficient.*PC3(:,:))+(P4ErosionCoefficient.*PC4(:,:))+(P3dErosionCoefficient.*P3d(:,:,1));

    %%    %(1) Run Swamp


    [Hstar,ColumnArraySwamp1,ColumnArraySwamp2,AdjascentLengthSwamp,OppositeLocationSwamp,flag,INum]=SwampProcesses03312021(Hstar,delta,L,flag,PC);

    %%    %(1.AV)            THIS AVALANCHE RUNS ONCE PER YEAR (every 26 time steps) AND IS THE FIRST AVALANCHE
    % if (0==mod(t,26)) && t~=0
    PassCount=1;
    CellsMoved=zeros(1,1000);
    [Hstar,flag,CellCt]=AVALANCHEtime03312021(Hstar,delta,L,flag,PC,t);
    CellsMoved(PassCount)=CellCt;
    while flag==1
        PassCount=PassCount+1;
        [Hstar,flag,CellCt]=AVALANCHEtime03312021(Hstar,delta,L,flag,PC,t);
        CellsMoved(PassCount)=CellCt;%
        if CellCt<=5
            flag=0;
        end
    end
    CellsMoved=CellsMoved(1,1:PassCount);
```

```
SwampAvArray=[NaN PassCount CellsMoved];
AvIC_Array1=SwampAvArray;
AV_ARRAY(TimeStep+1,1:PassCount+2,1)=SwampAvArray;


%****************** NOT USING LAND CATEGORIZATION - BUT MAY AGAIN LATER - DO NOT DELETE!!
for i=1:size(Hstar,1)
    for Icount=1:INum
        if ColumnArraySwamp2(i,Icount)~=0
            for j=ColumnArraySwamp1(i,Icount):ColumnArraySwamp2(i,Icount)
                Hstar(i,j,2)=2;
            end
        end
    end
end
%****************** NOT USING LAND CATEGORIZATION - BUT MAY AGAIN LATER - DO NOT DELETE!!


%%    %(2) Aeolian Transport

if WindData==0
    if Windspeed>=6
        WindCnt=floor(((max(Windspeed-Windmin,0))/(StormThreshold-Windmin))*24);
        for day=1:14
            %               for hour =1:WindCnt
                [Hstar]=AeolianTransport03212021(Hstar,delta,L,W,Windspeed,Windmin,StormThreshold,WindDir,PC);
            %               end
        end%
    end
end

if WindData==1 && t>0
    %WindDir=randi([1,8]);  can randomize wind or use data-weighted wind directions outlined below. (comment this if you randomize)
    for day=1:14
        WindSpeedParam=randi(2183);
        if WindSpeedParam>=1 && WindSpeedParam<=2116
            Windspeed=randi([0,5]);
        elseif WindSpeedParam>2116 && WindSpeedParam<=2183
            ATcntr=ATcntr+1;
            Windspeed=randi([6,15]);
        elseif WindSpeedParam>2183
            Windspeed=16;
        end
        WindParam=randi(2107);
        if WindParam>=1 && WindParam<=250
            WindDir=1;
        elseif WindParam>=251 && WindParam<=655
            WindDir=2;
        elseif WindParam>=656 && WindParam<=835
            WindDir=3;
        elseif WindParam>=836 && WindParam<=1091
            WindDir=4;
        elseif WindParam>=1092 && WindParam<=1464
            WindDir=5;
        elseif WindParam>=1465 && WindParam<=1717
```

```
                WindDir=6;
            elseif WindParam>=1718 && WindParam<=1865
                WindDir=7;
            elseif WindParam>=1866 && WindParam<=2107
                WindDir=8;
            end

            if Windspeed>=6
%                   WindCnt=floor(((max(Windspeed-Windmin,0))/(StormThreshold-Windmin))*24);
%                   for hour=1:WindCnt
                        [Hstar]=AeolianTransport03312021(Hstar,delta,L,W,Windspeed,Windmin,StormThreshold,WindDir,PC);
%                   end
            end
        end
    end

    %%   %(2.AV)              THIS AVALANCHE RUNS ONCE PER YEAR (every 26 time steps) AND IS THE SECOND AVALANCHE
    % if (0==mod(t,26)) && t~=0
    PassCount=1;
    CellsMoved=zeros(1,1000);
    [Hstar,flag,CellCt]=AVALANCHEtime03312021(Hstar,delta,L,flag,PC,t);
    CellsMoved(PassCount)=CellCt;
    while flag==1
        PassCount=PassCount+1;
        [Hstar,flag,CellCt]=AVALANCHEtime03312021(Hstar,delta,L,flag,PC,t);
        CellsMoved(PassCount)=CellCt;%
        %           if CellCt<=5
        %               flag=0;
        %           end
    end
    CellsMoved=CellsMoved(1,1:PassCount);
    Av2_Array=[TimeStep PassCount CellsMoved];
    AV_ARRAY(TimeStep+1,1:PassCount+2,2)=Av2_Array;
    % end

    %%   %(3) Plant Propogation

%land categroization used to be required for plant processes - this might be removeable
    %****************** NOT USING LAND CATEGORIZATION - BUT MAY AGAIN LATER - DO NOT DELETE!!
    [Hstar,P1,P2,P3,P4]=LandCategorizationUNUSED03312021(Hstar,P1,P2,P3,P4,PlantColumnArray,PlantColumnArray2,ColumnArraySwamp1,ColumnArraySwamp2,INum);
    %****************** NOT USING LAND CATEGORIZATION - BUT MAY AGAIN LATER - DO NOT DELETE!!

    if (0==mod(t,13))  %  choose 26 for 2-week timesteps;   52 for 1-week timesteps
        [Hstar,MeanBeachWidth,ESL,WSL,MESL,MWSL,OL]=Shoreline03312021(Hstar,delta,L,BchMax);%get shoreline to use in PlantPropagation
        [P1,P2,P3,P4,P3d]=PlantPropagation03312021(Hstar,t,P1,P2,P3,P4,W,S,delta,P3d,MaxSwampWidth,PlantRangeArray,alpha,DBE,gdrange1,gdrange2,PctMax,MasterMax,MWSL,MESL,ESL);
        if P3IC==0 %killing off any morella that was established due to bird activity
            P3(:,:,1)=min(P3(:,:,1),0);
        end
    end

    for i=1:size(H,1)
        for j=1:size(H,2)
            PC1(i,j)=P1(i,j,1)*(P1(i,j,1)>0);
```

```
            PC2(i,j)=P2(i,j,1)*(P2(i,j,1)>0);                                                   %
            PC3(i,j)=P3(i,j,1)*(P3(i,j,1)>0);
            PC4(i,j)=P4(i,j,1)*(P4(i,j,1)>0);
        end
    end
    PC=(P1ErosionCoefficient.*PC1(:,:))+(P2ErosionCoefficient.*PC2(:,:))+(P3ErosionCoefficient.*PC3(:,:))+(P4ErosionCoefficient.*PC4(:,:))+(P3dErosionCoefficient.*P3d(:,:,1));
    Ptot=PC1(:,:,1)+PC2(:,:,1)+PC3(:,:,1)+PC4(:,:,1)+P3d(:,:,1);

    %%   %(4) Marine Processes - edits
    %currently every 12 months, moves 2 slabs into ocean
    if (0==mod(t,26))
        if MPswitch==1
            PCmp=-999*ones(ROW,COLUMN);
            for i=1:ROW
                for j=1:COLUMN
                    if P1(i,j,1)>=0 || P2(i,j,1)>=0 ||P3(i,j,1)>=0 ||P4(i,j,1)>=0
                        PCmp(i,j)=(P1ErosionCoefficient*P1(i,j,1)*(P1(i,j,1)>0))+(P2ErosionCoefficient*P2(i,j,1)*(P2(i,j,1)>0))+(P3ErosionCoefficient*P3(i,j,1)*(P3(i,j,1)>0));%+(P4ErosionCoefficient*P4(i,j,1)*(P4(i,j,1)>0));
                    end
                end
            end
            %           PCmp=(P1ErosionCoefficient.*P1(:,:,1))+(P2ErosionCoefficient.*P2(:,:,1))+(P3ErosionCoefficient.*P3(:,:,1))+(P4ErosionCoefficient.*P4(:,:,1))+(P3dErosionCoefficient.*P3d(:,:,1));
            [Hstar,P3,MeanBeachWidth,ESL,WSL,MESL,MWSL,OL,SLRyrs,MigCnt,MigAccel]=MarineProcesses03312021(Hstar,delta,L,P1,P2,P3,P4,BchMax,OW,t,SLRyrs,PCmp,IslandArea,MigCnt,ScaleFactor,MigYr,Ma,MigAccel,MasterMax);
        elseif MPswitch==2
            [Hstar,PlantColumnArray,PlantColumnArray2,P3,flag,MeanBeachWidth]=OLDMarineProcesses03312021(Hstar,delta,L,flag,PC,P3,BchMax,BchW);
        end
        if MPswitch~=0
            MeanBeachWidthVec(1)=MeanBeachWidth;
            H=Hstar;
        end

    end

    %%  %(4.AV*)update PC before avalanching again
    %need to work this into the inside of PlantProcesses
    for i=1:size(H,1)
        for j=1:size(H,2)
            %       PC1(i,j)=P1(i,j,1)*(P1(i,j,1)>0);
            %       PC2(i,j)=P2(i,j,1)*(P2(i,j,1)>0);                                                %
            PC3(i,j)=P3(i,j,1)*(P3(i,j,1)>0);      %p3 is only thing that changes inside of marine processes - killed on beach
            %       PC4(i,j)=P4(i,j,1)*(P4(i,j,1)>0);
        end
    end
    PC=(P1ErosionCoefficient.*PC1(:,:))+(P2ErosionCoefficient.*PC2(:,:))+(P3ErosionCoefficient.*PC3(:,:))+(P4ErosionCoefficient.*PC4(:,:))+(P3dErosionCoefficient.*P3d(:,:,1));
    %%   %(4.AV) Avalanche after Marine Processes    THIS AVALANCHE RUNS EVERY TIME STEP AND IS THE FINAL AVALANCHE
    PassCount=1;
    CellsMoved=zeros(1,1000);
    [Hstar,flag,CellCt]=AVALANCHEtime03312021(Hstar,delta,L,flag,PC,t);
    CellsMoved(PassCount)=CellCt;
    while flag==1
        PassCount=PassCount+1;
```

```
            [Hstar,flag,CellCt]=AVALANCHEtime03312021(Hstar,delta,L,flag,PC,t);
            CellsMoved(PassCount)=CellCt;
        end
        CellsMoved=CellsMoved(1,1:PassCount);
        Av3_Array=[TimeStep PassCount CellsMoved];
        AV_ARRAY(TimeStep+1,1:PassCount+2,3)=Av3_Array;
        %%      %(5) Data and categories in H
        %***************** NOT USING LAND CATEGORIZATION - BUT MAY AGAIN LATER - DO NOT DELETE!!
        [Hstar,P1,P2,P3,P4]=LandCategorizationUNUSED03312021(Hstar,P1,P2,P3,P4,PlantColumnArray,PlantColumnArray2,ColumnArraySwamp1,
            ColumnArraySwamp2,INum);
        %***************** NOT USING LAND CATEGORIZATION - BUT MAY AGAIN LATER - DO NOT DELETE!!
        [Hstar,Centroid]=Qdata03312021(Hstar,P1,P2,P3,P4);%,ESL,WSL,MESL,MWSL,OppositeLocation);
        CentroidVec(t+1,:)=Centroid;

        %calculating mean percent cover for each Pi
        p1=P1(:,:,1)>0;p2=P2(:,:,1)>0;p3=P3(:,:,1)>0;p4=P4(:,:,1)>0;
        P1gt=P1(:,:,1).*p1;
        P2gt=P2(:,:,1).*p2;
        P3gt=P3(:,:,1).*p3;
        P4gt=P4(:,:,1).*p4;

        MnP1(t+2)=sum(P1gt,'all')/sum(p1,'all');
        MnP2(t+2)=sum(P2gt,'all')/sum(p2,'all');
        MnP3(t+2)=sum(P3gt,'all')/sum(p3,'all');
        MnP4(t+2)=sum(P4gt,'all')/sum(p4,'all');

        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        %%%%%                         Sea Level Rise                             %%%%%
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        if SLRswitch==1
            if 0==mod(t,156)%0==mod(t,156)    %156 timesteps = 312 weeks = 6 years
                %    if 0==mod(t,260)
                Hstar=Hstar-1; %working estimate for SLR in VA is 1.66cm/yr = .1m/6yrs
            end
        end
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        %%%%%                       End Sea Level Rise                           %%%%%
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

        H=round(Hstar,0);          %*****THIS IS THE ONLY UPDATE FOR H IN THE MAIN LOOP - OTHERWISE ALL UPDATES ARE MADE TO HSTAR

        if t==1
            Sv1=H(:,:,1);
            Sv1f=round(imgaussfilt(H(:,:,1),4),0);
            Sv1Cent=Centroid;
        elseif t==338
            Sv2=H(:,:,1);
            Sv2f=round(imgaussfilt(H(:,:,1),4),0);
            Sv2Cent=Centroid;
        elseif t==702
            Sv3=H(:,:,1);
            Sv3f=round(imgaussfilt(H(:,:,1),4),0);
            Sv3Cent=Centroid;
        end

        tEnd=cputime-tStart;
```

```
T(t+1)=toc;
fprintf('\n')
if t~=0
    Tnew=T(t+1)-T(t);
else
    Tnew=T(1);
end
fprintf('t=%d time: %0.1f',TimeStep,Tnew)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                  %%%%%%%%    During Images    %%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% % % % %ELEVATION

if ElevIm_during==1
    Hfilt=round(imgaussfilt(H(:,:,1),4),0);
    if (0==mod(t,DurImFreq))

           %
        figure

              %
        colormap(jet())
        imagesc(H(:,:,1),climsH);
        hold on
        if MElevCont==1
            contour(H(:,:,1),[-5 -5],'color','m','linewidth',1.1)
        end
        if SElevCont==1
            contour(Hfilt(:,:,1),[-0 -0],'color','k','linewidth',1.1)
        end
        if MElevCont==1 && SElevCont==1
            lgnd=legend('\color{white} marsh contour (-0.5m)','\color{white} sea level contour (0m)');
            set(lgnd,'color','none','location','southeast');
        elseif MElevCont==1 && SElevCont==0
            lgnd=legend('\color{white} marsh contour (-0.5m)');
            set(lgnd,'color','none','location','southeast');
        elseif SElevCont==1 && MElevCont==0
            lgnd=legend('\color{white} sea level contour (0m)');
            set(lgnd,'color','none','location','southeast');
        end
        colorbar
        title(sprintf('Elevation at %d years',t/26))
        hold off

    end
end

%This is the unsmoothed transect migration image
% % %     if TransectIm==1
% % %         if 0==mod(t,TranImFreq)
% % %             Trow=H(TranRow,:,1);
% % %             Trow(Trow<-5)=-6;
% % %             xlim1=find(Trow>=-5,1,'first')-100; xlim2=find(Trow>=-5,1,'last')+100;
```

