



VCU

Virginia Commonwealth University
VCU Scholars Compass

Theses and Dissertations


Graduate School

2021

Modeling Vegetation Effects on Barrier Island Evolution

Eric W. Schoen
Virginia Commonwealth University

Follow this and additional works at: <https://scholarscompass.vcu.edu/etd>

 Part of the [Biology Commons](#), [Geomorphology Commons](#), [Natural Resources Management and Policy Commons](#), [Other Environmental Sciences Commons](#), [Other Life Sciences Commons](#), and the [Other Physical Sciences and Mathematics Commons](#)

© Eric Schoen

Downloaded from

<https://scholarscompass.vcu.edu/etd/6703>

This Thesis is brought to you for free and open access by the Graduate School at VCU Scholars Compass. It has been accepted for inclusion in Theses and Dissertations by an authorized administrator of VCU Scholars Compass. For more information, please contact libcompass@vcu.edu.

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

Copyright ©2021 by Eric Schoen
All rights reserved

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

Acknowledgements	iii
List of Figures	vii
List of Tables	viii
Abstract	ix
1 Introduction	1
2 The Model	5
2.1 Plant Populations	6
2.2 Aeolian Transport	10
2.3 Avalanche	12
2.4 Marine Processes	13
3 Results	17
3.1 Experiment Design	17
3.2 Parameterization	20
3.3 Results - Variations on Initial Plant Conditions and Sea Level Rise	26
3.3.1 No Sea Level Rise	27
3.3.2 Constant Sea Level Rise	28
3.3.3 Accelerating Sea Level Rise	29

4 Conclusion	32
Bibliography	35
A Notation Index	39
B MATLAB code	42
B.1 Main code	42
B.2 Aeolian Transport code	65
B.3 Avalanche code	82
B.4 Plant Propagation code	92
B.5 Shoreline code	103
B.6 Marine Processes code	106

List of Figures

2.1	Flow chart of the model pathway.	6
2.2	8 cell neighborhood	12
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)	15
3.1	Original, color-coded images of Virginia barrier islands	18
3.2	Elevation maps for Smith and Parramore Island	19
3.3	Illustration of island evolution spanning 27 years	20
3.4	Plots of the elevation map for Smith island before and after 27 years of evolution.	22
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.	22
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.	23

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.	24
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.	25
3.9	Evolution of Smith island in the absence of sea level rise, $M_\alpha = 0$, taken at varying percentages of initial and maximum plant cover.	27
3.10	Evolution of Smith island with constant sea level rise, $M_\alpha = 1$, taken at varying percentages of initial and maximum plant cover.	28
3.11	Evolution of Smith island with accelerating sea level rise, $M_\alpha = 1.0075$, taken at varying percentages of initial and maximum plant cover.	30

List of Tables

2.1	Elevation ranges of each plant species.	7
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.	10
2.3	Reduction values for ranges of scaled total percent cover	16
3.1	Select parameters optimized during this study, along with the migration rate reduction values given in Table 2.3.	21
A.1	Model constant and parameter notion and values, * values are for Smith, Parramore respectively.	40
A.2	Model arrays and variables	41

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 transportation,” of sediment. Werner’s model omits any effects of vegetation on 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 in 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 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 δ 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 $i = 1, \dots, n$ and $j = 1, \dots, m$. 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.

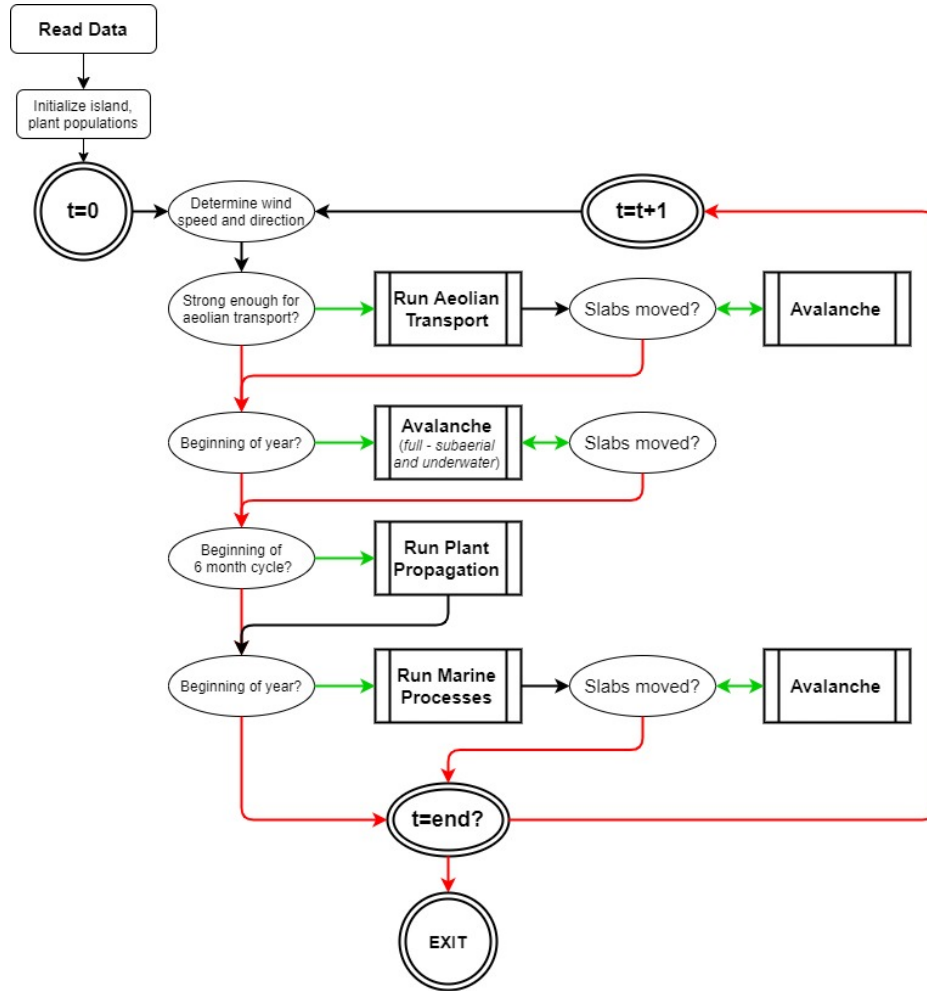


Figure 2.1: Flow chart of the model pathway.

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 $n = 1, 2, \dots, 8$, 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

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

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 $\sum_{n=1,2,\dots,8} \Omega_{k,n} \neq 0$. 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, η_k . We use these factors to modify the new population density, P'_k as given in equation (2.1) to

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

where we ensure growth does not exceed the maximum percent cover, η_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 β 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_∞ , 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 ($k = 1, 2, \text{ or } 4$),

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

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, ω , is sampled from the data set and the possible extent of a slab being transported is determined using Table 2.2.

Wind Speed (<i>m/s</i>)	Distance (<i>in cells</i>)
$\omega < 6$	0
$6 \leq \omega < 9$	1
$9 \leq \omega < 13$	2
$13 \leq \omega \leq 16$	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°) 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° [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

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

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, ρ_{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

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

where ω_L is the minimum wind speed required for aeolian transportation, and ω_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

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

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

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

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 in sea level rise, S , as

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

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 portion of the beach face. As noted by Davidson-Arnott [6], for shallow shore angles, θ_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 $H(i, j_s) \geq 0$.