```
%%%                     figure(1000)
%%%                     hold on
%%%                     plot(1:size(H,2),Trow,'Linewidth',2)
%%%                     xlim([xlim1 xlim2])
%%%                     ylim([-5,max(Trow)+20])
%%%                     title(sprintf('Island transect at row %d',TranRow))
%%%                     hold off
%%%                 end
%%%         end
        %%%Close up elev:
        if ElevImCU_during==1
            if (0==mod(t,DurCUImFreq))
                Wndw=250;
                MidRow=round(Centroid(2));
                MidCol=round(Centroid(1));
                figure
                imagesc(H(MidRow-Wndw:MidRow+Wndw,MidCol-Wndw:MidCol+Wndw,1),climsH);
                hold on
                ContourVec=[0:10:max(max(Hfilt))];
                colorbar
                contour(Hfilt(MidRow-Wndw:MidRow+Wndw,MidCol-Wndw:MidCol+Wndw,1),ContourVec);
                contour(Hfilt(MidRow-Wndw:MidRow+Wndw,MidCol-Wndw:MidCol+Wndw,1),ContourVec,'LineColor','k');
                [C,h]=contour(Hfilt(MidRow-Wndw:MidRow+Wndw,MidCol-Wndw:MidCol+Wndw,1),ContourVec,'LineColor','k');
                clabel(C,h)
                title(sprintf('Elevation Close-Up about Centroid at %d years',t/26))
%%% old one (lines at 10, 20 close up):
%               figure
    %
%               colormap(jet())
%               MidPt=round(CentroidVec(t+1,:));
%               MidCol=MidPt(1); MidRow=MidPt(2);
%               imagesc(H(MidRow-50:MidRow+50,MidCol-50:MidCol+50,1),climsH);
%               hold on
%               ContourVec=[0:5:max(max(Hfilt))];
%               contour(Hfilt(MidRow-50:MidRow+50,MidCol-50:MidCol+50,1),[10 10],'Color','m','linewidth',1.1);
%               contour(Hfilt(MidRow-50:MidRow+50,MidCol-50:MidCol+50,1),[20 20],'Color','k','linewidth',1.1);
%               lgnd=legend('\color{white} marsh contour (-0.5m)','\color{white} sea level contour (0m)');
%               set(lgnd,'color','none','location','southeast');
%               colorbar
%               title(sprintf('Elevation Close-Up about Centroid at %d years',t/26))
%               hold off


                hFig=figure;
                set(gcf,'PaperPositionMode','auto')
                set(hFig,'units','inches')
                hFig.Position(1)=.5;
                hFig.Position(2)=.5;
                hFig.Position(3)=2*hFig.Position(3);
                hFig.Position(4)=1.25*hFig.Position(4);
                subplot(1,2,1)
                colormap(jet())
                imagesc(H(MidRow-Wndw:MidRow+Wndw,MidCol-Wndw:MidCol+Wndw,1),climsH);
                title(sprintf('Elevation Close-Up about Centroid at %d years',t/26))
                subplot(1,2,2)
```

```
            [C,h] = contour(Hfilt(MidRow-Wndw:MidRow+Wndw,MidCol-Wndw:MidCol+Wndw,1),ContourVec);
            clabel(C,h)
            set(gca,'YDir','reverse')
            title(sprintf('Contours of Elevation Close-Up about Centroid at %d years',t/26))


            figure
            subplot(2,2,1)
            colormap bone
            imagesc(P1(MidRow-Wndw:MidRow+Wndw,MidCol-Wndw:MidCol+Wndw,1),climsP);
            title(sprintf('P_1 about centroid at %d years',t/26))
            subplot(2,2,2)
            colormap bone
            imagesc(P2(MidRow-Wndw:MidRow+Wndw,MidCol-Wndw:MidCol+Wndw,1),climsP);
            title(sprintf('P_2 about centroid at %d years',t/26))
            subplot(2,2,3)
            colormap bone
            imagesc(P3(MidRow-Wndw:MidRow+Wndw,MidCol-Wndw:MidCol+Wndw,1),climsP);
            title(sprintf('P_3 about centroid at %d years',t/26))
            subplot(2,2,4)
            colormap bone
            imagesc(P4(MidRow-Wndw:MidRow+Wndw,MidCol-Wndw:MidCol+Wndw,1),climsP);
           title(sprintf('P_4 about centroid at %d years',t/26))
        end
    end

    %%%%%Land Category

    if LandCatIm_during==1
        if (0==mod(t,DurImFreq)) %every 5 years

             %
            figure

                 %
            colormap(jet())
            clims=[min(min(H(:,:,2))) max(max(H(:,:,2)))];
            imagesc(H(:,:,2),clims);
            colorbar
            title(sprintf('Land Category at time step = %d',t))
        end
    end

    %%%%%Plant Percent Cover%

    if PlantPCIm_during==1
        if (0==mod(t,PlantDurImFreq))  %  2~/mo, choose 26~/year

            %%% AMMOPHILA
            figure
            colormap gray
            %         clims=[-.1 max(0,max(max(P1(:,:,1))))];
            imagesc(P1(:,:,1),climsP)
            colorbar
            title(sprintf('P1-Ammophila at %d years',t/26))
            %         title(sprintf('P1-Ammophila at time step = %d',t))
```

```
            %%% SPARTINA
            figure
            colormap gray
            %       clims=[-.1 max(0,max(max(P2(:,:,1))))];
            imagesc(P2(:,:,1),climsP)
            colorbar%
            title(sprintf('P2 - Spartina patens at %d years',t/26))
            %       title(sprintf('P2 - Spartina patens at time step = %d',t))
            %%% MORELLA
            figure
            colormap gray
            %       clims=[-.1 max(0,max(max(P3(:,:,1))))];
            imagesc(P3(:,:,1),climsP)
            colorbar%
            title(sprintf('P3 - Morella at %d years',t/26))
            %       title(sprintf('P3 - Morella at time step = %d',t))
            %%% DEAD MORELLA
            figure
            colormap gray
            %       clims=[-.1 max(0,max(max(P3d(:,:,1))))];
            imagesc(P3d(:,:,1),climsP)
            colorbar
            title(sprintf('P3d - DEAD Morella at %d years',t/26))
            %       title(sprintf('P3d - DEAD Morella at time step = %d',t))
            %%% SPARTINA (marsh)
            figure
            colormap gray
            %       clims=[-.1 max(1,max(max(P4(:,:,1))))];
            imagesc(P4(:,:,1),climsP)
            colorbar
            title(sprintf('P4 - Spartina alterniflora at %d years',t/26))
            %       title(sprintf('P4 - Spartina alterniflora at time step = %d',t))
        end

    end

end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%  END OF MAIN LOOP  %%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%     After Images     %%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%Elevation
if ElevIm_after==1
    figure
    colormap(jet())
    imagesc(H(:,:,1),climsH);
    hold on
    contour(H(:,:,1),[-0.5 -0.5],'color','black')
    colorbar
    title('After Image')
end

```

```
%Mean Plant Percent Cover Images
if MnPlantCvrIm==1
    figure
    hold on
    plot(MnP1)
    plot(MnP2)
    plot(MnP3)
    plot(MnP4)
    legend('P_1','P_2','P_3','P_4')
    xlabel('time in two week steps')
    ylabel('Mean Percent Cover')
    title(sprintf('Mean Percent Cover - %.0f years',time/26))
end

% % % % %Land Category

if LandCatIm_after==1
    figure
    colormap(jet())
    clims=[min(min(H(:,:,2))) max(max(H(:,:,2)))];
    imagesc(H(:,:,2),clims);
    colorbar
    title(sprintf('Land Category - After'))
end

% % % % %PLANTS (percent cover)

if PlantPCIm_after==1
    %%% AMMOPHILA
    figure
    colormap gray
    imagesc(P1(:,:,1),climsP)
    colorbar
    title(sprintf('P1-Ammophila After %d years',time/26))
    %%% SPARTINA
    figure
    colormap gray
    imagesc(P2(:,:,1),climsP)
    colorbar
    title(sprintf('P2 - Spartina patens after %d years',time/26))
    %%% MORELLA
    figure
    colormap gray
    imagesc(P3(:,:,1),climsP)
    colorbar%
    title(sprintf('P3 - Morella after %d years',time/26))
    %%% SPARTINA (marsh)
    figure
    colormap gray
    imagesc(P4(:,:,1),climsP)
    colorbar
    title(sprintf('P4 - Spartina alterniflora after %d years',time/26))
end

%% Contours at 0, 13, and 27 years
if Contour27==1
```

```
%Contours of the shoreline (0m elevation)
figure
    %
colormap(jet())
hold on
contour(Sv1f(:,:,1),[1 1],'Color','blue','LineWidth',2);          %beginning contour (after initialization)
contour(Sv2f(:,:,1),[1 1],'Color','green','LineWidth',2);         %contour at 13 years (t=338)
contour(Sv3f(:,:,1),[1 1],'Color','red','LineWidth',2);           %contour at 27 years (t=702)
scatter(Sv1Cent(1),Sv1Cent(2),'o','MarkerEdgeColor','k','MarkerFaceColor','b')
scatter(Sv2Cent(1),Sv2Cent(2),'o','MarkerEdgeColor','k','MarkerFaceColor','g')
scatter(Sv3Cent(1),Sv3Cent(2),'o','MarkerEdgeColor','k','MarkerFaceColor','r')
% scatter(Sv1Cent(1),Sv1Cent(2),'o','MarkerEdgeColor','w',...
%                                   'MarkerFaceColor','b',...
%                                   'LineWidth',1.25)
% scatter(Sv2Cent(1),Sv2Cent(2),'o','MarkerEdgeColor','w',...
%                                   'MarkerFaceColor','g',...
%                                   'LineWidth',1.25)
% scatter(Sv3Cent(1),Sv3Cent(2),'o','MarkerEdgeColor','w',...
%                                   'MarkerFaceColor','r',...
%                                   'LineWidth',1.25)
set(gca, 'YDir','reverse')
legend('Initial','13 years','27 years','Location','southeast')
title('Island Contours at 0m Elevation')

%contours including the marsh area:
%%% figure
    %
%%% colormap(jet())
%%% hold on
%%% contour(Sv1f(:,:,1),[-5 -5],'Color','blue','LineWidth',2);          %beginning contour (after initialization)
%%% contour(Sv2f(:,:,1),[-5 -5],'Color','green','LineWidth',2);         %contour at 13 years (t=338)
%%% contour(Sv3f(:,:,1),[-5 -5],'Color','red','LineWidth',2);           %contour at 27 years (t=702)
%%% scatter(Sv1Cent(1),Sv1Cent(2),'o','MarkerEdgeColor','k','MarkerFaceColor','b')
%%% scatter(Sv2Cent(1),Sv2Cent(2),'o','MarkerEdgeColor','k','MarkerFaceColor','g')
%%% scatter(Sv3Cent(1),Sv3Cent(2),'o','MarkerEdgeColor','k','MarkerFaceColor','r')
%%%% scatter(Sv3Cent(1),Sv3Cent(2),'*r')
%%%% scatter(Sv2Cent(1),Sv2Cent(2),'*g')
%%%% scatter(Sv1Cent(1),Sv1Cent(2),'*b')
%%% set(gca, 'YDir','reverse')
%%% legend('Initial','13 years','27 years','Location','southeast')
%%% title('Island Contours with Marshland (at -0.5m Elevation)')
end

if TransectIm==1
    figure
    Trow1=Sv1f(TranRow,:,1);
    Trow2=Sv2f(TranRow,:,1);
    Trow3=Sv3f(TranRow,:,1);
    Trow1(Trow1<-5)=-6; Trow2(Trow2<-5)=-6; Trow3(Trow3<-5)=-6;
    xlim1=find(Trow1>=-5,1,'first')-100; xlim2=find(Trow1>=-5,1,'last')+100;
    hold on
    plot(1:size(H,2),Trow1,'-','Linewidth',2)
    plot(1:size(H,2),Trow2,':','Linewidth',2)
    plot(1:size(H,2),Trow3,':','Linewidth',2)
```

```
    xlim([xlim1 xlim2])
    ylim([-5,max(Trow1)+20])
    title(sprintf('Island transect at row %d',TranRow))
    legend('Initial','13 years','27 years','Location','northwest')
    hold off
end

fprintf('DONE!')
toc

%!!!!!!!!!!!!!!!!!!!!!!!!!!no, seriously ... it's over.!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!%
```

# B.2 Aeolian Transport code

```
function [Hstar]=AeolianTransport03312021(Hstar,delta,L,W,Windspeed,Windmin,StormThreshold,WindDir,PC)

%Aeolian Transport is triggered by sufficient wind speeds chosen in the MainCode.
%AT determines the number of cells that the sediment will move based on the
%current windspeed:
%                          windspeed < 6 move 1 cell
%                          6<= windspeed < 11 move 2 cells
%                          11<= windspeed < 16 move 3 cells

%When each cell is polled the plant density on that cell is accounted for,
%but it is assumed that after initial saltation the sediment is moving
%freely enough so as not to require polling subsequent cells' plant cover.
%The angle between the current cell and the next potential cell is checked
%for every step 1, 2, and 3.

%each cell has a probability of moving either in the same direction as the
%EindDirection input or one of the two off-directions
% (i.e. if WindDir is North: 50% move North, 25% move NorthEast, 25% move NorthWest


MvF=.8; %movement probability factor (turn up for more movement, down for less)

Hp=padarray(Hstar(:,:,1),[3 3]);    %padding Hstar with three estra rows/cols in each direction - avoids index errors
                                        %cells moved into padded region are
                                        %deleted when Hstar is redeclared
                                        %after each loop
[n1 n2]=size(Hstar(:,:,1));         %array indexes for loop

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

% fprintf('I am running AT! ')


%can move up to three cells based on windspeed
MvCnt=1*(Windspeed>=6)+1*(Windspeed>=9)+1*(Windspeed>=12);

%***about MarshFlag*** - we are currently allowing the wind to affect cells in
%the swamp, but previously we were flagging them so that they would not be
%moved. If you do not want the wind to move swamp cells then change the
%rows labeled below

for i=4:n1+3
    Rnow=Hp(i,:);
    for j=n2+3:-1:4
        MarshFlag=0;        %flagging cells in the marsh
        if Rnow(j)>=0
            if (Rnow(j)> 0)&& Rnow(j)<=1
                if j==1
                    if Rnow(j)<Rnow(j+1) && Rnow(j+1)<Rnow(j+2)
                        MarshFlag=0;                         %<---change to MarshFlag=1; if wish to ignore marsh cells
                    end
                elseif j==n2
```

```
                if Rnow(j)>Rnow(j-1) && Rnow(j-1)>Rnow(j-2)
                    MarshFlag=0;                                  %<---change to MarshFlag=1; if wish to ignore marsh cells
                end
            else
                if Rnow(j)<Rnow(j+1) && Rnow(j-1)<Rnow(j)
                    MarshFlag=0;                                  %<---change to MarshFlag=1; if wish to ignore marsh cells
                end
            end
        end

        if Rnow(j)>0 && MarshFlag==0
                %probability for RNG use
                P=0.5;      %P splits the likelihood of moving in primary wind direction or the two alternative directions
                Rand=rand;  %for probability check
                MoveChance=(1-PC(i,j))*((max(Windspeed-Windmin,0))/(StormThreshold-Windmin))*MvF;   %cells chance of moving, gactors in plants and windpseed ranges
                dist=round(MvCnt*MoveChance,0); %number of cells a slab can potentially move

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

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

                Nhd=Hp(i-3:i+3,j-3:j+3); %7x7 moving window for movement up to three cells
                PCNhd=PC(i-3:i+3,j-3:j+3);
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