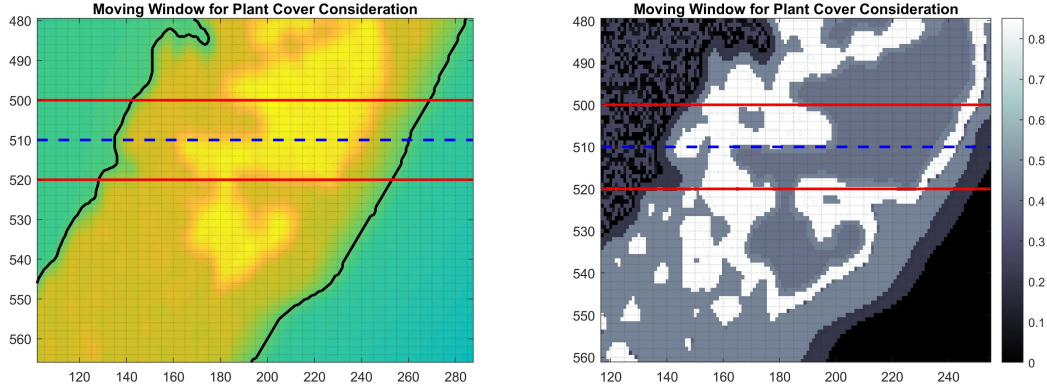


Figure 2.3: A moving window transect considering the impact of vegetation on the yearly migration of the shoreline for row i (given by the dashed line). Equation (2.9) takes into consideration all of the values of PC which fall within the red lines. The left image is the elevation map, while the right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

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

where ψ 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%	$0.5 \leq X_p$
70%	$0.35 \leq X_p < .5$
50%	$0.1 \leq X_p < .35$

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

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

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

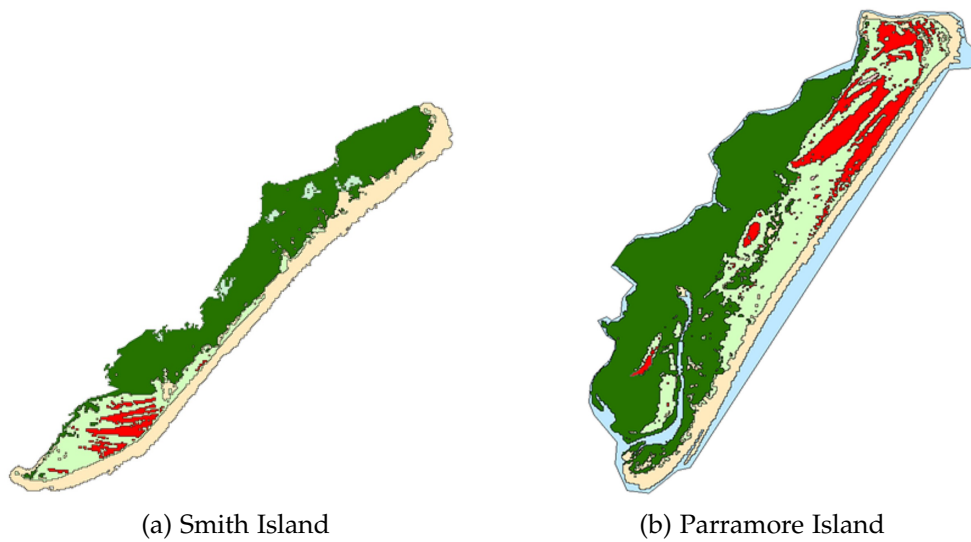
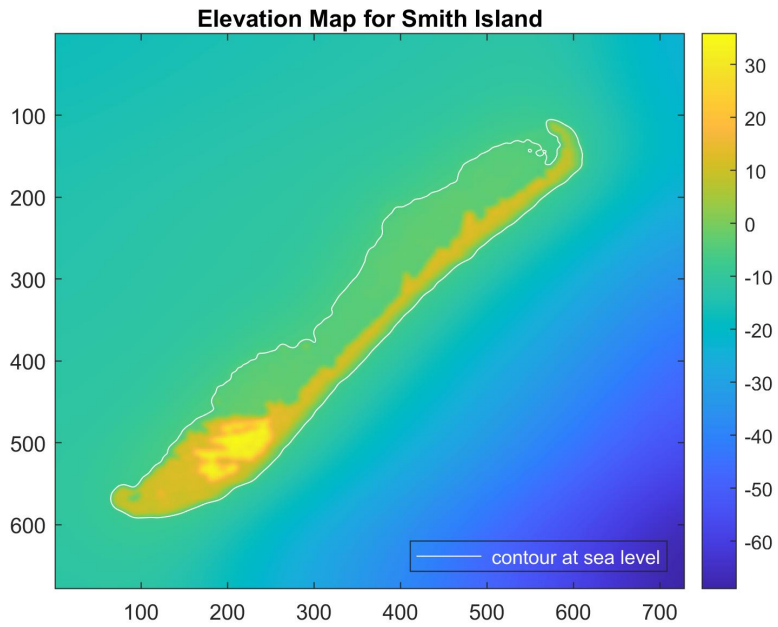


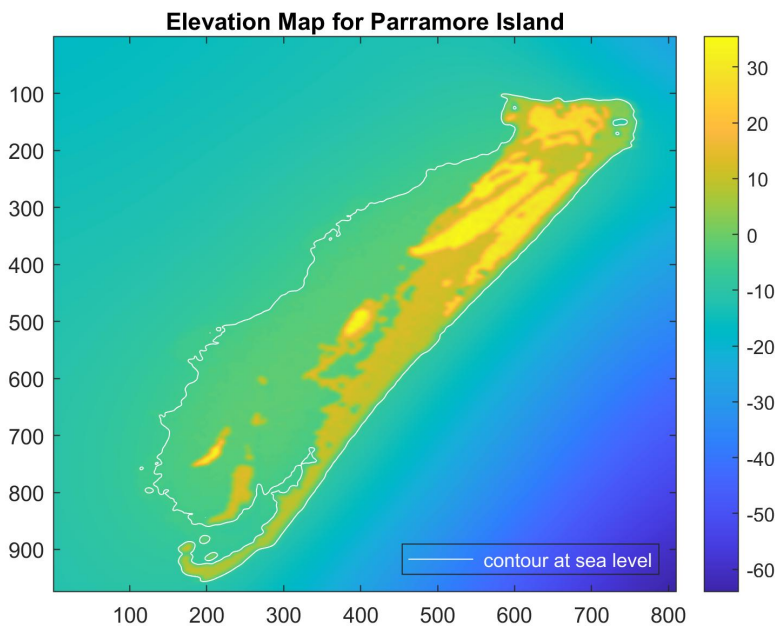
Figure 3.1: Original, color-coded images of Virginia barrier islands

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

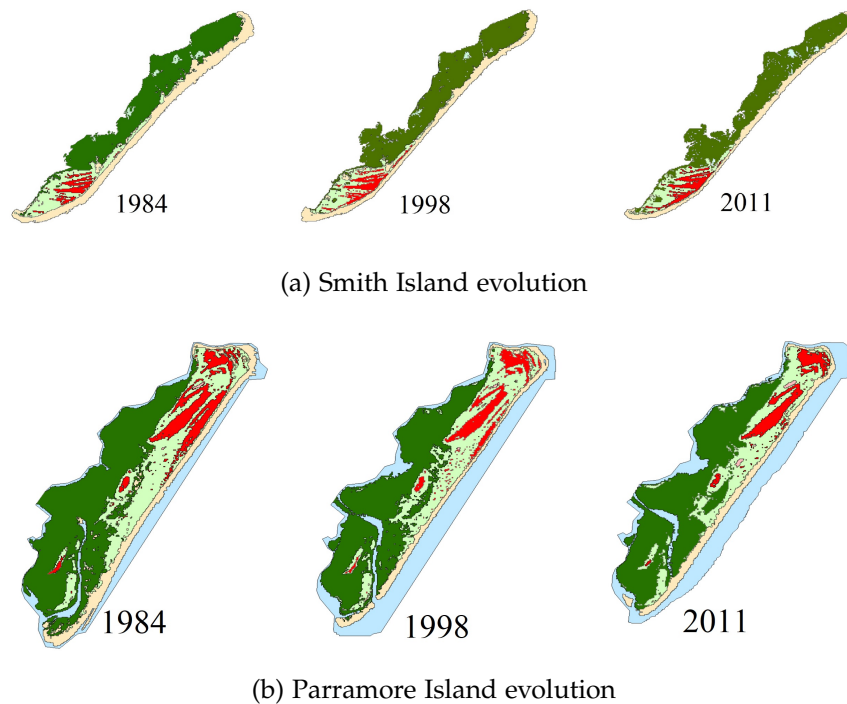


Figure 3.3: Illustration of island evolution spanning 27 years

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 α_1 is slightly larger than α_2 , reflecting the status of P_1 as the primary dune building grass. The coefficient α_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 γ_k along with the death by elevation 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
α_1	erosion coefficient for P_1	2/3
α_2	erosion coefficient for P_2	1/3
α_3	erosion coefficient for P_3	1
α_4	erosion coefficient for P_4	1
γ_k	growth range	[-0.02, 0.08]
β	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.

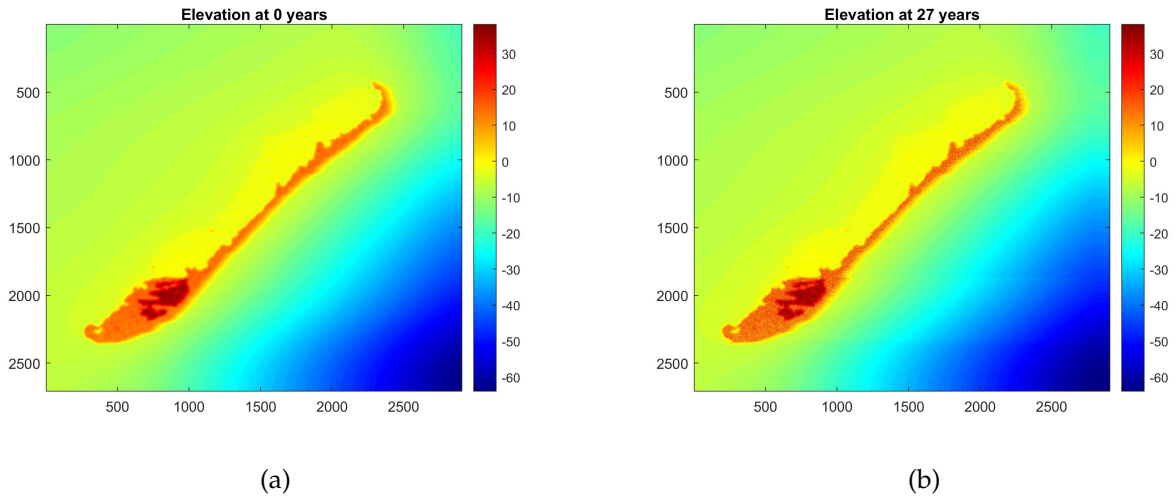


Figure 3.4: Plots of the elevation map for Smith island before and after 27 years of evolution.

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 aeolian transport and avalanching in more granular detail. We can also see, around the middle of both figures.

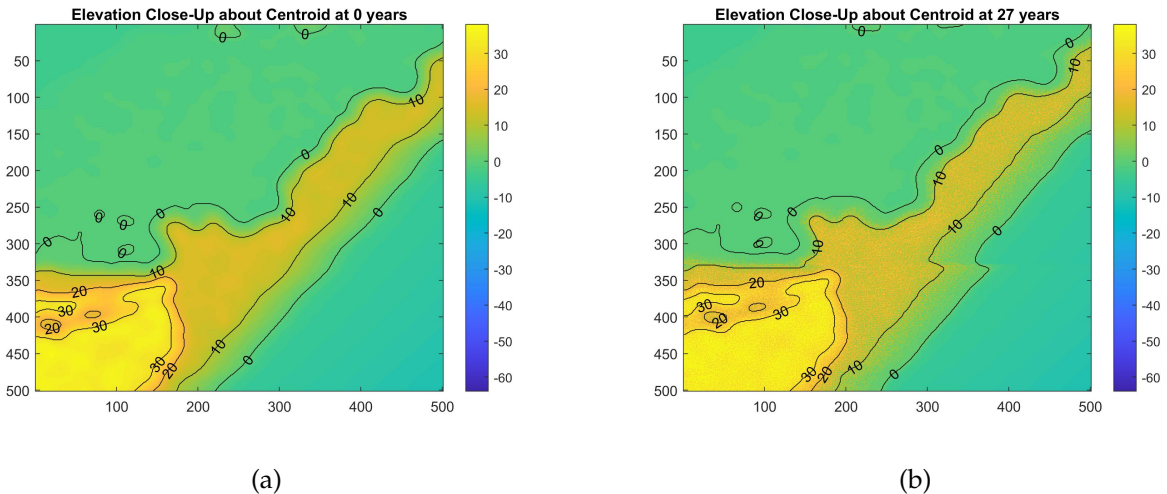
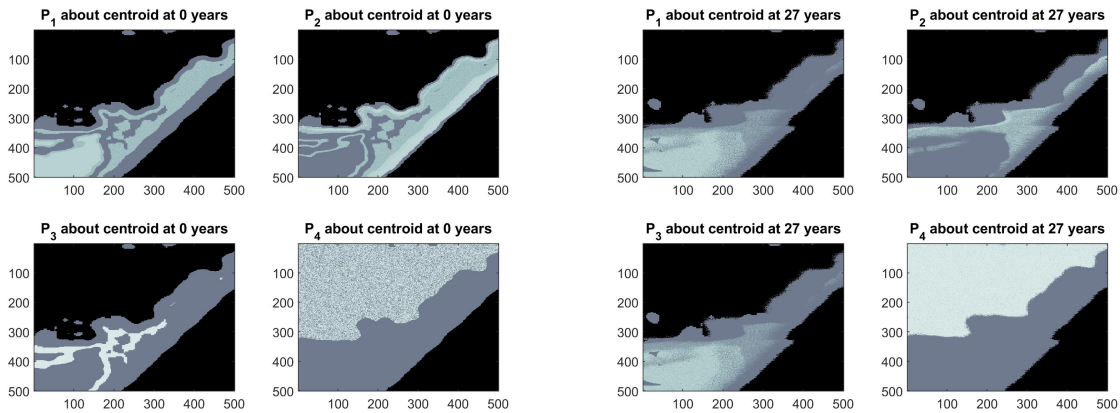


Figure 3.5: Close up of the elevation map with contours for areas around the centroid of Smith island before and after 27 years of evolution.

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.

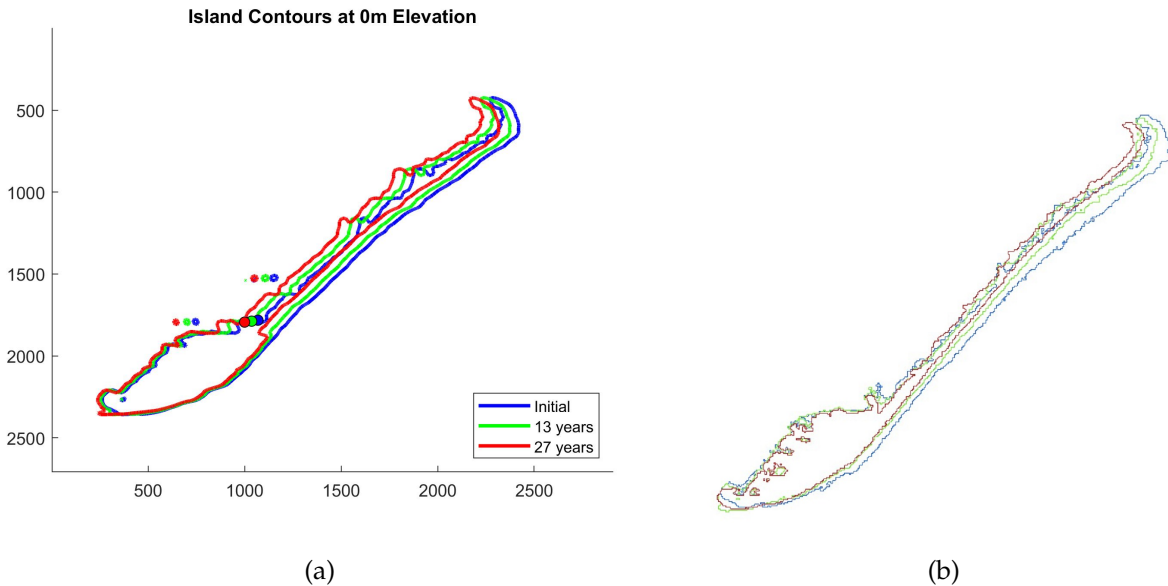


Figure 3.7: A progression of contours plots of elevation map taken at 0, 12, and 27 years compared to similarly timed contours of Smith island. The dots represent the centroid of the island for that year.