                %NORTH BLOCK

                %NORTH - NW

                if WindDir==1 && Rand<MoveChance
                    if Rand<((1-P)/2) %&& (i>3) && (j>3) %move to the port slab I don't think I need to worry about these if I use the padded array
                        DirV=[Nhd(4,4), Nhd(3,3), Nhd(2,2), Nhd(1,1)];  %vector of neighborhood Hstar values in the direction of chosen movement

        %next line for re-calculating chance of moving from new cell - if we wish to poll for plant cover at each step at some later time
                        PCNE=[PCNhd(4,4), PCNhd(3,3), PCNhd(2,2), PCNhd(1,1)];

                        flag=1; %trigger the cell movement
                        k=1;    %count for number of cells moved so far in while loop
                        while flag==1
                            if atan(((DirV(k+1)-DirV(k))*delta)/L)<(pi/12) % if the angle between current cell and cell being moved to is sufficiently shallow
                                DirV(k)=DirV(k)-1;                         %remove one slab from current cell
                                DirV(k+1)=DirV(k+1)+1;                     %add that cell to neighbor
                                k=k+1;                                     %add one to cell moved count
                                flag=1*(k<=dist);                          %repeat until k is equal to distance calculated earlier
        %next line for re-calculating chance of moving from new cell - if we wish to poll for plant cover at each step at some
```

```
later time
                    MoveChance=(1-PCNE(k))*((max(Windspeed-Windmin,0))/(StormThreshold-Windmin))*MvF;
                    if Rand >= MoveChance
                        flag=0;
                    end
                else
                    flag=0;                                     %if too steep an angle is encountered, movement stops
                end
            end
            Nhd(4,4)=DirV(1);                                   %replace neighborhood values with the direction vector values
            Nhd(3,3)=DirV(2);
            Nhd(2,2)=DirV(3);
            Nhd(1,1)=DirV(4);

            %NORTH - N
        elseif Rand>=((1-P)/2) && Rand<((1+P)/2) %&& (i>1)    %move with the wind / North
            DirV=[Nhd(4,4), Nhd(3,4), Nhd(2,4), Nhd(1,4)];
            PCNE=[PCNhd(4,4), PCNhd(3,4), PCNhd(2,4), PCNhd(1,4)];
            flag=1;
            k=1;
            while flag==1
                if atan(((DirV(k+1)-DirV(k))*delta)/L)<(pi/12)
                    DirV(k)=DirV(k)-1;
                    DirV(k+1)=DirV(k+1)+1;
%                     cellcnt=cellcnt+1;
%                     MoveChance=(1-PCNE(k+1))*((max(Windspeed-Windmin,0))/(StormThreshold-Windmin));
                    k=k+1;
                    flag=1*(k<=dist);
                    MoveChance=(1-PCNE(k))*((max(Windspeed-Windmin,0))/(StormThreshold-Windmin))*MvF;
                    if Rand >= MoveChance
                        flag=0;
                    end
                else
                    flag=0;
                end
            end
            Nhd(4,4)=DirV(1);
            Nhd(3,4)=DirV(2);
            Nhd(2,4)=DirV(3);
            Nhd(1,4)=DirV(4);


            %NORTH - NE
        elseif Rand>=((1+P)/2) && (i>1) && (j<size(Hstar,2))        %move to the starboard slab
            DirV=[Nhd(4,4), Nhd(3,5), Nhd(2,6), Nhd(1,7)];
            PCNE=[PCNhd(4,4), PCNhd(3,5), PCNhd(2,6), PCNhd(1,7)];
            flag=1;
            k=1;
            while flag==1
                if atan(((DirV(k+1)-DirV(k))*delta)/L)<(pi/12)
                    DirV(k)=DirV(k)-1;
                    DirV(k+1)=DirV(k+1)+1;
%                     cellcnt=cellcnt+1;
%                     MoveChance=(1-PCNE(k+1))*((max(Windspeed-Windmin,0))/(StormThreshold-Windmin));
```

```
                        k=k+1;
                        flag=1*(k<=dist);
                        MoveChance=(1-PCNE(k))*((max(Windspeed-Windmin,0))/(StormThreshold-Windmin))*MvF;
                        if Rand >= MoveChance
                            flag=0;
                        end
                    else
                        flag=0;
                    end
                end
                Nhd(4,4)=DirV(1);
                Nhd(3,5)=DirV(2);
                Nhd(2,6)=DirV(3);
                Nhd(1,7)=DirV(4);
            end

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

        %NORTH - EAST BLOCK

        elseif WindDir==2 && rand<MoveChance

            %MOVE N
            if Rand<((1-P)/2) %&& (i>1)
                DirV=[Nhd(4,4), Nhd(3,4), Nhd(2,4), Nhd(1,4)];
                PCNE=[PCNhd(4,4), PCNhd(3,4), PCNhd(2,4), PCNhd(1,4)];
                flag=1;
                k=1;
                while flag==1
                    if atan(((DirV(k+1)-DirV(k))*delta)/L)<(pi/12)
                        DirV(k)=DirV(k)-1;
                        DirV(k+1)=DirV(k+1)+1;
%                         cellcnt=cellcnt+1;
%                         MoveChance=(1-PCNE(k+1))*((max(Windspeed-Windmin,0))/(StormThreshold-Windmin));
                        k=k+1;
                        flag=1*(k<=dist);
                        MoveChance=(1-PCNE(k))*((max(Windspeed-Windmin,0))/(StormThreshold-Windmin))*MvF;
                        if Rand >= MoveChance
                            flag=0;
                        end
                    else
                        flag=0;
                    end
                end
                Nhd(4,4)=DirV(1);
                Nhd(3,4)=DirV(2);
                Nhd(2,4)=DirV(3);
                Nhd(1,4)=DirV(4);

%               %MOVE NE
            elseif Rand>=((1-P)/2) && Rand<((1+P)/2)% && (i>1) && (j<size(Hstar,2))
                DirV=[Nhd(4,4), Nhd(3,5), Nhd(2,6), Nhd(1,7)];
                PCNE=[PCNhd(4,4), PCNhd(3,5), PCNhd(2,6), PCNhd(1,7)];
                flag=1;
```

```
            k=1;
            while flag==1
                if atan(((DirV(k+1)-DirV(k))*delta)/L)<(pi/12)
                    DirV(k)=DirV(k)-1;
                    DirV(k+1)=DirV(k+1)+1;
%                   cellcnt=cellcnt+1;
%                   MoveChance=(1-PCNE(k+1))*((max(Windspeed-Windmin,0))/(StormThreshold-Windmin));
                    k=k+1;
                    flag=1*(k<=dist);
                    MoveChance=(1-PCNE(k))*((max(Windspeed-Windmin,0))/(StormThreshold-Windmin))*MvF;
                    if Rand >= MoveChance
                        flag=0;
                    end
                else
                    flag=0;
                end
            end
            Nhd(4,4)=DirV(1);
            Nhd(3,5)=DirV(2);
            Nhd(2,6)=DirV(3);
            Nhd(1,7)=DirV(4);

%               %MOVE E
        elseif Rand>=((1+P)/2)% && (j<size(Hstar,2))
            DirV=[Nhd(4,4), Nhd(4,5), Nhd(4,6), Nhd(4,7)];
            PCNE=[PCNhd(4,4), PCNhd(4,5), PCNhd(4,6), PCNhd(4,7)];
            flag=1;
            k=1;
            while flag==1
                if atan(((DirV(k+1)-DirV(k))*delta)/L)<(pi/12)
                    DirV(k)=DirV(k)-1;
                    DirV(k+1)=DirV(k+1)+1;
%                   cellcnt=cellcnt+1;
%                   MoveChance=(1-PCNE(k+1))*((max(Windspeed-Windmin,0))/(StormThreshold-Windmin));
                    k=k+1;
                    flag=1*(k<=dist);
                    MoveChance=(1-PCNE(k))*((max(Windspeed-Windmin,0))/(StormThreshold-Windmin))*MvF;
                    if Rand >= MoveChance
                        flag=0;
                    end
                else
                    flag=0;
                end
            end
            Nhd(4,4)=DirV(1);
            Nhd(4,5)=DirV(2);
            Nhd(4,6)=DirV(3);
            Nhd(4,7)=DirV(4);
        end

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

    %EAST BLOCK

```

```
                    elseif WindDir==3 && rand<MoveChance

                      %MOVES NE
                      if Rand<((1-P)/2)% && (i>1) && (j<size(Hstar,2))
                        DirV=[Nhd(4,4), Nhd(3,5), Nhd(2,6), Nhd(1,7)];
                        PCNE=[PCNhd(4,4), PCNhd(3,5), PCNhd(2,6), PCNhd(1,7)];
                        flag=1;
                        k=1;
                        while flag==1
                          if atan(((DirV(k+1)-DirV(k))*delta)/L)<(pi/12)
                            DirV(k)=DirV(k)-1;
                            DirV(k+1)=DirV(k+1)+1;
%                             cellcnt=cellcnt+1;
%                             MoveChance=(1-PCNE(k+1))*((max(Windspeed-Windmin,0))/(StormThreshold-Windmin));
                            k=k+1;
                            flag=1*(k<=dist);
                            MoveChance=(1-PCNE(k))*((max(Windspeed-Windmin,0))/(StormThreshold-Windmin))*MvF;
                            if Rand >= MoveChance
                              flag=0;
                            end
                          else
                            flag=0;
                          end
                        end
                        Nhd(4,4)=DirV(1);
                        Nhd(3,5)=DirV(2);
                        Nhd(2,6)=DirV(3);
                        Nhd(1,7)=DirV(4);

                        %MOVES E
                      elseif Rand>=((1-P)/2) && Rand<((1+P)/2) %&& (j<size(Hstar,2))
                        DirV=[Nhd(4,4), Nhd(4,5), Nhd(4,6), Nhd(4,7)];
                        PCNE=[PCNhd(4,4), PCNhd(4,5), PCNhd(4,6), PCNhd(4,7)];
                        flag=1;
                        k=1;
                        while flag==1
                          if atan(((DirV(k+1)-DirV(k))*delta)/L)<(pi/12)
                            DirV(k)=DirV(k)-1;
                            DirV(k+1)=DirV(k+1)+1;
%                             cellcnt=cellcnt+1;
%                             MoveChance=(1-PCNE(k+1))*((max(Windspeed-Windmin,0))/(StormThreshold-Windmin));
                            k=k+1;
                            flag=1*(k<=dist);
                            MoveChance=(1-PCNE(k))*((max(Windspeed-Windmin,0))/(StormThreshold-Windmin))*MvF;
                            if Rand >= MoveChance
                              flag=0;
                            end
                          else
                            flag=0;
                          end
                        end
                        Nhd(4,4)=DirV(1);
                        Nhd(4,5)=DirV(2);
                        Nhd(4,6)=DirV(3);
                        Nhd(4,7)=DirV(4);

```

```
                %MOVES SE
            elseif Rand>=((1+P)/2)% && (i<size(Hstar,1)) && (j<size(Hstar,2))
                DirV=[Nhd(4,4), Nhd(5,5), Nhd(6,6), Nhd(7,7)];
                PCNE=[PCNhd(4,4), PCNhd(5,5), PCNhd(6,6), PCNhd(7,7)];
                flag=1;
                k=1;
                while flag==1
                    if atan(((DirV(k+1)-DirV(k))*delta)/L)<(pi/12)
                        DirV(k)=DirV(k)-1;
                        DirV(k+1)=DirV(k+1)+1;
%                         cellcnt=cellcnt+1;
%                         MoveChance=(1-PCNE(k+1))*((max(Windspeed-Windmin,0))/(StormThreshold-Windmin));
                        k=k+1;
                        flag=1*(k<=dist);
                        MoveChance=(1-PCNE(k))*((max(Windspeed-Windmin,0))/(StormThreshold-Windmin))*MvF;
                        if Rand >= MoveChance
                            flag=0;
                        end
                    else
                        flag=0;
                    end
                end
                Nhd(4,4)=DirV(1);
                Nhd(5,5)=DirV(2);
                Nhd(6,6)=DirV(3);
                Nhd(7,7)=DirV(4);


            end

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

            %SOUTH-EAST BLOCK

        elseif WindDir==4 && rand<MoveChance

            %MOVES E
            if Rand<((1-P)/2)% && (j<size(Hstar,2))
                DirV=[Nhd(4,4), Nhd(4,5), Nhd(4,6), Nhd(4,7)];
                PCNE=[PCNhd(4,4), PCNhd(4,5), PCNhd(4,6), PCNhd(4,7)];
                flag=1;
                k=1;
                while flag==1
                    if atan(((DirV(k+1)-DirV(k))*delta)/L)<(pi/12)
                        DirV(k)=DirV(k)-1;
                        DirV(k+1)=DirV(k+1)+1;
%                         cellcnt=cellcnt+1;
%                         MoveChance=(1-PCNE(k+1))*((max(Windspeed-Windmin,0))/(StormThreshold-Windmin));
                        k=k+1;
                        flag=1*(k<=dist);
                        MoveChance=(1-PCNE(k))*((max(Windspeed-Windmin,0))/(StormThreshold-Windmin))*MvF;
                        if Rand >= MoveChance
                            flag=0;
```

```
                    end
                else
                    flag=0;
                end
            end
            Nhd(4,4)=DirV(1);
            Nhd(4,5)=DirV(2);
            Nhd(4,6)=DirV(3);
            Nhd(4,7)=DirV(4);


            %MOVES SE
        elseif Rand>=((1-P)/2) && Rand<((1+P)/2)% && (i<size(Hstar,1)) && (j<size(Hstar,2))
            DirV=[Nhd(4,4), Nhd(5,5), Nhd(6,6), Nhd(7,7)];
            PCNE=[PCNhd(4,4), PCNhd(5,5), PCNhd(6,6), PCNhd(7,7)];
            flag=1;
            k=1;
            while flag==1
                if atan(((DirV(k+1)-DirV(k))*delta)/L)<(pi/12)
                    DirV(k)=DirV(k)-1;
                    DirV(k+1)=DirV(k+1)+1;
%                    cellcnt=cellcnt+1;
%                    MoveChance=(1-PCNE(k+1))*((max(Windspeed-Windmin,0))/(StormThreshold-Windmin));
                    k=k+1;
                    flag=1*(k<=dist);
                    MoveChance=(1-PCNE(k))*((max(Windspeed-Windmin,0))/(StormThreshold-Windmin))*MvF;
                    if Rand >= MoveChance
                        flag=0;
                    end
                else
                    flag=0;
                end
            end
            Nhd(4,4)=DirV(1);
            Nhd(5,5)=DirV(2);
            Nhd(6,6)=DirV(3);
            Nhd(7,7)=DirV(4);

            %MOVES S
        elseif Rand>=((1+P)/2) && (i<size(Hstar,1))
            DirV=[Nhd(4,4), Nhd(5,4), Nhd(6,4), Nhd(7,4)];
            PCNE=[PCNhd(4,4), PCNhd(5,4), PCNhd(6,4), PCNhd(7,4)];
            flag=1;
            k=1;
            while flag==1
                if atan(((DirV(k+1)-DirV(k))*delta)/L)<(pi/12)
                    DirV(k)=DirV(k)-1;
                    DirV(k+1)=DirV(k+1)+1;
%                    cellcnt=cellcnt+1;
%                    MoveChance=(1-PCNE(k+1))*((max(Windspeed-Windmin,0))/(StormThreshold-Windmin));
                    k=k+1;
                    flag=1*(k<=dist);
                    MoveChance=(1-PCNE(k))*((max(Windspeed-Windmin,0))/(StormThreshold-Windmin))*MvF;
                    if Rand >= MoveChance
                        flag=0;
                    end
```