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.

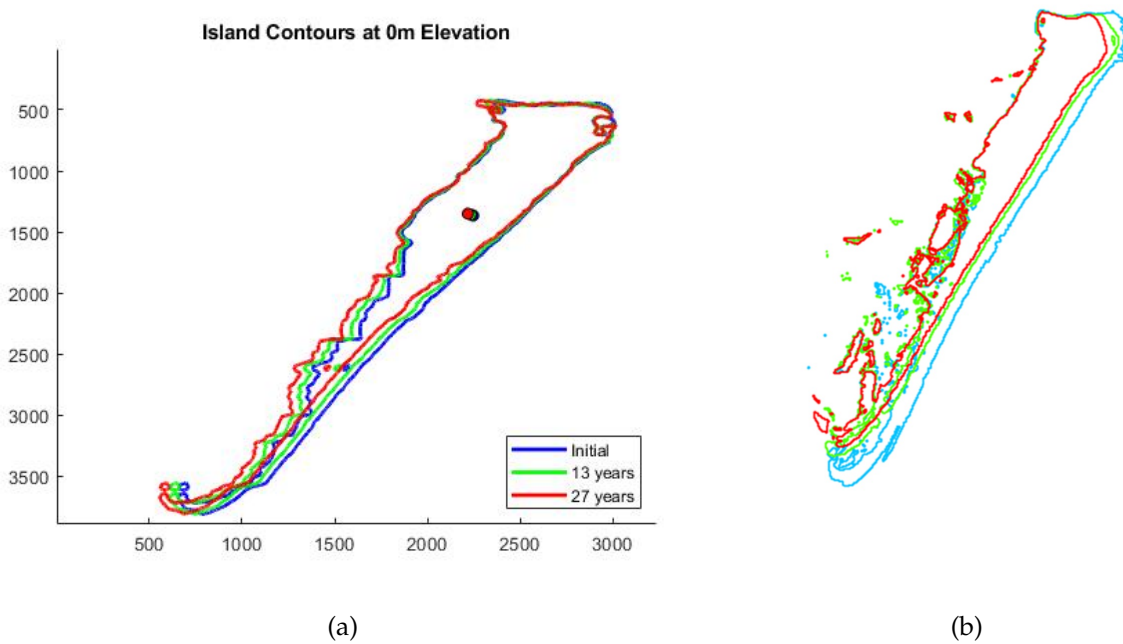


Figure 3.8: A progression of contour plots of elevation map taken at 0, 12, and 27 years compared to similarly timed contours of Parramore island. The dots represent the centroid of the island for that year.

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.

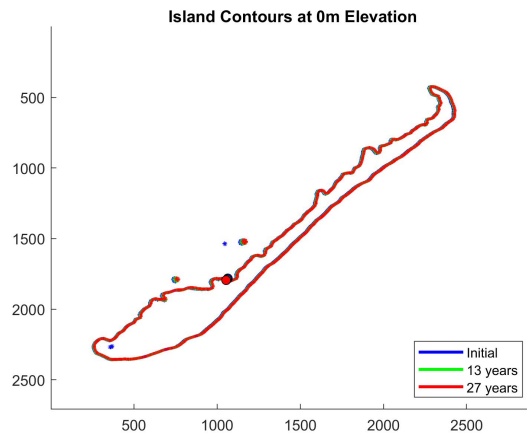
We note here that there are a few differences between the model behavior and the observed island behavior that can be pointed out in Figures 3.7 and 3.8. There is some lateral shrinking of the islands and additional regression in the southern part of Parramore Island, for instance. However, the overall 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

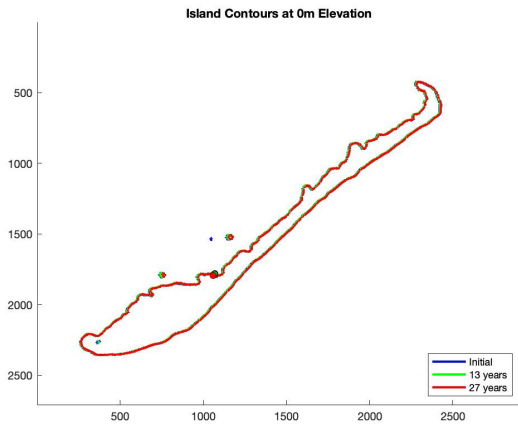
We simulate the island under varying conditions of sea level rise and plant percent coverage to demonstrate the dependence of barrier island geomorphology on sea level rise and the presence of vegetation. Sea level rise is taken to be either non existent, constant, or accelerating. Plant percent cover 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

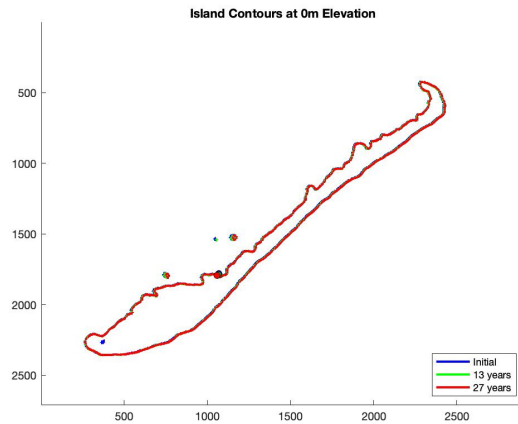
Island evolution in the absence of sea level rise is achieved by taking $M_\alpha = 0$. It follows that the island migration rate, R , would be reduced to zero by the rate acceleration function given in equation (2.10). The contours are given for Smith island.



(a) 0% plant cover



(b) 50% plant cover



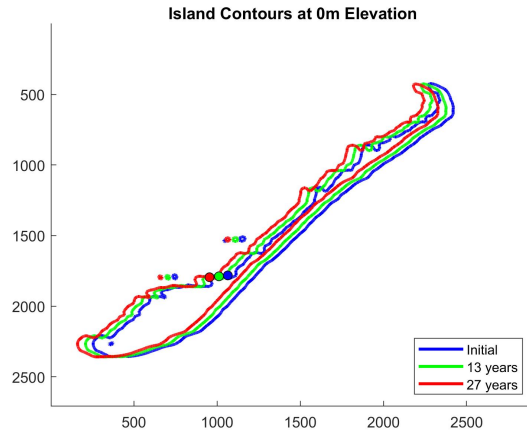
(c) 100% plant cover

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

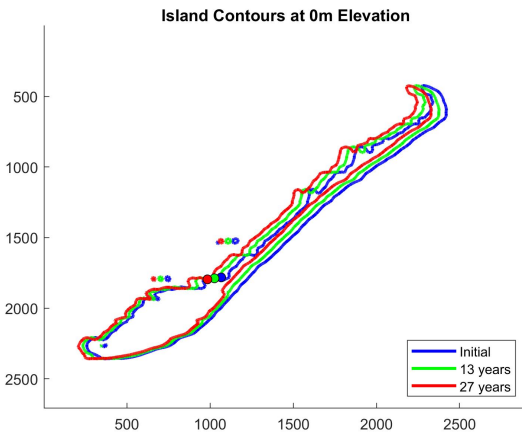
As expected, there is virtually no movement of the island. Shorelines vary in the slightest degree, which is evidence that the aeolian transport and avalanche processes are still being carried out. The same reasoning explains the slight change in centroid position.

3.3.2 Constant Sea Level Rise

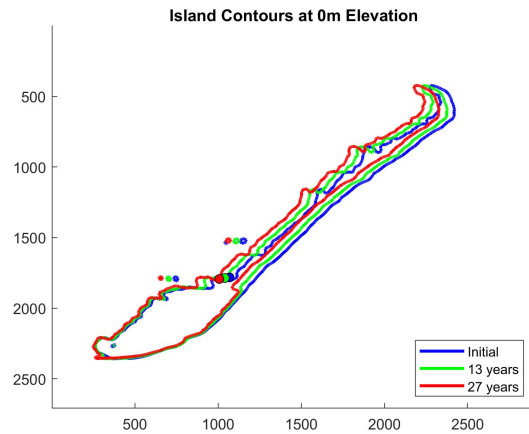
Constant sea level rise is simulated by taking the migration acceleration growth factor to be $M_a = 1$.



(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.

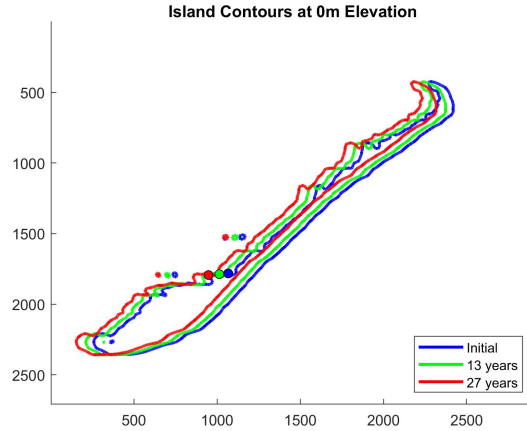
We can see in the Figure 3.10 that there is a notable change in the evolutionary behavior as the initial and maximum percent cover values are increased. The island observed in the absence of all plants, Figure 3.10a, can be seen to migrate uniformly. The curvature of the Eastern coastline remains generally 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 occurring 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.

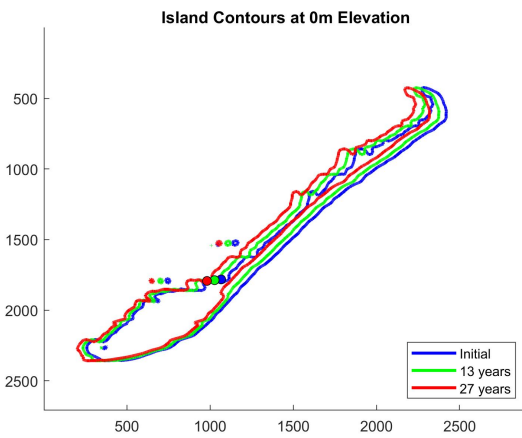
Note also the orientation of the centroids. The line connecting the centroid points becomes gradually more curved as the plant cover increases. We also see a small isthmus forming at the very southern tip of the island. When considered together, these observations demonstrate the island gradually curving, 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

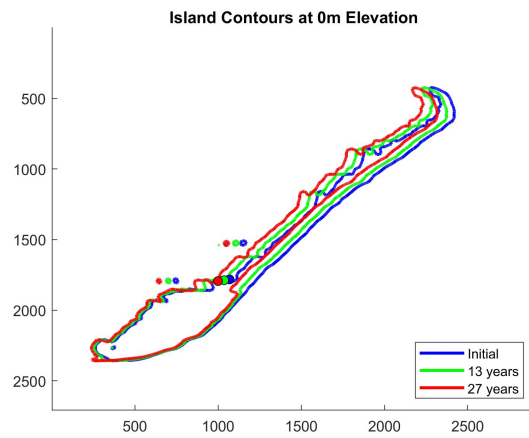
Accelerating sea level rise is achieved using a growth rate of 0.75%, as determined by the parameterization process. This gives us a growth factor $M_a = 1.0075$ which, when applied in equation (2.10), results in a gradual acceleration of landward migration. We would expect to see trends similar but more pronounced than those with constant sea 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 L , this means that the centroid of the island migrated by $29 * L$ meters in the first 13 years, and an additional $33L$ meters in the following 14 years, for a total of $62L$ 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 $33L$ meters in the first 13 years and 36.5 meters in the last 14 years, for a total of $69.5L$ meters in 27 years. The overall migration is then $7.5L$ 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 response 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 *
δ	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
β	death by elevation percentage	30%
γ_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
η_k	plant percent cover maximum for each P_k , $k = 1, 2, 3, 4$	80%, 80%, 60%, 60%
M_∞	global percent cover maximum (for all P_k on a given cell)	80%
ω_L	minimum wind speed required for sediment transport	6 meters/second
ω_H	threshold for storm event	16 meters/second
α_k	erosion coefficient for each P_k , $k = 1, 2, 3, 4$	0.667, 0.333, 1, 1
θ_o	angle of repose for avalanche	$\pi/6$
R_o	initial rate of shoreline retreat	15 m/yr
θ_o	angle of repose for avalanche	$\pi/6$
$2w$	width of transectional window to calculate X_p	10 cells
M_α	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, α_k
α	percent cover on cell in excess of M_∞
l	number of grass species on cell ($l \in \{1, 2, 3\}$)
θ'	current cell angle of elevation with respect to a neighboring cell
ρ_{av}	probability of avalanching
ρ_{ea}	probability of aeolian transport at step d
d	aeolian transport distance in cell widths ($d \in \{1, 2, 3\}$)
ω	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)
θ_b	approximate angle of foreshore slope (Bruun Rule)
j_s	horizontal location of the shoreline for row i
X_p	migration rate reduction factor
Ψ	area of subaerial portion of island within transectional window

Table A.2: Model arrays and variables

Appendix B

MATLAB code

B.1 Main code

```
1 %% MainCode092820
2 % MPswitch=1;
3 % ATcntr=0;
4 % SLRswitch=0;
5 % is the updated main code for barrier island evolution.
6 % this version of the code is build upon the original code by Greg Robson
7
8 % off-hand key notes (NOT EVEN REMOTELY COMPREHENSIVE)
9 % **ARRAYS
10 %     H - main elevation matrix
11 %         (only updated in initialization and at end of the main loop)
12 %     Hstar - dummy elevation array for changes in subroutines
13 %         *third dimension unused - was for land categorization
14 %     P_i - Plant perent coverage matrices,
15 %     P3d - tracks dead morella
16 %     ~Pi(:, :, 1) current cell percent cover
17 %     ~Pi(:, :, 2) initial elevation of current cell
18 %         (currently unused, will be needed to check for burial)
19 %     PC - Combine Pi's for a total percent cover array - (no cell >1)
20 %         (Pi stores value -999 in cells without that plant - i.e.
21 %         water, so PC is the array with only cells greater than 0
22 %     Ptot= Sun of all PCi
23 %     W - water table data (currently unused)
24 %     S - salinity (?) data (currently unused)
25 % **General PARAMETERS
26 %     time = number of iterations (two week time steps)
27 %     delta - slab heigh (meters)
28 %     L - slab length and width (meters)
29
30 % **Routines and Governing Parameters
31 %     Main Code
32 %     t - current time step
```

```

33 % (2) AeolianTransport
34 % Pe - prob. of erosion
35 % Pd - prob. deposition
36 % n - number of slabs which can move due to wind
37 % (3) Plant Propogation
38 % currently runs twice per year
39 % first time ignores morella and is meant to simulate springtime
40 % growth for grasses (P1,P2,P4 - allow them more chance to grow)
41 % PiIC - Plant initial percent cover for P1,P2,P3,P4
42 % PiErosionCoefficient - factors into AT
43 % P-Burial - number of slabs until death
44
45 % (4) Marine Processes
46 % currently runs once every 3 months
47 %
48 % avalanche:
49 % whole domain (underwater and island) ever 5th year time step
50 % once per year runs THREE times per step over only subaerial (ground)
51 % after SwampProcess, AeolianTransport, and before loop end
52 % otherwise runs once per time step, before loop end
53 %
54 % This version contains NO use of Land Categorization
55 tic
56
57 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
58 %% %%%%%%%%%LOAD ELEVATION MATRIX%%%%%%%%%
59 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
60 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
61 % PARRAMORE ISLAND DATA:%
62 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
63 % IslandArea=14400000;
64 % 12km x 1200 m = 14400000 - Relative Influence Antecedent... paper, 12.4 m/yr
65 filename='Parramore03312021.mat'
66 IslandArea=19070000; %for parramore
67 data=importdata(filename);
68 H=data; %H will be the main elevation matrix
69 TranChk=1;
70
71
72
73 % filename='EricIsland.txt'
74 % IslandArea=41000;%EricIsland
75
76 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
77 % SMITH ISLAND DATA:%
78 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
79 % filename='Smith03312021.mat'
80 % IslandArea=9065000; %for Smith
81 % data=importdata(filename);
82 % H=data; %H will be the main elevation matrix
83 % TranChk=2;
84
85 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
86
87
88 ScaleFactor=floor(sqrt(IslandArea/(sum(sum(H(:, :)>0)))));