```
                        else
                            flag=0;
                        end
                    end
                    Nhd(4,4)=DirV(1);
                    Nhd(5,4)=DirV(2);
                    Nhd(6,4)=DirV(3);
                    Nhd(7,4)=DirV(4);

                end

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

                %SOUTH BLOCK

            elseif WindDir==5 && rand<MoveChance

                %MOVES SE
                if Rand<((1-P)/2) %&& (i<size(Hstar,1)) && (j<size(Hstar,2))
                    DirV=[Nhd(4,4), Nhd(5,5), Nhd(6,6), Nhd(7,7)];
                    PCNE=[PCNhd(4,4), PCNhd(5,5), PCNhd(6,6), PCNhd(7,7)];
                    flag=1;
                    k=1;
                    while flag==1
                        if atan(((DirV(k+1)-DirV(k))*delta)/L)<(pi/12)
                            DirV(k)=DirV(k)-1;
                            DirV(k+1)=DirV(k+1)+1;
%                               cellcnt=cellcnt+1;
%                               MoveChance=(1-PCNE(k+1))*((max(Windspeed-Windmin,0))/(StormThreshold-Windmin));
                            k=k+1;
                            flag=1*(k<=dist);
                            MoveChance=(1-PCNE(k))*((max(Windspeed-Windmin,0))/(StormThreshold-Windmin))*MvF;
                            if Rand >= MoveChance
                                flag=0;
                            end
                        else
                            flag=0;
                        end
                    end
                    Nhd(4,4)=DirV(1);
                    Nhd(5,5)=DirV(2);
                    Nhd(6,6)=DirV(3);
                    Nhd(7,7)=DirV(4);

                    %MOVES S
                elseif Rand>=((1-P)/2) && Rand<((1+P)/2) %&& (i<size(Hstar,1))
                    DirV=[Nhd(4,4), Nhd(5,4), Nhd(6,4), Nhd(7,4)];
                    PCNE=[PCNhd(4,4), PCNhd(5,4), PCNhd(6,4), PCNhd(7,4)];
                    flag=1;
                    k=1;
                    while flag==1
                        if atan(((DirV(k+1)-DirV(k))*delta)/L)<(pi/12)
                            DirV(k)=DirV(k)-1;
                            DirV(k+1)=DirV(k+1)+1;
```

```
%                  cellcnt=cellcnt+1;
%                  MoveChance=(1-PCNE(k+1))*((max(Windspeed-Windmin,0))/(StormThreshold-Windmin));
                 k=k+1;
                 flag=1*(k<=dist);
                 MoveChance=(1-PCNE(k))*((max(Windspeed-Windmin,0))/(StormThreshold-Windmin))*MvF;
                 if Rand >= MoveChance
                     flag=0;
                 end
             else
                 flag=0;
             end
         end
         Nhd(4,4)=DirV(1);
         Nhd(5,4)=DirV(2);
         Nhd(6,4)=DirV(3);
         Nhd(7,4)=DirV(4);

         %MOVES SW
     elseif Rand>=((1+P)/2) %&& (i<size(Hstar,1)) %&& (j>1)
         DirV=[Nhd(4,4), Nhd(5,3), Nhd(6,2), Nhd(7,1)];
         PCNE=[PCNhd(4,4), PCNhd(5,3), PCNhd(6,2), PCNhd(7,1)];
         flag=1;
         k=1;
         while flag==1
             if atan(((DirV(k+1)-DirV(k))*delta)/L)<(pi/12)
                 DirV(k)=DirV(k)-1;
                 DirV(k+1)=DirV(k+1)+1;
%                  cellcnt=cellcnt+1;
%                  MoveChance=(1-PCNE(k+1))*((max(Windspeed-Windmin,0))/(StormThreshold-Windmin));
                 k=k+1;
                 flag=1*(k<=dist);
                 MoveChance=(1-PCNE(k))*((max(Windspeed-Windmin,0))/(StormThreshold-Windmin))*MvF;
                 if Rand >= MoveChance
                     flag=0;
                 end
             else
                 flag=0;
             end
         end
         Nhd(4,4)=DirV(1);
         Nhd(5,3)=DirV(2);
         Nhd(6,2)=DirV(3);
         Nhd(7,1)=DirV(4);


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

     %SOUTH-WEST BLOCK

   elseif WindDir==6 && rand<MoveChance

     %MOVES S
     if Rand<((1-P)/2) %&& (i<size(Hstar,1))
         DirV=[Nhd(4,4), Nhd(5,4), Nhd(6,4), Nhd(7,4)];
```

```
                PCNE=[PCNhd(4,4), PCNhd(5,4), PCNhd(6,4), PCNhd(7,4)];
                flag=1;
                k=1;
                while flag==1
                    if atan(((DirV(k+1)-DirV(k))*delta)/L)<(pi/12)
                        DirV(k)=DirV(k)-1;
                        DirV(k+1)=DirV(k+1)+1;
%                           cellcnt=cellcnt+1;
%                           MoveChance=(1-PCNE(k+1))*((max(Windspeed-Windmin,0))/(StormThreshold-Windmin));
                        k=k+1;
                        flag=1*(k<=dist);
                        MoveChance=(1-PCNE(k))*((max(Windspeed-Windmin,0))/(StormThreshold-Windmin))*MvF;
                        if Rand >= MoveChance
                            flag=0;
                        end
                    else
                        flag=0;
                    end
                end
                Nhd(4,4)=DirV(1);
                Nhd(5,4)=DirV(2);
                Nhd(6,4)=DirV(3);
                Nhd(7,4)=DirV(4);


                %MOVES SW
            elseif Rand>=((1-P)/2) && Rand<((1+P)/2) %&& (i<size(Hstar,1)) && (j>1)
                DirV=[Nhd(4,4), Nhd(5,3), Nhd(6,2), Nhd(7,1)];
                PCNE=[PCNhd(4,4), PCNhd(5,3), PCNhd(6,2), PCNhd(7,1)];
                flag=1;
                k=1;
                while flag==1
                    if atan(((DirV(k+1)-DirV(k))*delta)/L)<(pi/12)
                        DirV(k)=DirV(k)-1;
                        DirV(k+1)=DirV(k+1)+1;
%                           cellcnt=cellcnt+1;
%                           MoveChance=(1-PCNE(k+1))*((max(Windspeed-Windmin,0))/(StormThreshold-Windmin));
                        k=k+1;
                        flag=1*(k<=dist);
                        MoveChance=(1-PCNE(k))*((max(Windspeed-Windmin,0))/(StormThreshold-Windmin))*MvF;
                        if Rand >= MoveChance
                            flag=0;
                        end
                    else
                        flag=0;
                    end
                end
                Nhd(4,4)=DirV(1);
                Nhd(5,3)=DirV(2);
                Nhd(6,2)=DirV(3);
                Nhd(7,1)=DirV(4);


                %MOVES W
            elseif Rand>=((1+P)/2) %&& (j>1)
                DirV=[Nhd(4,4), Nhd(4,3), Nhd(4,2), Nhd(4,1)];
```

```
                        PCNE=[PCNhd(4,4), PCNhd(4,3), PCNhd(4,2), PCNhd(4,1)];
                        flag=1;
                        k=1;
                        while flag==1
                            if atan(((DirV(k+1)-DirV(k))*delta)/L)<(pi/12)
                                DirV(k)=DirV(k)-1;
                                DirV(k+1)=DirV(k+1)+1;
%                                 cellcnt=cellcnt+1;
%                                 MoveChance=(1-PCNE(k+1))*((max(Windspeed-Windmin,0))/(StormThreshold-Windmin));
                                k=k+1;
                                flag=1*(k<=dist);
                                MoveChance=(1-PCNE(k))*((max(Windspeed-Windmin,0))/(StormThreshold-Windmin))*MvF;
                                if Rand >= MoveChance
                                    flag=0;
                                end
                            else
                                flag=0;
                            end
                        end
                        Nhd(4,4)=DirV(1);
                        Nhd(4,3)=DirV(2);
                        Nhd(4,2)=DirV(3);
                        Nhd(4,1)=DirV(4);

                    end

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

                    %WEST BLOCK

                elseif WindDir==7 && rand<MoveChance

                    %MOVES SW
                    if Rand<((1-P)/2) && (i<size(Hstar,1)) %&& (j>1)
                        DirV=[Nhd(4,4), Nhd(5,3), Nhd(6,2), Nhd(7,1)];
                        PCNE=[PCNhd(4,4), PCNhd(5,3), PCNhd(6,2), PCNhd(7,1)];
                        flag=1;
                        k=1;
                        while flag==1
                            if atan(((DirV(k+1)-DirV(k))*delta)/L)<(pi/12)
                                DirV(k)=DirV(k)-1;
                                DirV(k+1)=DirV(k+1)+1;
%                                 cellcnt=cellcnt+1;
%                                 MoveChance=(1-PCNE(k+1))*((max(Windspeed-Windmin,0))/(StormThreshold-Windmin));
                                k=k+1;
                                flag=1*(k<=dist);
                                MoveChance=(1-PCNE(k))*((max(Windspeed-Windmin,0))/(StormThreshold-Windmin))*MvF;
                                if Rand >= MoveChance
                                    flag=0;
                                end
                            else
                                flag=0;
                            end
                        end
```

```
                Nhd(4,4)=DirV(1);
                Nhd(5,3)=DirV(2);
                Nhd(6,2)=DirV(3);
                Nhd(7,1)=DirV(4);

                %MOVES W
            elseif Rand>=((1-P)/2) && Rand<((1+P)/2) %&& (j>1)
                DirV=[Nhd(4,4), Nhd(4,3), Nhd(4,2), Nhd(4,1)];
                PCNE=[PCNhd(4,4), PCNhd(4,3), PCNhd(4,2), PCNhd(4,1)];
                flag=1;
                k=1;
                while flag==1
                    if atan(((DirV(k+1)-DirV(k))*delta)/L)<(pi/12)
                        DirV(k)=DirV(k)-1;
                        DirV(k+1)=DirV(k+1)+1;
%                         cellcnt=cellcnt+1;
%                         MoveChance=(1-PCNE(k+1))*((max(Windspeed-Windmin,0))/(StormThreshold-Windmin));
                        k=k+1;
                        flag=1*(k<=dist);
                        MoveChance=(1-PCNE(k))*((max(Windspeed-Windmin,0))/(StormThreshold-Windmin))*MvF;
                        if Rand >= MoveChance
                            flag=0;
                        end
                    else
                        flag=0;
                    end
                end
                Nhd(4,4)=DirV(1);
                Nhd(4,3)=DirV(2);
                Nhd(4,2)=DirV(3);
                Nhd(4,1)=DirV(4);

                %MOVES NW
            elseif Rand>=((1+P)/2) %&& (i>1) && (j>1)
                DirV=[Nhd(4,4), Nhd(3,3), Nhd(2,2), Nhd(2,1)];
                PCNE=[PCNhd(4,4), PCNhd(3,3), PCNhd(2,2), PCNhd(1,1)];
                flag=1;
                k=1;
                while flag==1
                    if atan(((DirV(k+1)-DirV(k))*delta)/L)<(pi/12)
                        DirV(k)=DirV(k)-1;
                        DirV(k+1)=DirV(k+1)+1;
%                         cellcnt=cellcnt+1;
%                         MoveChance=(1-PCNE(k+1))*((max(Windspeed-Windmin,0))/(StormThreshold-Windmin));
                        k=k+1;
                        flag=1*(k<=dist);
                        MoveChance=(1-PCNE(k))*((max(Windspeed-Windmin,0))/(StormThreshold-Windmin))*MvF;
                        if Rand >= MoveChance
                            flag=0;
                        end
                    else
                        flag=0;
                    end
                end
                Nhd(4,4)=DirV(1);
                Nhd(3,3)=DirV(2);
```

```
                Nhd(2,2)=DirV(3);
                Nhd(1,1)=DirV(4);

            end

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

            %NORTH-WEST

        elseif WindDir==8 && rand<MoveChance

            %MOVES W
            if Rand<((1-P)/2) %&& (j>1)
                DirV=[Nhd(4,4), Nhd(4,3), Nhd(4,2), Nhd(4,1)];
                PCNE=[PCNhd(4,4), PCNhd(4,3), PCNhd(4,2), PCNhd(4,1)];
                flag=1;
                k=1;
                while flag==1
                    if atan(((DirV(k+1)-DirV(k))*delta)/L)<(pi/12)
                        DirV(k)=DirV(k)-1;
                        DirV(k+1)=DirV(k+1)+1;
%                         cellcnt=cellcnt+1;
%                         MoveChance=(1-PCNE(k+1))*((max(Windspeed-Windmin,0))/(StormThreshold-Windmin));
                        k=k+1;
                        flag=1*(k<=dist);
                        MoveChance=(1-PCNE(k))*((max(Windspeed-Windmin,0))/(StormThreshold-Windmin))*MvF;
                        if Rand >= MoveChance
                            flag=0;
                        end
                    else
                        flag=0;
                    end
                end
                Nhd(4,4)=DirV(1);
                Nhd(4,3)=DirV(2);
                Nhd(4,2)=DirV(3);
                Nhd(4,1)=DirV(4);

                %MOVES NW
            elseif Rand>=((1-P)/2) && Rand<((1+P)/2) %&& (i>1) && (j>1)
                DirV=[Nhd(4,4), Nhd(3,3), Nhd(2,2), Nhd(2,1)];
                PCNE=[PCNhd(4,4), PCNhd(3,3), PCNhd(2,2), PCNhd(1,1)];
                flag=1;
                k=1;
                while flag==1
                    if atan(((DirV(k+1)-DirV(k))*delta)/L)<(pi/12)
                        DirV(k)=DirV(k)-1;
                        DirV(k+1)=DirV(k+1)+1;
%                         cellcnt=cellcnt+1;
%                         MoveChance=(1-PCNE(k+1))*((max(Windspeed-Windmin,0))/(StormThreshold-Windmin));
                        k=k+1;
                        flag=1*(k<=dist);
                        MoveChance=(1-PCNE(k))*((max(Windspeed-Windmin,0))/(StormThreshold-Windmin))*MvF;
                        if Rand >= MoveChance
```

```
                                flag=0;
                            end
                        else
                            flag=0;
                        end
                    end
                    Nhd(4,4)=DirV(1);
                    Nhd(3,3)=DirV(2);
                    Nhd(2,2)=DirV(3);
                    Nhd(1,1)=DirV(4);

                    %MOVES N
                elseif Rand>=((1+P)/2) %&& (i>1)
                    DirV=[Nhd(4,4), Nhd(3,4), Nhd(2,4), Nhd(1,4)];
                    PCNE=[PCNhd(4,4), PCNhd(3,4), PCNhd(2,4), PCNhd(1,4)];
                    flag=1;
                    k=1;
                    while flag==1
                        if atan(((DirV(k+1)-DirV(k))*delta)/L)<(pi/12)
                            DirV(k)=DirV(k)-1;
                            DirV(k+1)=DirV(k+1)+1;
%                            cellcnt=cellcnt+1;
%                            MoveChance=(1-PCNE(k+1))*((max(Windspeed-Windmin,0))/(StormThreshold-Windmin));
                            k=k+1;
                            flag=1*(k<=dist);
                            MoveChance=(1-PCNE(k))*((max(Windspeed-Windmin,0))/(StormThreshold-Windmin))*MvF;
                            if Rand >= MoveChance
                                flag=0;
                            end
                        else
                            flag=0;
                        end
                    end
                    Nhd(4,4)=DirV(1);
                    Nhd(3,4)=DirV(2);
                    Nhd(2,4)=DirV(3);
                    Nhd(1,4)=DirV(4);
%
%                     flag=1;
%                     cnt=0;
%                     k=1;
%                     while flag==1
%                         if atan(((Nhd(4-k,4)-Nhd(4-k+1,4))*delta)/L)<(pi/12)
%                             Nhd(4-k+1,4)=Nhd(4-k+1,4)-1;%4-k+1=4-(k-1)
%                             Nhd(4-k,4)=Nhd(4-k,4)+1;
%                             MoveChance=(1-PCNhd(4-k,4))*((max(Windspeed-Windmin,0))/(StormThreshold-Windmin));%(
    calculating, but not using yet)
%                             k=k+1;
%                             cnt=cnt+1;
%                             flag=1*(k<=dist);%*(Nhd(4,4)>=0)
%                         else
%                             flag=0;
%                         end
%                     end
                end
                end
```

```
                    end
                Hp(i-3:i+3,j-3:j+3,1)=Nhd;
                end
            else
            end
        end
    end

    Hstar(:,:,1)=Hp(4:n1+3,4:n2+3);

end %end of AT function

%COMMENT: Old way of polling may have made it easier to change number of
%steps in each direction that a cell can possible move:

%it used this neighborhood function to declare a local region of 3 cells in
%each direction. If we used a variablelike D=#cells wished to move

%then...