```

```

89 % InterpFactor=.75*sqrt(ScaleFactor); %ScaleFactor 2
90 InterpFactor=0.5*sqrt(ScaleFactor); %ScaleFactor 4
91
92 H=interp2(H,InterpFactor);
93 ScaleFactor=floor(sqrt(IslandArea/(sum(sum(H(:, :)>0)))));
94 if TranChk==1
95     TranRow=floor(.25*size(H,1)); %choose row for transect for parramore
96 elseif TranChk==2
97     TranRow=floor(0.75*size(H,1)); %choose row for transect for smith
98 elseif exists(TranChk)==0
99     TranRow=floor(0.5*size(H,1));
100 end
101 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
102 %%%%%%%%%END - LOAD ELEVATION MATRIX%%%%%%%%
103 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
104
105 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
106 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
107 %%                %%%%%%%%% PARAMETERS                %%%%%%%%%                %
108 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%                %
109 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%                %
110 fprintf('\n')
111 fprintf('Initializing...')
112     %cheat sheet:
113 time=702; %2600~100 yrs;%1300~50 yrs;%780~30 yrs;%520~20 yrs;260~10 yrs;
114 delta=0.1;
115 % L=1;%<-----ScaleFactor
116 L=ScaleFactor;
117 BchMax =1.5; %greatest elevation that the beach spreads inland to
118 OW=0; %switch for overwashing - might use when we get storms incorporated (if windspeed>16m/s OW=1 for on, etc)
119 BchW=10; %unused - maximum beach width
120 MPswitch=1; %changes marine processes: --(0 or any ~=1,2,3) for OFF
121 % --(1) for ON {new version - update using same equil. slope}
122 % --(2)for ON {old version - update equil. slope every 12 wks}
123     MarineProcesses03312021
124     OLDMarineProcesses03312021
125 DepositSupply=3; %number of slabs of sediment supplied to beach each time MarinePRocesses is run (3 months)
126 SLRswitch=0; %turns sea level rise ON(1)/OFF(0 or any ~=1)
127 SLRYrs=[ 8 7 6 5]; %years at which to perform SLR at rate SLR and/or migration...
128 Ma= 1;%1.0075; %shoreline migration growth factor, use 1 for no accel/noSLR
129 MigAccel=0; %will need to track magnitude of migration change
130 MigYr=15; %initial yearly migration of the island in meters/year
131 ScaleFactor=floor(sqrt(IslandArea/(sum(sum(H(:, :), 1)>0)))));
132 % R=zeros(size(H,1),1);
133 MaxSwampWidth=1000; %how far the swamp should go out into the water - just have to make something up for now
134
135 WindData=1; %0~steady windspeed/direction ,1~windspeed and direction from empirical data
136
137 if WindData==0
138     Windspeed=16;
139     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
140     WindDir=1; %wind direction % 1=N 2=NE 3=E 4=SE 5=S 6=SW 7=W 8=NW %
141     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
142 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```

```

140 end
141 Windmin=6;
142 StormThreshold=16;
143 ATcntr=0; %testing variable - to count number of times AT is called
144
145 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% PLANT PARAMETERS %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
146 %Plant initial conditions:
147 PlantRangeArray=[1 5;0.75 3;1.5 2.5;-0.5 0]; %all of the elevation ranges for p1-p4
148 P1IC=1.0;
149 P2IC=1.0;
150 P3IC=1.0;
151 P4IC=1.0;
152 P1PetMax=.6; %largest percentage we will allow any plant population on a given cell to attain
153 P2PetMax=.6;
154 P3PetMax=.8;
155 P4PetMax=.8;
156 PetMax=[P1PetMax P2PetMax P3PetMax P4PetMax];
157 MasterMax=1.0;% The most any cell can permit - 80% plant coverage
158 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
159 alpha=.01; %propagation rate for each populated cell
160 DBE=.3; %death by elevation rate for each populated cell outside of plant's elevation range
161 gdrange1=[-.02:.01:.08]; %range of percent values for growth/death for plant populations at (0, 50)% cover
162 gdrange2=[-.02:.01:.08];%[-.04:.01:.04]; %range of percent values for growth/death for plant pops greater than 50% cover
163
164 %Elevation Matrix dependent parameters:
165 ROW=size(H,1);
166 COLUMN=size(H,2);
167 H(:, : ,2)=zeros(ROW,COLUMN); %Creating extra dimension for H
168 Hstar=H; %initializing dummy matrix
169 MigCnt=zeros(ROW,1); %used to store how many meters the shoreline should have receded by, shoreline moves when MigCnt>
    ScalingFactor
170 M=max(max(H(:, : ,1)));
171 m=min(min(H(:, : ,1)));
172 W=zeros(ROW,COLUMN); %do not comment out - needed as input
173 S=zeros(ROW,COLUMN); %do not comment out - needed as input
174
175 %parameters unused so far (check with Greg)
176 % numslabg=0;
177 % q=0;
178 % Salt=1;
179 % ps=0.6;
180 AV_ARRAY=zeros(time,1000,3); %tracks cell movement in AV. routine
181 %%
182 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
183 %%% Imaging Options %%%
184
185 % ***note that we do not currently have a running land categorization
186 % subroutine, so leave those options commented or at 0 - I think this
187 % will eventually come back into use so I am leaving the lines there***
188
189 DurImFreq=78; %how often to output "during" elevation images (26~yearly, 130~every 5 years)
190 MElevCont=0;
191 SElevCont=0;
192 PlantDurImFreq=234; %same, just for plant images
193 DurCUImFreq=78;
194

```

```

195 ElevImCU.during=1;
196 Contour27=1;      %contours at 0, 13, and 27 years
197 DurContour=0;    %images of the island outline
198
199 TransectIm=1;     %images of 2D island transect
200   TranImFreq=234; %every 9 years
201   if TransectIm==1 && exist('TranRow') == 0
202       TranRow=floor(.5*size(H,1)); %choose row for transect
203   end
204
205 MnPlantCvrIm=1;   %mean percent cover for each plant, displayed at end of routine 1 to turn on, any other number to turn off
206 ElevIm_before=1;   ElevIm_during=1;   ElevIm_after=0;
207 LandCatIm_before=0; LandCatIm_during=0; LandCatIm_after=0;      %leave these at 0 until we have a working
    LandCategorization routine
208 PlantPCIm_before=0; PlantPCIm_during=0; PlantPCIm_after=0;
209 SingleStepICIm=0; %plots from initialization process :
210 %includes elevation and Plant initial conditions for
211 %one pass through Swamp, PlantPropagation and
212 %AeolionTransport
213 climH=[min(min(H(:, :), 1)) max(max(H(:, :), 1))];
214 climP=[-1 1];
215 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
216 %%           %%%%%%%%%%Initializing Plant Arrays%%%%%%%%%
217 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
218 %GRASS #1 (Ammophila)(burial resistant)
219 P1=zeros(ROW,COLUMN,2);
220 P1Burial=2;        %meters of burial until death
221 P1ErosionCoefficient=(2/3); %used in formula d=c1P1+c2P2+c3P3+c4P4
222
223 %GRASS #2 (Spartina)
224 P2=zeros(ROW,COLUMN,2);
225 P2Burial=0.5;     %meters of burial until death
226 P2ErosionCoefficient=(1/3);
227
228 %SHRUB #1 (Morella)(bird-dispersed seeds)
229 P3=zeros(ROW,COLUMN,2);
230 P3Burial=1.5;     %meters of burial until death
231 P3ErosionCoefficient=1;
232
233 %dead morella will track when morella reaches death and store dead
234 %debris data in P3d(:, : ,1) for some number of years which will be counted in
235 %P3d(:, : ,2) - this is only updated inside of PlantProp
236 P3d=zeros(ROW,COLUMN,2);
237 P3dErosionCoefficient=1;
238
239 %SECOND TYPE OF SPARTINA!!!
240 P4=zeros(ROW,COLUMN,2);
241 P4Burial=0.5;
242 P4ErosionCoefficient=(1/3);
243
244 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
245 % INITIAL PLANT CONDITIONS %
246 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
247 % % % Each plant species has a preferred "altitude" that it grows best in
248 % % % (min and max among all species is 0.75m and 5m, respectively).
249 % % % We set each cell of "appropriate" elevation to be covered by some% of the

```



```

250 %%% respective plant species. For cells which are submerged under water, we
251 %%% set the proportion of plant coverage to -1.
252 %%%
253 %%% We have deleted the initialization of plant population since we are using
254 %%% real data.
255 %%% It is still important to ensure that the 2nd layer of each
256 %%% plant matrix is initiated.
257 fprintf('\n')
258 fprintf('Seeding the island...')
259 PC1=zeros(size(H,1),size(H,2));
260 PC2=zeros(size(H,1),size(H,2));
261 PC3=zeros(size(H,1),size(H,2));
262 PC4=zeros(size(H,1),size(H,2));
263 [Hstar,MeanBeachWidth,ESL,WSL,MESL,MWSL,OL]=Shoreline03312021(Hstar,delta,L,BchMax);%function to return shoreline(s)
264 for i=1:size(H,1)
265     for j=1:size(H,2)
266         if (delta*H(i,j,1))>=1 && (delta*H(i,j,1))<=5
267             P1(i,j,1)=P1IC;
268             P1(i,j,2)=H(i,j,1);
269         elseif (H(i,j,1)-W(i,j))<0
270             P1(i,j,1)=-999;
271             P1(i,j,2)=0;
272         end
273         PC1(i,j)=P1(i,j,1)*(P1(i,j,1)>0);
274         if (delta*H(i,j,1))>=0.25 && (delta*H(i,j,1))<=3
275             P2(i,j,1)=P2IC;
276             P2(i,j,2)=H(i,j,1);
277         elseif (H(i,j,1)-W(i,j))<0
278             P2(i,j,1)=-999;
279             P2(i,j,2)=0;%
280         end
281         PC2(i,j)=P2(i,j,1)*(P2(i,j,1)>0);
282         if (delta*H(i,j,1))>=1.5 && (delta*H(i,j,1))<=2.5
283             P3(i,j,1)=P3IC;
284             P3(i,j,2)=H(i,j,1);
285         elseif (H(i,j,1)-W(i,j))<0
286             P3(i,j,1)=-999;
287             P3(i,j,2)=0;
288         end
289         PC3(i,j)=P3(i,j,1)*(P3(i,j,1)>0);
290         if (delta*H(i,j,1))>=-0.5 && (delta*H(i,j,1))<=1
291             if ESL(i)~=0
292                 if MWSL(i)<j && MESL(i)>=j
293                     if rand<0.5
294                         P4(i,j,1)=P4IC;
295                         P4(i,j,2)=H(i,j,1);
296                     end
297                 else
298                     P4(i,j,1)=0;
299                     P4(i,j,2)=0;
300                 end
301             end
302             WesternCells=zeros(1,MaxSwampWidth);
303             EasternCells=zeros(1,MaxSwampWidth);
304             for mm=1:min(j-1,size(WesternCells,2))
305                 WesternCells(mm)=delta*H(i,j-mm,1);

```

```

306 %         end
307 %         for mm=1:min(size(EasternCells,2),size(H,2)-j)
308 %             EasternCells(mm)=delta*H(i,j+mm,1);
309 %         end
310 %         CheckWesternCells=WesternCells<=-.5;%must be within MaxSwampWidth cells to the west of a cell which is outside of
marsh range
311 %         CheckEasternCells=EasternCells>0;
312 %         if sum(CheckWesternCells)==0 || sum(CheckEasternCells)==0
313 %             P4(i,j,1)=-999;
314 %             P4(i,j,2)=0;
315 %         end%
316 %         if sum(CheckWesternCells)~=0 &&5<rand
317 %             P4(i,j,1)=P4IC;
318 %             P4(i,j,2)=H(i,j,1);%
319 %         end
320 %         elseif (delta*H(i,j,1)-W(i,j))<-.5 || (delta*H(i,j,1)-W(i,j))>1
321 %             if (delta*H(i,j,1)<PlantRangeArray(4,1)) %if less than min height (-0.5)
322 %                 P4(i,j,1)=-999;
323 %                 P4(i,j,2)=0;
324 %             elseif (delta*H(i,j,1)>PlantRangeArray(4,2)) %elseif greater than max height, make 0 (no death by elev.
just kill)
325 %                 P4(i,j,1)=0;
326 %                 P4(i,j,2)=0;
327 %             end
328 %         end
329 %         PC4(i,j)=P4(i,j,1)*(P4(i,j,1)>0);
330 %     end
331 % end
332 % t=NaN;
333 % [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);
334 % for i=1:size(H,1)
335 %     for j=1:size(H,2)
336 %         PC1(i,j)=P1(i,j,1)*(P1(i,j,1)>0);
337 %         PC2(i,j)=P2(i,j,1)*(P2(i,j,1)>0); %
338 %         PC3(i,j)=P3(i,j,1)*(P3(i,j,1)>0);
339 %         PC4(i,j)=P4(i,j,1)*(P4(i,j,1)>0);
340 %     end
341 % end
342 % PC=(P1ErosionCoefficient.*PC1(:,:))+(P2ErosionCoefficient.*PC2(:,:))+(P3ErosionCoefficient.*PC3(:,:))+(P4ErosionCoefficient.*PC4
(:,:))+(P3dErosionCoefficient.*P3d(:,:1));
343 % Ptot=P1(:,:1)+P2(:,:1)+P3(:,:1)+P4(:,:1)+P3d(:,:1);
344 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
345 %%%%%%%%%END - INITIALIZING PLANT ARRAYS%%%%%%%%
346 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
347
348 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
349 %%%killing plants on half of island test%%
350 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
351 % if KillSwitch==1;
352 %     for i=floor(0.5*(size(H,1))):size(H,1) %killing bottom half plants
353 %         for j=1:size(H,2)
354 %             if P1(i,j,1)>0
355 %                 P1(i,j,1)=0;
356 %                 PC1(i,j)=0;
357 %             end

```

```

358         if P2(i,j,1)>0
359             P2(i,j,1)=0;
360             PC2(i,j)=0;
361         end
362         if P3(i,j,1)>0
363             P3(i,j,1)=0;
364             PC3(i,j)=0;
365         end
366         if P4(i,j,1)>0
367             P4(i,j,1)=0;
368             PC4(i,j)=0;
369         end
370     end
371 end
372 PC=(P1ErosionCoefficient.*PC1(:,:))+(P2ErosionCoefficient.*PC2(:,:))+(P3ErosionCoefficient.*PC3(:,:))+(P4ErosionCoefficient.*
PC4(:,:))+(P3dErosionCoefficient.*P3d(:,: ,1));
373 end
374 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
375 %%%END of half island plant test conditions%%
376 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
377
378 %%          %%%%%%%%%BEFORE IMAGES%%%%%%%%
379 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
380
381 if ElevIm_before==1
382     Hfilt=round(imgaussfilt(H(:,: ,1) ,2) ,0);
383     %%%%%%%%%Elevation%%%%%%%%
384     figure %
385     colormap(jet()) %
386     imagesc(H(:,: ,1) ,climsH) %
387     hold on
388     if MElevCont==1
389         contour(H(:,: ,1) ,[-5 -5] , 'color' , 'm' , 'linewidth' ,1.1)
390     end
391     if SElevCont==1
392         contour(Hfilt(:,: ,1) ,[-0 -0] , 'color' , 'k' , 'linewidth' ,1.1)
393     end
394     if MElevCont==1 && SElevCont==1
395         lgnd=legend('\color{white} marsh contour (-0.5m)' , '\color{white} sea level contour (0m)');
396         set(lgnd , 'color' , 'none' , 'location' , 'southeast');
397     elseif MElevCont==1 && SElevCont==0
398         lgnd=legend('\color{white} marsh contour (-0.5m)');
399         set(lgnd , 'color' , 'none' , 'location' , 'southeast');
400     elseif SElevCont==1 && MElevCont==0
401         lgnd=legend('\color{white} sea level contour (0m)');
402         set(lgnd , 'color' , 'none' , 'location' , 'southeast');
403     end
404     colorbar %
405     title('Before Image'); %
406     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
407 end
408
409 if PlantPCIm_before==1
410     %%%%%%%%%PLANTS%%%%%%%%
411     %%% AMMOPHILA
412     figure

```