%
% function N1=NewNhd(R,C,H)
% if R<=(size(H,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(H,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
% N1=zeros(3,3);                    %<-------------CHANGED THIS TO zeros(D,D)******

% N1=[H(Rm,Cm),H(Rm,C),H(Rm,Cp);H(R,Cm),H(R,C),H(R,Cp);H(Rp,Cm),H(Rp,C),H(Rp,Cp)];
%
% end

% *******************and used the old method of checking:

% *******************All of these 4's would need to be replaced with D+1

```

```
%                   flag=1;
%                   cnt=0;
%                   k=1;
%                   while flag==1
%                       if atan(((Nhd(4-k,4)-Nhd(4-k+1,4))*delta)/L)<(pi/12)
%                           Nhd(4-k+1,4)=Nhd(4-k+1,4)-1;%4-k+1=4-(k-1)
%                           Nhd(4-k,4)=Nhd(4-k,4)+1;
%                           MoveChance=(1-PCNhd(4-k,4))*((max(Windspeed-Windmin,0))/(StormThreshold-Windmin));%(calculating, but not using yet)
%                           k=k+1;
%                           cnt=cnt+1;
%                           flag=1*(k<=dist);%*(Nhd(4,4)>=0)
%                       else
%                           flag=0;
%                       end
%                   end
```

## B.3 Avalanche code

```
function [Hstar,flag,CellCt] = AVALANCHEtime03312021(Hstar,delta,L,flag,PC,t)
%% This version of avalanche:
%   ~weights the probability of avalanching to favor the direction of the
%        greatest violation of the angle of repose.
%   ~takes t as input to track if avalanching is required over whole domain
%        (once per 5 years)
%   ~most recent edit code was ATTEMPT2newAVALANCHEtime0928
%
 AoR=pi/6;
% AoR=pi/9;

%%
% tic

Hstarflag=Hstar;
CellCt=0;
flag=0;
FLAG=1;
alpha=1;
% fprintf('I am running AV!  ')
Nhd=4;                          %size of neighborhood check (4 or 8)
beta1=zeros(1,Nhd);    %angle of repose
beta2=zeros(1,Nhd);    %
check1=zeros(1,Nhd);
check2=zeros(1,Nhd);

%once per 5 year we check every cell in domain, both above and  below sea level

if (0==mod(t,26)) || exist('t','var')==0
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

        Hx=Hstar(R,C);                                          %current cell value
        N=[Hstar(Rm,C),Hstar(R,Cp),Hstar(Rp,C),Hstar(R,Cm)];     %neighborhood

        beta1=atan(((Hx-N)*delta)/L);
        check1=(beta1>=(AoR));
        %Avdir=find(beta1>=AoR);
        beta2=alpha*((beta1/(AoR))*(1-PC(R,C)));
        check2=(RAND<beta2);


        if sum(check2)~=0
            while sum(check1)~=0 && sum(check2)~=0  %if I use a while loop, there's no point to using probability
                %posbeta1=beta1;
                %posbeta1(find(beta1<0))=0;
                %Nprob=posbeta1/sum(posbeta1);
                AVbeta1=zeros(1,4);
                AVbeta1(check1)=beta1(check1);
                AVbeta1(check2)=AVbeta1(check2);
                Nprob=AVbeta1/sum(AVbeta1);
                prob=rand;
                Hcheck=Hx;
                Hx=Hx-1*check1(1)*check2(1)*(prob<Nprob(1))-1*check1(2)*check2(2)*(prob<Nprob(2))-1*check1(3)*check2(3)*(
                    prob<Nprob(3))-1*check1(4)*check2(4)*(prob<Nprob(4));
%                 if Hx-Hcheck==0
%                     FLAG=0;
%                     break
%                 end
                N(1)=N(1)+check1(1)*check2(1)*(prob<Nprob(1));
                N(2)=N(2)+check1(2)*check2(2)*(prob<Nprob(2));
                N(3)=N(3)+check1(3)*check2(3)*(prob<Nprob(3));
                N(4)=N(4)+check1(4)*check2(4)*(prob<Nprob(4));
                beta1=newbeta1(N,Hx);
                beta2=newbeta2(N,beta1);
                check1=newcheck1(N,beta1);
                check2=newcheck2(N,beta2);
                %flag=1;
                if Hx-Hcheck~=0
                    CellCt=CellCt+1;
                    flag=1;
                end
                if sum(check1)==0
                    Hstar(R,C)=Hx;
                    Hstar(Rm,C)=N(1);
                    Hstar(R,Cp)=N(2);
                    Hstar(Rp,C)=N(3);
                    Hstar(R,Cm)=N(4);
                    %fprintf('Changes have been made')
                    FLAG=0;
```

```
                            if Hx >=0
                                break
                            end
                        end
                        if check1*check2'==0
                            break
                        end
                    end
                    if sum(check1)==0
                        FLAG=0;
                        %no avalanching needed
                    end
                elseif sum(check2)==0 && sum(check1)~=0
                    FLAG=0;
                    %fprintf('Too many plants!')
                elseif sum(check1)==0
                    FLAG=0;     %only other case should be if both sum to 0
                end

%           else%{if Hstar(R,C)<=0}
%               Erosioncheck=0;
%               FLAG=0;
%           end
            FLAG=0;
        end
    end
end
%for other instances of avalanche we only check subaerial portions of island
else
%   ISLANDindex=Hstar(:,:,1)>0; %all cells with land
%   Hstarflag=cumsum(ISLANDindex,2) == 1 & ISLANDindex; %this outputs an array same size as H but with a 1 in the first cell>0
%   IslandCheck=sum(Hstarflag');    %outputs a row vector with 0~ no col>0, 1~col>0 (ie yes/no land in that row)
%   COLindex1 = Hstarflag*(1:size(Hstar,2))';   %the index of the first positive value in a row
%   COLindex2 = zeros(size(COLindex1));
%   for ii=1:length(COLindex1)
%       if ii == 101
%           fprintf('yay')
%       end
%
%       if COLindex1(ii)~=0
%           RowNow=ISLANDindex(ii,:);
%           Icount=0;
%           for jj = COLindex1(ii):1:size(RowNow)
%               if RowNow(jj)==1 && RowNow(jj+1)==0
%                   Icount=Icount+1;
%                   COLindex2(ii,Icount)=jj;
%               end
%           end
%       end
%   end
IslandColumnArray1=zeros([size(Hstar,1) 1]);

for i=1:size(Hstar,1)
    ElevCheck1=Hstar(i,:,1)>=0;  %first cell above water
    Icount=0;%%
    for j=2:size(Hstar,2)
```

```
        if ElevCheck1(j)==1 && ElevCheck1(j-1)==0      %if delta*Hstar(i,j)>-.5 && Hstar(i,j-1)<=-.5 %&& ColumnArraySwamp1(i,Icount)
            ==0
            Icount=Icount+1;%
            IslandColumnArray1(i,Icount)=j;

                %
            %break;

                %
        end

            %
    end

        %
end
INum=size(IslandColumnArray1,2);
IslandColumnArray2=zeros(size(IslandColumnArray1(:,:)));
for i=1:size(Hstar,1)
    ElevCheck2=Hstar(i,:,1)<=0;
    if sum(IslandColumnArray1(i,1:INum)~=0)>0
        for Icount=1:INum
            if IslandColumnArray1(i,Icount)~=0
                for j=IslandColumnArray1(i,Icount):size(Hstar,2)
                    if ElevCheck2(j)==1 && IslandColumnArray2(i,Icount)==0
                        IslandColumnArray2(i,Icount)=j;
                        break;
    %                   elseif sum(ElevCheck2)==0
    %                       ColumnArraySwamp1(i,Icount)=0;
                    end
                end
            end
        end
    end
end

for R=1:size(Hstar,1)
    for Icount=1:INum
        if IslandColumnArray2(R,Icount)~=0
            for C=IslandColumnArray1(R,Icount):IslandColumnArray2(R,Icount)
                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

                Hx=Hstar(R,C);                                          %current cell value
                N=[Hstar(Rm,C),Hstar(R,Cp),Hstar(Rp,C),Hstar(R,Cm)];    %neighborhood

                beta1=atan(((Hx-N)*delta)/L);
                check1=(beta1>=(AoR));
                %Avdir=find(beta1>=AoR);
                beta2=alpha*((beta1/(AoR))*(1-PC(R,C)));
                check2=(RAND<beta2);


        %          if sum(check1)~=0
        %              if sum(check2)~=0
        %              end
        %          end
                    if R==241
                        if C==492
%                               fprintf('STOP!')
                        end
                    end

                if sum(check2)~=0
                    if check1*check2'~=0
                        while  sum(check1)~=0 && sum(check2)~=0  %if I use a while loop, there's no point to using probability
                            %posbeta1=beta1;
                            %posbeta1(find(beta1<0))=0;
                            %Nprob=posbeta1/sum(posbeta1);
                            AVbeta1=zeros(1,4);
                            AVbeta1(check1)=beta1(check1);
                            AVbeta1(check2)=AVbeta1(check2);
                            Nprob=AVbeta1/sum(AVbeta1);
                            prob=rand;
                            Hcheck=Hx;
                            Hx=Hx-1*check1(1)*check2(1)*(prob<Nprob(1))-1*check1(2)*check2(2)*(prob<Nprob(2))-1*check1(3)*check2(3)*(prob<Nprob(3))-1*check1(4)*check2(4)*(prob<Nprob(4));
          %                   if Hx-Hcheck==0
          %                       FLAG=0;
          %                       break
          %                   end
                            N(1)=N(1)+check1(1)*check2(1)*(prob<Nprob(1));
```

```
                                                N(2)=N(2)+check1(2)*check2(2)*(prob<Nprob(2));
                                                N(3)=N(3)+check1(3)*check2(3)*(prob<Nprob(3));
                                                N(4)=N(4)+check1(4)*check2(4)*(prob<Nprob(4));
                                                beta1=newbeta1(N,Hx);
                                                beta2=newbeta2(N,beta1);
                                                check1=newcheck1(N,beta1);
                                                check2=newcheck2(N,beta2);
                                                %flag=1;
                                                if Hx-Hcheck~=0
                                                    CellCt=CellCt+1;
                                                    flag=1;
                                                end
                                                if sum(check1)==0
                                                    Hstar(R,C)=Hx;
                                                    Hstar(Rm,C)=N(1);
                                                    Hstar(R,Cp)=N(2);
                                                    Hstar(Rp,C)=N(3);
                                                    Hstar(R,Cm)=N(4);
                                                    %fprintf('Changes have been made')
                                                    FLAG=0;
                                                    if Hx >=0
                                                        break
                                                    end
                                                end
                                                if check1*check2'==0
                                                    break
                                                end
                                            end
                                        end
                                        if sum(check1)==0
                                            FLAG=0;
                                            %no avalanching needed
                                        end
                                    elseif sum(check2)==0 && sum(check1)~=0
                                        FLAG=0;
                                        %fprintf('Too many plants!')
                                    elseif sum(check1)==0
                                        FLAG=0;      %only other case should be if both sum to 0
                                    end

                                else%{if Hstar(R,C)<=0}
                                    Erosioncheck=0;
                                    FLAG=0;
                                end
                                FLAG=0;
                            end
                        end
                    end
            end

        end
    end
% end

%this portion is for elevation maps with multiple full sized islands within the domain
%we rarely need this, but it's here just in case
```

```
% for R=1:size(Hstar,1)
%     %ISLANDindex=(Hstar(R,:,1)>0);
%     if sum(ISLANDindex)~=0      %skip whole rows of water
%         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
%
%                     Hx=Hstar(R,C);                                          %current cell value
%                     N=[Hstar(Rm,C),Hstar(R,Cp),Hstar(Rp,C),Hstar(R,Cm)];    %neighborhood
%
%                     beta1=atan(((Hx-N)*delta)/L);
%                     check1=(beta1>=(AoR));
%                     %Avdir=find(beta1>=AoR);
%                     beta2=alpha*((beta1/(AoR))*(1-PC(R,C)));
%                     check2=(RAND<beta2);
%
%
%     %                 if sum(check1)~=0
%     %                     if sum(check2)~=0
%     %                     end
%     %                 end
%                         if R==24
%     %                         fprintf('STOP!')
%                         end
%
%                     if sum(check2)~=0
%                         if check1*check2'~=0
```

```
%                     while  sum(check1)~=0 && sum(check2)~=0  %if I use a while loop, there's no point to using probability
%                         %posbeta1=beta1;
%                         %posbeta1(find(beta1<0))=0;
%                         %Nprob=posbeta1/sum(posbeta1);
%                         AVbeta1=zeros(1,4);
%                         AVbeta1(check1)=beta1(check1);
%                         AVbeta1(check2)=AVbeta1(check2);
%                         Nprob=AVbeta1/sum(AVbeta1);
%                         prob=rand;
%                         Hcheck=Hx;
%                         Hx=Hx-1*check1(1)*check2(1)*(prob<Nprob(1))-1*check1(2)*check2(2)*(prob<Nprob(2))-1*check1(3)*check2(3)*(prob<Nprob(3))-1*check1(4)*check2(4)*(prob<Nprob(4));
%       %                   if Hx-Hcheck==0
%       %                       FLAG=0;
%       %                       break
%       %                   end
%                         N(1)=N(1)+check1(1)*check2(1)*(prob<Nprob(1));
%                         N(2)=N(2)+check1(2)*check2(2)*(prob<Nprob(2));
%                         N(3)=N(3)+check1(3)*check2(3)*(prob<Nprob(3));
%                         N(4)=N(4)+check1(4)*check2(4)*(prob<Nprob(4));
%                         beta1=newbeta1(N,Hx);
%                         beta2=newbeta2(N,beta1);
%                         check1=newcheck1(N,beta1);
%                         check2=newcheck2(N,beta2);
%                         %flag=1;
%                         if Hx-Hcheck~=0
%                             flag=1;
%                         end
%                         if sum(check1)==0
%                             Hstar(R,C)=Hx;
%                             Hstar(Rm,C)=N(1);
%                             Hstar(R,Cp)=N(2);
%                             Hstar(Rp,C)=N(3);
%                             Hstar(R,Cm)=N(4);
%                             %fprintf('Changes have been made')
%                             FLAG=0;
%                             if Hx >=0
%                                 break
%                             end
%                         end
%                         if check1*check2'==0
%                             break
%                         end
%                     end
%                 end
%                 if sum(check1)==0
%                     FLAG=0;
%                     %no avalanching needed
%                 end
%             elseif sum(check2)==0 && sum(check1)~=0
%                 FLAG=0;
%                 %fprintf('Too many plants!')
%             elseif sum(check1)==0
%                 FLAG=0;     %only other case should be if both sum to 0
%             end
```

```
%
%                    else%{if  Hstar(R,C)<=0}
%                        Erosioncheck=0;
%                        FLAG=0;
%                    end
%                    FLAG=0;
%                end
%            end
%        end
%
% end
% end


%Local functions for finding angle of repose and probability WRT plant cover
function B1=newbeta1(N,Hx)
    %B1=zeros(1,length(N));
    B1=atan(((Hx-N)*delta)/L);


%       for k=1:length(N)
%       B1(k)=atan(((Hx-N(k))*delta)/L);
%       end
end

function B2=newbeta2(N,beta1)
    %B2=zeros(1:length(N));
    B2=alpha*((beta1/(AoR))*(1-PC(R,C)));

%       for k=1:length(N)
%       B2(k)=alpha*((beta1(k)/(AoR))*(1-PC(R,C)));
%       end
end

function C1=newcheck1(N,beta1)
    %C1=zeros(1,length(N));
    C1=(beta1>=(AoR));
%       for k=1:length(N)
%       C1(k)=(beta1(k)>=(AoR));
%       end
end

function C2=newcheck2(N,beta2)
    %C2=zeros(1,length(N));
    C2=(RAND<beta2);
%       for k=1:length(N)
%       C2(k)=(RAND<beta2(k));
%       end
end

%FLAGCHECK=Hstarflag-Hstar;
%if sum(sum(Hstarflag-Hstar))~=0
    %flag=1;
%end
```

```
% toc
end
```

## B.4 Plant Propagation code

```matlab
function [P1,P2,P3,P4,P3d]=PlantPropagation03312021(Hstar,t,P1,P2,P3,P4,W,S,delta,P3d,MaxSwampWidth,PlantRangeArray,alpha,DBE,gdrange1,gdrange2,PctMax,MasterMax,MWSL,MESL,ESL)
% function [P1,P2,P3,P4,P3d]=PlantPropagation03312021(Hstar,P1,P2,P3,P4,W,S,delta,P1Burial,P2Burial,P3Burial,P4Burial,P3d,MaxSwampWidth,t)

test=zeros(size(P1,1),size(P2,2));
clims=[0 0];

P1B4=P1(:,:,1);
P2B4=P2(:,:,1);
P3B4=P3(:,:,1);
P4B4=P4(:,:,1);

% first loop does all propagating and death of all populations,
% second loop is death by competition for cells > MasterMAX


%   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! ')
%recently moved to main code:
% PlantRangeArray=[1 5;0.75 3;1.5 2.5;-0.5 1]; %all of the elevation ranges for p1-p4
% alpha=.01; %propagation rate for each populated cell
% DBE=.3;      %death by elevation rate for each populated cell outside of plant's elevation range
% gdrange1=[-.02:.01:.08]; %range of percent values for growth/death for plant populations at (0, 50)% cover
% gdrange2=[-.02:.01:.08];%[-.04:.01:.04]; %range of percent values for growth/death for plant pops greater than 50% cover
% P1PctMax=.6;                   %largest percentage we will allow any plant population on a given cell to attain
% P2PctMax=.6;
% P3PctMax=.8;
% P4PctMax=.8;
% PctMax=[P1PctMax P2PctMax P3PctMax P4PctMax];
% MasterMax=0.8;% The most any cell can permit - 80% plant coverage


SwampWidth=MaxSwampWidth; % number of cells wide that the swamp should be - should find a better way of establishing, 10 'looks right' for now
WesternCells=zeros(1,SwampWidth);

dH=delta*Hstar(:,:,1);           %to make sure everything works correctly in this subroutine we work in meters and not slabs - since plant data is in meters


```

```
%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%DEATH BY WATER TABLE STUFF%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%not using-we have(?) data, but I've never seen us use water table concept


% HWcheck=H(:,:,1)-W(:,:);
% HWcheck1=(HWcheck>0);
% HWcheck2a(:,:)=(delta*HWcheck>=-0.5);
% HWcheck2b=(delta*HWcheck<=1);

% for i=1:size(H,1)
%     for j=1:size(H,2)
%         if H(i,j,2)~=-1
%         if HWcheck2a(i,j)==1
%             if HWcheck2b(i,j) ==1
%                 if H(i,j,2)==2
%                     if P4(i,j,1)==-999
%                         P4(i,j,1)=0;
%                     end
%                 end
%             end
%         end
%
%         if HWcheck1(i,j)==1
%             if P1(i,j,1)==-999
%                 P1(i,j,1)=0;
%             end
%
%             if P2(i,j,1)==-999
%                P2(i,j,1)=0;
%             end
%
%             if P3(i,j,1)==-999
%                 P3(i,j,1)=0;
%             end
%         end
%         if HWcheck1(i,j)==0
%             P1(i,j,1)=-999;
%             P1(i,j,2)=0;
%             P2(i,j,1)=-999;
%             P2(i,j,2)=0;
%             P3(i,j,1)=-999;
%             P3(i,j,2)=0;
%         end
%         end
%     end
% end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%GROWTH/DEATH/PROP%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%splitting P_i into arrays for P_i(:,:,1) and P_i(:,:,2)
PA=zeros(size(Hstar,1),size(Hstar,2),4); %(P)lant (A)rray for tracking plant pct cover in growth/death/prop loop
PA(:,:,1)=P1(:,:,1);
PA(:,:,2)=P2(:,:,1);
PA(:,:,3)=P3(:,:,1);
PA(:,:,4)=P4(:,:,1);
PHA=zeros(size(Hstar,1),size(Hstar,2),4); %(P)lant (H)eight (A)rray for tracking plant init. elev. in growth/death/prop loop
PHA(:,:,1)=P1(:,:,2);
PHA(:,:,2)=P2(:,:,2);
PHA(:,:,3)=P3(:,:,2);
PHA(:,:,4)=P4(:,:,2);

% plants do not propagate during initialization, skip straight to beath by competition
 if isnan(t)==0
%   Beginning of the year, plants use full gdrange (death and growth)
    if mod(t,26)==0
        for i=1:size(PA,3)
            PAB4=PA(:,:,i);
            Px=PA(:,:,i);
            Ph=PHA(:,:,i);
            for R=1:size(Hstar,1)
                for C=1:size(Hstar,2)
                    if dH(R,C)<-0.5
                        Px(R,C)=-999;
                        Ph(R,C)=0;
                    elseif ESL(R)~=0 && C>ESL(R)
                        Px(R,C)=-999;
                        Ph(R,C)=0;
                    else
                        if i==3
                            P3d(R,C,2)=P3d(R,C,2)-1*(P3d(R,C,2)>0);      %if i=3 (working on P3) we remove a counter from the p3d array if there is a value stored there
                            %                     if P3d(R,C,2)==0
                            %                         P3(R,C,1)=0;
                            %                     end
                            if dH(R,C)<0 && P3d(R,C,1)>0                  %if elev. goes below zero we get rid of dead morella
                                P3d(R,C,1)=0;
                                P3d(R,C,2)=0;
                            end
                        end
                        if i~=4                                           %work on all Pi arrays except p4 first
                            if (dH(R,C)<PlantRangeArray(i,1)) || (dH(R,C)>PlantRangeArray(i,2)) %if we are outside of the elevation range for this plant...
                                if Px(R,C)~=-999                                               %but we aren't underwater...
                                    if i==3 && Px(R,C)>0 && dH(R,C)>0 &&t~=0          %if it's p3 outside of elevation range we have to create a dead morella value
                                        PxB4=Px(R,C);                             %store p3 %cover before death by elev.
                                        Px(R,C)=Px(R,C) - DBE*Px(R,C);%-999;          %changed on 9/21 so that death by elevation is not a sudden drop to 0
                                        if PxB4>0.01 && Px(R,C)<0.01               %if newly below 1% we create a p3d cell and kill off remaining p3
                                            Px(R,C)=-999;
```

```
                Ph(R,C)=0;
                P3d(R,C,1)=.05;
                P3d(R,C,2)=10;
            end
        elseif Px(R,C) >0 && dH(R,C)>=0                                        %if it's p1 or p2 outside of elevation we remove a percentage
            Px(R,C)=Px(R,C)-DBE*Px(R,C);%-999;          %changed on 9/21 see above
            if Px(R,C)<0.01                              %if that percentage drops below .1% we kill it
                Px(R,C) = 0;                             %since it is outside of it's elev. range we negate the cell for future popln.
                Ph(R,C)=0;
            end
        else                                             %should only have cells now underwater which we will negate

            Ph(R,C)=0;
        end
    end
end
if (dH(R,C)>=PlantRangeArray(i,1)) && (dH(R,C)<=PlantRangeArray(i,2))   %if we are inside of the plant elevation range
    %                  resetting negated cells which were previously submerged
    if Px(R,C)==-999
        Px(R,C)=0;
        Ph(R,C)=Hstar(R,C,1);
    end
    %                  find the neighborhood, calc new popln for
    %                  growth/propgatn :
    %                    some % * (how many neighbors are
    %                  populated)
    Nhd=NewNhd(R,C,PAB4);
    if Px(R,C) <.5
        PxB4=Px(R,C);
        kk=randi([1,length(gdrange1)]);
        beta=gdrange1(kk);
        Px(R,C)=min(Px(R,C)+sum(beta*Nhd)+beta*(beta>0)*(i==3),PctMax(i));          %grow by sum of neighbors* beta, if it's morella grow a little more?
        if i==3 && PxB4>=0.05 && Px(R,C)<=.05                                        % add's chance of spread by birds anywhere inside of p3 range (same below)
            Px(R,C)=0;
            Ph(R,C)=0;
            P3d(R,C,1)=.05;
            P3d(R,C,2)=5;
        end
    else
        kk=randi([1,length(gdrange2)]);
        beta=gdrange2(kk);
        Px(R,C)=min(Px(R,C)+sum(beta*Nhd)+beta*(beta>0)*(i==3),PctMax(i));          %grow by sum of neighbors* beta, if it's morella grow a little more?
    end
    %                 record elevations for all cells inside the range,
    %                 elevation=zero if no popln exists on cell currently
    if Px(R,C)==0
        Ph(R,C)=0;
    elseif Px(R,C)>0
        Ph(R,C)=Hstar(R,C,1);
```

```
                elseif Px(R,C)<0
                    Px(R,C)=0;
                    Ph(R,C)=0;
                end
            end
        elseif i==4
            if (dH(R,C)<PlantRangeArray(i,1)) %if less than min height (-0.5)
                if Px(R,C)~=-999    %negate (should already be done from first loop after R,C declared

                    Ph(R,C)=0;
                end
            elseif (dH(R,C)>PlantRangeArray(i,2)) %elseif greater than max height, make 0 (no death by elev. just kill)
                Px(R,C)=0;
                Ph(R,C)=0;
            end
            if (dH(R,C)>=PlantRangeArray(i,1)) && (dH(R,C)<=PlantRangeArray(i,2))
                if ESL(R)~=0
                    if MWSL(R)<=C && MESL(R)>=C
                        Nhd=NewNhd(R,C,Px);
                        if Px(R,C) <.5
                            kk=randi([1,length(gdrange1)]);
                            beta=gdrange1(kk);
                            Px(R,C)=min(Px(R,C)+sum(beta*Nhd)+beta*(beta>0)*(i==3),PctMax(i));   %this also kills by a

                        else
                            kk=randi([1,length(gdrange2)]);
                            beta=gdrange2(kk);
                            Px(R,C)=min(Px(R,C)+sum(beta*Nhd)+beta*(beta>0)*(i==3),PctMax(i));
                        end
                    else

                    end
                end
                Nhd=NewNhd(R,C,Px);
                if Px(R,C) <.5
                    kk=randi([1,length(gdrange1)]);
                    beta=gdrange1(kk);
                    Px(R,C)=min(Px(R,C)+sum(beta*Nhd)+beta*(beta>0)*(i==3),PctMax(i));   %this also kills by a
                else
                    kk=randi([1,length(gdrange2)]);
                    beta=gdrange2(kk);
                    Px(R,C)=min(Px(R,C)+sum(beta*Nhd)+beta*(beta>0)*(i==3),PctMax(i));
                end
                %           record elevations for all cells inside the range,
                %           elevation=zero if no popln exists on cell currently
                if Px(R,C)==0
                    Ph(R,C)=0;
                elseif Px(R,C)>0
                    Ph(R,C)=Hstar(R,C,1);
                elseif Px(R,C)<0 && Px(R,C)~=-999
                    Px(R,C)=0;
                    Ph(R,C)=0;
                end
            end
```

```
                        end
                    end
                end
            end
        PA(:,:,i)=Px;
        PHA(:,:,i)=Ph;
        end


%   Half year, plants only grow (no death, simulate spring time growth)
    else
        Spr_gdrange1=gdrange1(gdrange1>0);
        Spr_gdrange2=gdrange2(gdrange2>0);
        for i=1:size(PA,3)
            PAB4=PA(:,:,i);
            Px=PA(:,:,i);
            Ph=PHA(:,:,i);
            for R=1:size(Hstar,1)
                for C=1:size(Hstar,2)
                    if dH(R,C)<-.5
                        Px(R,C)=-999;
                        Ph(R,C)=0;
                    elseif ESL(R)>0 && C>ESL(R)
                        Px(R,C)=-999;
                        Ph(R,C)=0;
                    else
                        if i==3
                            if dH(R,C)<0 && P3d(R,C,1)>0                    %if elev. goes below zero we get rid of dead morella
                                P3d(R,C,1)=0;
                                P3d(R,C,2)=0;
                            end
                        end
                        if i~=4                                             %work on all Pi arrays except p4 first
                            if (dH(R,C)<PlantRangeArray(i,1)) || (dH(R,C)>PlantRangeArray(i,2)) %if we are outside of the elevation range for this plant...
                                if Px(R,C)~=-999                                                   %but we aren't underwater...
                                    if i==3 && Px(R,C)>0 && dH(R,C)>0 &&t~=0           %if it's p3 outside of elevation range we have to create a dead morella value
                                        PxB4=Px(R,C);                                %store p3 %cover before death by elev.
                                        Px(R,C)=Px(R,C) - DBE*Px(R,C);%-999;            %changed on 9/21 so that death by elevation is not a sudden drop to 0
                                        if PxB4>0.01 && Px(R,C)<0.01               %if newly below 1% we create a p3d cell and kill off remaining p3
                                            Px(R,C)=0;
                                            Ph(R,C)=0;
                                            P3d(R,C,1)=.05;
                                            P3d(R,C,2)=5;
                                        end
                                    elseif Px(R,C) >0 && dH(R,C)>=0                                              %if it's p1 or p2 outside of elevation we remove a percentage
                                        Px(R,C)=Px(R,C)-DBE*Px(R,C);%-999;             %changed on 9/21 see above
                                        if Px(R,C)<0.001                              %if that percentage drops below .1% we kill it
                                            Px(R,C) = 0;                              %since it is outside of it's elev. range we drop to 0, will negate after elev drops below-0.5.
                                            Ph(R,C)=0;
                                        end
```

```
                        else                                    %should only have cells now underwater which we will negate
                            Px(R,C)=0;
                            Ph(R,C)=0;
                        end
                    end
                end
                if i~=3 && (dH(R,C)>=PlantRangeArray(i,1)) && (dH(R,C)<=PlantRangeArray(i,2))   %if we are inside of the plant elevation range, don't do morella this time
                    %                   resetting negated cells which were previously submerged
                    if Px(R,C)==-999
                        Px(R,C)=0;
                        Ph(R,C)=Hstar(R,C,1);
                    end
                    %                   find the neighborhood, calc new popln for
                    %                   growth/propgatn :
                    %                     some % * (how many neighbors are
                    %                   populated)
                    Nhd=NewNhd(R,C,PAB4);
                    if Px(R,C) <.5
                        PxB4=Px(R,C);
                        kk=randi([1,length(Spr_gdrange1)]);
                        beta=Spr_gdrange1(kk);
                        Px(R,C)=min(Px(R,C)+sum(beta*Nhd)+beta*(beta>0)*(i==3),PctMax(i));          %grow by sum of neighbors* beta, if it's morella grow a little more?
                        if i==3 && PxB4>=0.05 && Px(R,C)<=.05                                       %add's chance of spread by birds anywhere inside of p3 range (same below)
                            Px(R,C)=0;
                            Ph(R,C)=0;
                            P3d(R,C,1)=.05;
                            P3d(R,C,2)=10;
                        end
                    else
                        kk=randi([1,length(Spr_gdrange2)]);
                        beta=Spr_gdrange2(kk);
                        Px(R,C)=min(Px(R,C)+sum(beta*Nhd)+beta*(beta>0)*(i==3),PctMax(i));          %grow by sum of neighbors* beta, if it's morella grow a little more?
                    end
                    %                 record elevations for all cells inside the range,
                    %                 elevation=zero if no popln exists on cell currently
                    if Px(R,C)==0
                        Ph(R,C)=0;
                    elseif Px(R,C)>0
                        Ph(R,C)=Hstar(R,C,1);
                    elseif Px(R,C)<0
                        Px(R,C)=0;
                        Ph(R,C)=0;
                    end
                end
            elseif i==4
                if (dH(R,C)<PlantRangeArray(i,1)) %if less than min height (-0.5)
                    if Px(R,C)~=-999    %negate (should already be done from first loop after R,C declared
                        Px(R,C)=-999;
                        Ph(R,C)=0;
                    end
                elseif (dH(R,C)>PlantRangeArray(i,2)) %elseif greater than max height, make 0 (no death by elev. just
```

```
                                kill)
                                Px(R,C)=0;
                                Ph(R,C)=0;
                            end
                            if (dH(R,C)>=PlantRangeArray(i,1)) && (dH(R,C)<=PlantRangeArray(i,2))
                                if ESL(R)~=0
                                    if MWSL(R)<C && MESL(R)>=C
                                        Nhd=NewNhd(R,C,Px);
                                        if Px(R,C) <.5
                                            kk=randi([1,length(gdrange1)]);
                                            beta=gdrange1(kk);
                                            Px(R,C)=min(Px(R,C)+sum(beta*Nhd)+beta*(beta>0)*(i==3),PctMax(i));   %this also kills
                                                by a

                                        else
                                            kk=randi([1,length(gdrange2)]);
                                            beta=gdrange2(kk);
                                            Px(R,C)=min(Px(R,C)+sum(beta*Nhd)+beta*(beta>0)*(i==3),PctMax(i));
                                        end
                                    else
                                        Px(R,C)=0;
                                    end
                                end
                                %               record elevations for all cells inside the range,
                                %               elevation=zero if no popln exists on cell currently
                                if Px(R,C)==0
                                    Ph(R,C)=0;
                                elseif Px(R,C)>0
                                    Ph(R,C)=Hstar(R,C,1);
                                elseif Px(R,C)<0 && Px(R,C)~=-999
                                    Px(R,C)=0;
                                    Ph(R,C)=0;
                                end
                            end
                        end
                    end
                end
            end
        PA(:,:,i)=Px;
        PHA(:,:,i)=Ph;
        end
    end
end


%now updating changes in original plant matrices
P1(:,:,1)=PA(:,:,1);
P2(:,:,1)=PA(:,:,2);
P3(:,:,1)=PA(:,:,3);
P4(:,:,1)=PA(:,:,4);

P1(:,:,2)=PHA(:,:,1);
P2(:,:,2)=PHA(:,:,2);
P3(:,:,2)=PHA(:,:,3);
P4(:,:,2)=PHA(:,:,4);

```

```
%new temp plant matrices removing the -999 values to get an accurate total
%of all poplns on cells
Pt1=max(P1(:,:,1),0);
Pt2=max(P2(:,:,1),0);
Pt3=max(P3(:,:,1),0);
Pt4=max(P4(:,:,1),0);

Ptot=Pt1+Pt2+Pt3+Pt4;
%Death by Comp
fprintf('\n')
fprintf('The plants are killing each other!!!')
for i =1:size(Hstar,1)
    for j =1:size(Hstar,2)
        if Ptot(i,j,1)>MasterMax
            Px1=Pt1(i,j,1);
            Px2=Pt2(i,j,1);
            Px3=Pt3(i,j,1);
            Px4=Pt4(i,j,1);
            if Px3==MasterMax
                P1(i,j,1)=0;
                P2(i,j,1)=0;
                P4(i,j,1)=0;
            elseif (Px3 < MasterMax) && (Px3>0)
                k=(MasterMax-Px3)/(Px1+Px2+Px4);
                P1(i,j,1)=Px1*k;
                P2(i,j,1)=Px2*k;
                P4(i,j,1)=Px4*k;
            elseif Px3==0
                %if Px1>=(k/4) && Px2>=(k/4) && Px3>=(k/4) && Px4>=(k/4)
                k=(Px1+Px2+Px4)-MasterMax;
                if Px1>=(k/3) && Px2>=(k/3) && Px4>=(k/3)
                    P1(i,j,1)=Px1-(k/3);
                    P2(i,j,1)=Px2-(k/3);
                    P4(i,j,1)=Px4-(k/3);
                    %Only two are bigger than k/3
                    %%1 2%%
                elseif Px1>=(k/3) && Px2>=(k/3) && Px4<(k/3)
                    P1(i,j,1)=Px1-((k/3)+(((k/3)-Px4)/2));
                    P2(i,j,1)=Px2-((k/3)+(((k/3)-Px4)/2));
                    P4(i,j,1)=0;
                    %%1 4%%
                elseif Px1>=(k/3) && Px2<(k/3) && Px4>=(k/3)
                    P1(i,j,1)=Px1-((k/3)+(((k/3)-Px4)/2));
                    P2(i,j,1)=0;
                    P4(i,j,1)=Px4-((k/3)+(((k/3)-Px2)/2));
                    %%2 4%%
                elseif Px1<(k/3) && Px2>=(k/3) && Px4>=(k/3)
                    P1(i,j,1)=0;
                    P2(i,j,1)=Px2-((k/3)+(((k/3)-Px1)/2));
                    P4(i,j,1)=Px4-((k/3)+(((k/3)-Px1)/2));
                    %Only one is bigger than k/3%
                    %%1%%
                elseif Px1>=(k/3) && Px2<(k/3) && Px4<(k/3)
                    P1(i,j,1)=Px1-((k/3)+((k/3)-Px2)+((k/3)-Px4));
                    P2(i,j,1)=0;
                    P4(i,j,1)=0;
```

```
                    %%2%%
                elseif Px1<(k/3) && Px2>=(k/3) && Px4<(k/3)
                    P1(i,j,1)=0;
                    P2(i,j,1)=Px2-((k/3)+((k/3)-Px1)+((k/3)-Px4));
                    P4(i,j,1)=0;
                    %%4%%
                elseif Px1<(k/3) && Px2<(k/3) && Px4>=(k/3)
                    P1(i,j,1)=0;
                    P2(i,j,1)=0;
                    P4(i,j,1)=Px4-((k/3)+((k/3)-Px1)+((k/3)-Px2));
                end
            end
        end
    end
end


%currently run this outside of PlantProcesses
for i=1:size(Hstar,1)
    for j=1:size(Hstar,2)
        PC1(i,j)=P1(i,j,1)*(P1(i,j,1)>0);
        PC2(i,j)=P2(i,j,1)*(P2(i,j,1)>0);
        PC3(i,j)=P3(i,j,1)*(P3(i,j,1)>0);
        PC4(i,j)=P4(i,j,1)*(P4(i,j,1)>0);

    end
end
Ptot=PC1+PC2+PC3+PC4;


    function N1=NewNhd(R,C,PAB4)
        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
        N1=zeros(1,8);
```

```
            %       PB4=PA(:,:,i);
            N1=[PAB4(Rm,Cm),PAB4(Rm,C),PAB4(Rm,Cp),PAB4(R,Cp),PAB4(Rp,Cp),PAB4(Rp,C),PAB4(Rp,Cm),PAB4(R,Cm)];
            for k=1:8
                if N1(k)==-999
                    N1(k)=0;
                elseif N1(k)<0.01
                    N1(k)=0;
                end
            end
        end

end
```

## B.5 Shoreline code

```
function [Hstar,MeanBeachWidth,ESL,WSL,MESL,MWSL,OL]=Shoreline03312021(Hstar,delta,L,BchMax)
%this function is just to return the eastern shoreline for plant initialization
%P4 not permitted to grow east of ESL - (E)astern (S)hore (L)ine

[n1 n2]=size(Hstar(:,:,1));
AL=zeros(n1,1); %AdjascentLength~width of beach
OL=zeros(n1,1); %OppositeLocation~innermost reach of beach/index of first cell>BchMax
ESL=zeros(n1,1);        %EAST shoreline (column value per row;
WSL=zeros(n1,1);      %formerly "WesternShore"
MESL=zeros(n1,1);    %formerly "SlineSwamp"
MWSL=zeros(n1,1);    %formerly "WBndrySwamp"

beta=zeros([n1 1]);              %this gets called but isn't currently used, stores profile slope
Beta=zeros([n1 1]);              %bruun rule beta

R=zeros(n1,1);
DC=zeros(n1,2);


dH=delta*Hstar(:,:,1);  %making new array of Hstar values in terms of meters instead of slabs (ease of use)

for i=1:n1
    Rnow=dH(i,:);
    if Rnow(n2)<0
        for j1=n2-1:-1:2                 %declaring eastern shoreline by looking for first cell with positive elevation west of a cell with negative elevation
            if Rnow(j1)>=0 && Rnow(j1+1)<=0 && ESL(i)==0
                if Rnow(j1-1)~=0
                    ESL(i)=j1;
                end
            end
            if j1==2 && ESL(i)==0
                ESL(i)=ESL(i)+1*(Rnow(1)>=0);            %special condition for first column - avoid index errors
            end
            if ESL(i)~=0 && ESL(i)~=1                   %if we found a shoreline (that wasn't in col 1) need to declare AL and OL
                DC(i,2)=max(Rnow);                      %will use max of current row if no cell satisfies being greater than BchMax
                DC(i,1)=find(Rnow==DC(i,2),1,'last');
                flag=0;
                while flag==0
                    for j2=j1-1:-1:1                            % looking for dune crest starting with shoreline and moving west
                        if Rnow(j2)>=BchMax                     % if we find a cell >= BchMax
                            OL(i)=j2;                           % then that cell is the dune crest
                            AL(i)=j1-j2;                        % width of the beach is shoreline-dunecrest (indexes)
                            flag=1;
                            break
                        elseif Rnow(j2)<0                       % if we go below water
                            m=1;
                            while j2-m>=1
                                if Rnow(j2-m)>0
                                    j2=j2-m;                    %if we get back above water, change j2 and keep looking for swamp/dune crest
                                    m=1;
```

```
                                elseif Rnow(j2-m)<=-0.5
                                    OL(i)=(j2)+find(Rnow(j2+1:j1)==max(Rnow(j2+1:j1)),1,'last');                % make the dune crest the max height of the positive elevation portion of the island
                                    AL(i)=j1-OL(i);                    %width of beach is shoreline index - the index of max height of subaerial island
                                    flag=1;
                                    break
                                elseif m==j2-1
                                    OL(i)=DC(i,1);                     %if we search the rest of the row and don't find swamp/dune crest use max(row) as dune crest
                                    AL(i)=j1-OL(i);
                                    m=j2;
                                    flag=1;
                                else
                                    m=m+1;
                                end

                            end
                            if j2==1 %j2==2                    %if we
                                OL(i)=DC(i,1);                 % make the dune crest the max height of the positive elevation portion of the island
                                AL(i)=j1-OL(i);                    %width of beach is shoreline - the max height of subaerial island
                                flag=1;
                            end
                        elseif j2==1
                            OL(i)=DC(i,1);                 % make the dune crest the max height of the positive elevation portion of the island
                            AL(i)=j1-OL(i);                    %width of beach is shoreline - the max height of subaerial island
                            flag=1;
                        end
                        if OL(i)>0 && AL(i)>0
                            break
                        end
                    end
                end
            end
            if OL(i)>0 && AL(i)>0
                break
            end
        end
    end
end

for ii=1:n1
    if ESL(ii)~=0
        Rnow=dH(ii,:);
        MWSL(ii)=find(Rnow>=-0.5,1,'first');
        DC(ii)=find(Rnow==max(Rnow),1,'first');
        SwmpChk=Rnow(MWSL(ii)+1:DC(ii));
        j=find(SwmpChk>1,1,'first');
        if isempty(j)==1
            MESL(ii)=DC(ii);
        else
            MESL(ii)=MWSL(ii)+j;
        end
    end
```

```
end

MBWfactor=(AL~=0);  %calculating changes in beach width - does nothing unless used outside fcn in an image
MeanBeachWidth=(sum(AL(MBWfactor)))/sum(MBWfactor); %same

end
```

## B.6 Marine Processes code

```
function [Hstar,P3,MeanBeachWidth,ESL,WSL,MESL,MWSL,OL,SLRyrs,MigCnt,MigAccel]=MarineProcesses03312021(Hstar,delta,L,P1,P2,P3,P4,BchMax,OW,t,SLRyrs,PCmp,IslandArea,MigCnt,ScaleFactor,MigYr,Ma,MigAccel,MasterMax)
alpha=1;
AoR=pi/6;


T=t/26;
AccelCheck=MigYr*(Ma)^T
if floor(AccelCheck)>MigYr
    Mig=floor(AccelCheck);
elseif AccelCheck==0
    Mig=AccelCheck;
else
    Mig=MigYr;
end

nn=5;   %number of rows above and below current row to check plant density when calculating migration

if isnan(t)==1
    Tflag=1;
    t=0;
else
    Tflag=0;
end

test=0;
SLR=0.00635;    %was for Bruun Rule testing - not currently used
Htest=0; %handy in editing to just declare this
Ptest=0; %handy in editing to just declare this

[n1 n2]=size(Hstar(:,:,1));
AL=zeros(n1,1); %AdjacentLength~width of beach
OL=zeros(n1,1); %OppositeLocation~innermost reach of beach/index of first cell>BchMax
ESL=zeros(n1,1);        %EAST shoreline (column value per row;
WSL=zeros(n1,1);     %formerly "WesternShore"
MESL=zeros(n1,1);    %formerly "SlineSwamp"
MWSL=zeros(n1,1);    %formerly "WBndrySwamp"

beta=zeros([n1 1]);             %this gets called but isn't currently used, stores profile slope
Beta=zeros([n1 1]);             %bruun rule beta

R=zeros(n1,1);
DC=zeros(n1,2);


dH=delta*Hstar(:,:,1);  %making new array of Hstar values in terms of meters instead of slabs (ease of use)

for i=1:n1
    Rnow=dH(i,:);
    Pnow=P3(i,:,1);
    PHnow=P3(i,:,2);
    if Rnow(n2)<0
        for j1=n2-1:-1:2                %declaring eastern shoreline by looking for first cell with positive elevation west of a cell with negative elevation
```

```
        if Rnow(j1)>=0 && Rnow(j1+1)<=0 && ESL(i)==0
            if Rnow(j1-1)~=0
                ESL(i)=j1;
            end
        end
        if j1==2 && ESL(i)==0
            ESL(i)=ESL(i)+1*(Rnow(1)>=0);           %special condition for first column - avoid index errors
        end
        if ESL(i)~=0 && ESL(i)~=1                   %if we found a shoreline (that wasn't in col 1) need to declare AL and OL
            DC(i,2)=max(Rnow);                      %will use max of current row if no cell satisfies being greater than BchMax
            DC(i,1)=find(Rnow==DC(i,2),1,'last');
            flag=0;
            while flag==0
                for j2=j1-1:-1:1                            % looking for dune crest starting with shoreline and moving west
                    if Rnow(j2)>=BchMax                     % if we find a cell >= BchMax
                        OL(i)=j2;                           % then that cell is the dune crest
                        AL(i)=j1-j2;                        % width of the beach is shoreline-dunecrest (indexes)
                        flag=1;
                        break
                    elseif Rnow(j2)<0                       % if we go below water
                        m=1;
                        while j2-m>=1
                            if Rnow(j2-m)>0
                                j2=j2-m;                    %if we get back above water, change j2 and keep looking for swamp/dune crest
                                m=1;
                            elseif Rnow(j2-m)<=-0.5
                                OL(i)=(j2)+find(Rnow(j2+1:j1)==max(Rnow(j2+1:j1)),1,'last');            % make the dune crest the max height of the positive elevation portion of the island
                                AL(i)=j1-OL(i);             %width of beach is shoreline index - the index of max height of subaerial island
                                flag=1;
                                break
                            elseif m==j2-1
                                OL(i)=DC(i,1);              %if we search the rest of the row and don't find swamp/dune crest use max(row) as dune crest
                                AL(i)=j1-OL(i);
                                m=j2;
                                flag=1;
                            else
                                m=m+1;
                            end

                        end
                        if j2==1 %j2==2                 %if we
                            OL(i)=DC(i,1);              % make the dune crest the max height of the positive elevation portion of the island
                            AL(i)=j1-OL(i);                 %width of beach is shoreline - the max height of subaerial island
                            flag=1;
                        end
                    elseif j2==1
                        OL(i)=DC(i,1);              % make the dune crest the max height of the positive elevation portion of the island
                        AL(i)=j1-OL(i);                 %width of beach is shoreline - the max height of subaerial island
```

```
                            flag=1;
                        end
                        if OL(i)>0 && AL(i)>0
                            break
                        end
                    end
                end
            end
            if OL(i)>0 && AL(i)>0
                break
            end
        end
        if ESL(i)>1                   %if the island has been found in this row
            for m=0:AL(i)             %removing P3 from the beach
                j=OL(i)+m;
                if Pnow(j)>0
                    Pnow(j)=0;
                    PHnow(j)=0;
                end
            end
            P3(i,:,1)=Pnow;
            P3(i,:,2)=PHnow;
        elseif ESL(i)==1         %if shoreline is first cell make OL, AL first cell
            OL(i)=1;
            AL(i)=1;
            if Pnow(1)>0
                Pnow(1)=0;
                PHnow(1)=0;
            end
        end
    end
end

MBWfactor=(AL~=0);  %calculating changes in beach width - does nothing unless used outside fcn in an image
MeanBeachWidth=(sum(AL(MBWfactor)))/sum(MBWfactor); %same



%now calculating the foreshore slope and resetting cells
cnt=0;
DofCi=zeros(n1,1);
xdoc=sym('xdoc');

for i=1:n1
    if  ESL(i)>0
        cnt=cnt+1;
        Rnow=dH(i,:);
        if AL(i)~=0 %if we have a shoreline and the adjascent length is not zero
            beta(i)=tan(Rnow(OL(i))/AL(i)); %calculate the slope of the shoreline
            R(i)=1/beta(i);                     %part of Bruun rule - unused
        elseif AL(i)==0
            beta(i)=.0003;    %if there is a shore that is one cell wide (special condition above) use common shore slope
            R(i)=1/beta(i);
        end
    end
end
```

```
cnt=0;
shortR=0; %for removing NaN and infinity slopes - possible with holes and ponds, but unlikely (pretty sure I debugged this issue)
for i=1:n1
    if ESL(i)>0
        cnt=cnt+1;
        shortR(cnt)=R(i);%*(isnan(R(i)==0)); %uncomment this if NaN issue (see above) comes up
    end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%for-Vector-of-Migration-Years%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%below is for using a vector of predetermined years to trigger migration
% YrsPast=sum(SLRyrs(SLRyrs<0));                  %any negative years in vector are summed
% YrCnt=(t/26)+YrsPast;                           %current number of years since last migration
% if YrCnt==max(SLRyrs(SLRyrs>0))                 %if current #yrs is the next number of years in vector of vaues to trigger migration
%     MigChk=1;                                   %trigger migration
%     kk=find(SLRyrs==max(SLRyrs(SLRyrs>0)))      %find that yr value in vector
%     SLRyrs(kk)=-SLRyrs(kk);                     %replace with negative year so it will be summed as years past
% else
%     MigChk=0;
% end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%END%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%For_Yearly_Migration%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

Plx=zeros(n1,1);
if 0==mod(t,26)% && t~=0
    MigChk=1;
    if MigChk==1
        for i=1:n1
            if i<nn
                PlChkArea=PCmp(1:1+nn,:,1);  %plant cover for rows near current row
                Plx(i)=mean(PlChkArea(PlChkArea>=0));
                if sum(dH(i,:)>=0)==0
                    MigCnt(i)=MigCnt(i)+Mig;
                else
                    if Plx(i) >= 0.3
                        MigCnt(i)=MigCnt(i)+round(0.00*Mig);   %reduce migration by 70% if nearby weighted plant cover exceeds 50%
                    elseif Plx(i) >= 0.2 && Plx(i) < 0.3
                        MigCnt(i)=MigCnt(i)+round(0.3*Mig);   %reduce migration by 50% if nearby weighted plant cover exceeds 35% but not 50%
                    elseif Plx(i) >= 0.1 && Plx(i) < 0.1
                        MigCnt(i)=MigCnt(i)+round(0.5*Mig);   %reduce migration by 30% if nearby weighted plant cover exceeds 10% but not 35%
                    else
                        MigCnt(i)=MigCnt(i)+Mig;  %full effect of migration if nearby plant cover less than 10%
                    end
                end
                if MigCnt(i)>ScaleFactor
                    %                    MigR=1;                                  %set MigR=1 if using scaled version of migration
```

```
                MigR=floor(MigCnt(i)/ScaleFactor);
                fprintf('\n')
                fprintf('THE SHORELINE AT ROW %d IS MIGRATING EAST BY %d COLUMNS!',i,MigR)
                RnowDummy=zeros(1,n2);
                P1RnowDummy=RnowDummy;
                P2RnowDummy=RnowDummy;
                P3RnowDummy=RnowDummy;
                P4RnowDummy=RnowDummy;
                RnowDummy(1:n2-MigR)=dH(i,MigR+1:n2);
                P1RnowDummy(1:n2-MigR)=P1(i,MigR+1:n2);
                P2RnowDummy(1:n2-MigR)=P1(i,MigR+1:n2);
                P3RnowDummy(1:n2-MigR)=P1(i,MigR+1:n2);
                P4RnowDummy(1:n2-MigR)=P1(i,MigR+1:n2);
                dH(i,1:n2-MigR)=RnowDummy(1:n2-MigR);
                P1(i,1:n2-MigR)=P1RnowDummy(1:n2-MigR);
                P2(i,1:n2-MigR)=P1RnowDummy(1:n2-MigR);
                P3(i,1:n2-MigR)=P1RnowDummy(1:n2-MigR);
                P4(i,1:n2-MigR)=P1RnowDummy(1:n2-MigR);
                MigCnt(i)=MigCnt(i)-MigR*ScaleFactor;
            end
        elseif nn<i && i<n1-nn
%               if i>2000
%                   fprintf('oo')
%               end
            PlChkArea=PCmp(i-nn:i+nn,:,1);  %plant cover for nows near current row
            Plx(i)=mean(PlChkArea(PlChkArea>=0));
            if sum(dH(i,:)>=0)==0
                MigCnt(i)=MigCnt(i)+Mig;
            else
                if Plx(i) >= 0.3
                    MigCnt(i)=MigCnt(i)+round(0.00*Mig);   %reduce migration by 70% if nearby weighted plant cover exceeds 50%
                elseif Plx(i) >= 0.2 && Plx(i) < 0.3
                    MigCnt(i)=MigCnt(i)+round(0.3*Mig);   %reduce migration by 50% if nearby weighted plant cover exceeds 35% but not 50%
                elseif Plx(i) >= 0.1 && Plx(i) < 0.2
                    MigCnt(i)=MigCnt(i)+round(0.5*Mig);   %reduce migration by 30% if nearby weighted plant cover exceeds 10% but not 35%
                else
                    MigCnt(i)=MigCnt(i)+Mig;  %full effect of migration if nearby plant cover less than 10%
                end
            end
            if MigCnt(i)>ScaleFactor
                %                MigR=1;                              %set MigR=1 if using scaled version of migration
                MigR=floor(MigCnt(i)/ScaleFactor);
                fprintf('\n')
                fprintf('THE SHORELINE AT ROW %d IS MIGRATING EAST BY %d COLUMNS!',i,MigR)
                RnowDummy=zeros(1,n2);
                P1RnowDummy=RnowDummy;
                P2RnowDummy=RnowDummy;
                P3RnowDummy=RnowDummy;
                P4RnowDummy=RnowDummy;
                RnowDummy(1:n2-MigR)=dH(i,MigR+1:n2);
                P1RnowDummy(1:n2-MigR)=P1(i,MigR+1:n2);
                P2RnowDummy(1:n2-MigR)=P1(i,MigR+1:n2);
                P3RnowDummy(1:n2-MigR)=P1(i,MigR+1:n2);
```

```
                    P4RnowDummy(1:n2-MigR)=P1(i,MigR+1:n2);
                    dH(i,1:n2-MigR)=RnowDummy(1:n2-MigR);
                    P1(i,1:n2-MigR)=P1RnowDummy(1:n2-MigR);
                    P2(i,1:n2-MigR)=P1RnowDummy(1:n2-MigR);
                    P3(i,1:n2-MigR)=P1RnowDummy(1:n2-MigR);
                    P4(i,1:n2-MigR)=P1RnowDummy(1:n2-MigR);
                    MigCnt(i)=MigCnt(i)-MigR*ScaleFactor;
                end
            elseif i>n1-nn
                PlChkArea=PCmp(n1-nn:n1,:,1);  %plant cover for nows near current row
                Plx(i)=mean(PlChkArea(PlChkArea>=0));
                if sum(dH(i,:)>=0)==0
                    MigCnt(i)=MigCnt(i)+Mig;
                else
                    if Plx(i) >= 0.5
                        MigCnt(i)=MigCnt(i)+round(0.00*Mig);   %reduce migration by 70% if nearby weighted plant cover exceeds 50%
                    elseif Plx(i) >= 0.35 && Plx(i) < 0.5
                        MigCnt(i)=MigCnt(i)+round(0.3*Mig);   %reduce migration by 50% if nearby weighted plant cover exceeds 35% but not 50%
                    elseif Plx(i) >= 0.1 && Plx(i) < 0.35
                        MigCnt(i)=MigCnt(i)+round(0.5*Mig);   %reduce migration by 30% if nearby weighted plant cover exceeds 10% but not 35%
                    else
                        MigCnt(i)=MigCnt(i)+Mig;  %full effect of migration if nearby plant cover less than 10%
                    end
                end
                if MigCnt(i)>ScaleFactor
                    %                    MigR=1;                                       %set MigR=1 if using scaled version of migration
                    MigR=floor(MigCnt(i)/ScaleFactor);
                    fprintf('\n')
                    fprintf('THE SHORELINE AT ROW %d IS MIGRATING EAST BY %d COLUMNS!',i,MigR)
                    RnowDummy=zeros(1,n2);
                    P1RnowDummy=RnowDummy;
                    P2RnowDummy=RnowDummy;
                    P3RnowDummy=RnowDummy;
                    P4RnowDummy=RnowDummy;
                    RnowDummy(1:n2-MigR)=dH(i,MigR+1:n2);
                    P1RnowDummy(1:n2-MigR)=P1(i,MigR+1:n2);
                    P2RnowDummy(1:n2-MigR)=P1(i,MigR+1:n2);
                    P3RnowDummy(1:n2-MigR)=P1(i,MigR+1:n2);
                    P4RnowDummy(1:n2-MigR)=P1(i,MigR+1:n2);
                    dH(i,1:n2-MigR)=RnowDummy(1:n2-MigR);
                    P1(i,1:n2-MigR)=P1RnowDummy(1:n2-MigR);
                    P2(i,1:n2-MigR)=P1RnowDummy(1:n2-MigR);
                    P3(i,1:n2-MigR)=P1RnowDummy(1:n2-MigR);
                    P4(i,1:n2-MigR)=P1RnowDummy(1:n2-MigR);
                    MigCnt(i)=MigCnt(i)-MigR*ScaleFactor;
                end
            end
        end
    end
end
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%END%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
    %       *****RESETTING EQUILIBRIUM PROFILE OF SHORELINE HAS BEEN REMOVED FOR THIS VERSION*****
    %        ***see Fix2MarineProcesses02202021 for most recent version of resetting profile***

    dH=round(dH,1); %need to round to 1 dec. place if redeclared shoreline
    Hstar(:,:,1)=(1/delta)*dH;  %convert back to slabs when redeclaring Hstar

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    if Tflag==1
        t=NaN;  %reset t if this is initialization MP so it doesn't throw of loop in MainCode
    end

end
```