```

413     colormap gray
414     imagesc(P1(:,:,1),climsP)
415     colorbar
416     title('P1 - Ammophila IC')
417     %%% SPARTINA
418     figure
419     colormap gray
420     imagesc(P2(:,:,1),climsP)
421     colorbar
422     title('P2 - Spartina patens IC')
423     %%% MORELLA
424     figure
425     colormap gray
426     imagesc(P3(:,:,1),climsP)
427     colorbar
428     title('P3 - Morella IC')
429     %%% SPARTINA (marsh)
430     figure
431     colormap gray
432     imagesc(P4(:,:,1),climsP)
433     colorbar
434     title('P4 - Spartina alterniflora IC')
435 end
436 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
437 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%Single Step Subroutine Images%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
438 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
439 fprintf('\n')
440 fprintf('Running startup sequences:')
441 fprintf('\n')
442 fprintf('Running Swamp Processes...')
443 %Initial - Run Swamp
444     [Hstar,ColumnArraySwamp1,ColumnArraySwamp2,AdjascentLengthSwamp,OppositeLocationSwamp,flag,INum]=SwampProcesses03312021(Hstar,
         delta,L,flag,PC);
445 fprintf('done')
446 fprintf('\n')
447 fprintf('Running Avalanche...')
448 % (1.5) post swamp avalanche
449 PassCount=1;
450 CellsMoved=zeros(1,1000);
451 [Hstar,flag,CellCt]=AVALANCHE03312021(Hstar,delta,L,flag,PC);
452 CellsMoved(PassCount)=CellCt;
453 while flag==1
454     PassCount=PassCount+1;
455     [Hstar,flag,CellCt]=AVALANCHE03312021(Hstar,delta,L,flag,PC);
456     CellsMoved(PassCount)=CellCt;%
457 end
458 CellsMoved=CellsMoved(1,1:PassCount);
459 SwampAvArray=[NaN PassCount CellsMoved]; %tracking number of passes and number of cells moved per pass
460 AvIC_Array1=SwampAvArray; %AvIC_Array1 tracks avalanche data for initialization routines
461 H=Hstar; %update H
462 fprintf('done')
463
464
465 %***** NOT USING LAND CATEGORIZATION - BUT MAY AGAIN LATER - DO NOT DELETE!!
466 for i=1:size(Hstar,1)
467     for Icount=1:INum

```

```

468         if ColumnArraySwamp2(i,Icount)~=0
469             for j=ColumnArraySwamp1(i,Icount):ColumnArraySwamp2(i,Icount)
470                 Hstar(i,j,2)=2;
471             end
472         end
473     end
474 end
475 %***** NOT USING LAND CATEGORIZATION - BUT MAY AGAIN LATER - DO NOT DELETE!!
476
477 %post swamp processes/avalanche1 images
478 if SingleStepICIm==1
479     figure('NumberTitle','off','Name','After Swamp');
480     colormap(jet()) %
481     %   clim=[min(min(Hstar(:,1))) max(max(Hstar(:,1)))]; %
482     imagesc(Hstar(:,1),climsH)%
483     title('ICs After Swamp')
484     colorbar
485 end
486 fprintf('\n')
487 fprintf('Running Marine Processes ... ')
488 %Initial - Marine Processes
489 MeanBeachWidthVec=zeros(1,time+2); %for tracking beach width evolution
490 if MPswitch==1
491     % need version of PC with erosion coefficients AND with negative values (zeros only where plants COULD grow) to calculate
492     % migration
493     PCmp=-999*ones(ROW,COLUMN);
494     for i=1:ROW
495         for j=1:COLUMN
496             if P1(i,j,1)>=0 || P2(i,j,1)>=0 || P3(i,j,1)>=0 || P4(i,j,1)>=0
497                 PCmp(i,j)=(P1ErosionCoefficient.*P1(i,j,1).*(P1(i,j,1)>0))+(P2ErosionCoefficient.*P2(i,j,1).*(P2(i,j,1)>0))+(
498                     P3ErosionCoefficient.*P3(i,j,1).*(P3(i,j,1)>0));%+(P4ErosionCoefficient.*P4(i,j,1).*(P4(i,j,1)>0));
499             end
500         end
501     end
502     Pt1=max(P1(:,1),0);
503     Pt2=max(P2(:,1),0);
504     Pt3=max(P3(:,1),0);
505     Pt4=max(P4(:,1),0);
506     PCmp=(P1ErosionCoefficient.*Pt1)+(P2ErosionCoefficient.*Pt2)+(P3ErosionCoefficient.*Pt3)+(P4ErosionCoefficient.*Pt4);
507     [Hstar,P3,MeanBeachWidth,ESL,WSL,MESL,MWSL,OppositeLocation,SLRyrs,MigCnt,MigAccel]=MarineProcesses03312021(Hstar,delta,L,P1,
508         P2,P3,P4,BchMax,OW,t,SLRyrs,PCmp,IslandArea,MigCnt,ScaleFactor,MigYr,Ma,MigAccel,MasterMax);
509     MeanBeachWidthVec(1)=MeanBeachWidth;
510     H=Hstar;
511 elseif MPswitch==2
512     [Hstar,PlantColumnArray,PlantColumnArray2,P3,flag,MeanBeachWidth]=OLDMarineProcesses03312021(Hstar,delta,L,flag,PC,P3,BchMax,
513         BchW);
514     MeanBeachWidthVec(1)=MeanBeachWidth;
515     H=Hstar;
516 end
517 fprintf('done')
518
519 if SingleStepICIm==1
520     figure('NumberTitle','off','Name','After Marine Processes');
521     colormap(jet()) %
522     imagesc(Hstar(:,1),climsH)%
523     title('ICs after Swamp, Marine Processes')

```

```

520     colorbar
521 end
522
523 %Redeclare PC since P3 has been updated
524 %PC=(P1ErosionCoefficient.*P1(:, :, 1))+(P2ErosionCoefficient.*P2(:, :, 1))+(P3ErosionCoefficient.*P3(:, :, 1))+(P4ErosionCoefficient.*
        P4(:, :, 1)); %
525 for i=1:size(H,1)
526     for j=1:size(H,2)
527         PC1(i, j)=P1(i, j, 1)*(P1(i, j, 1)>0);
528         PC2(i, j)=P2(i, j, 1)*(P2(i, j, 1)>0); %
529         PC3(i, j)=P3(i, j, 1)*(P3(i, j, 1)>0);
530         PC4(i, j)=P4(i, j, 1)*(P4(i, j, 1)>0);
531     end
532 end
533 PC=(P1ErosionCoefficient.*PC1(:, :, :))+ (P2ErosionCoefficient.*PC2(:, :, :))+ (P3ErosionCoefficient.*PC3(:, :, :))+ (P4ErosionCoefficient.*PC4
        (:, :, :))+ (P3dErosionCoefficient.*P3d(:, :, 1));
534
535 %post marine processes avalanche
536 PassCount=1;
537 CellsMoved=zeros(1,1000);
538 [Hstar, flag, CellCt]=AVALANCHE03312021(Hstar, delta, L, flag, PC);
539 CellsMoved(PassCount)=CellCt;
540 while flag==1
541     PassCount=PassCount+1;
542     [Hstar, flag, CellCt]=AVALANCHE03312021(Hstar, delta, L, flag, PC);
543     CellsMoved(PassCount)=CellCt;%
544 end
545 CellsMoved=CellsMoved(1, 1:PassCount);
546 AvIC_Array2=[NaN PassCount CellsMoved];
547 H=Hstar;
548
549 %***** NOT USING LAND CATEGORIZATION - BUT MAY AGAIN LATER - DO NOT DELETE!!
550 %     %Initial Declare all categories in H
551 PlantColumnArray=zeros([size(H,1) 1]); %P3 isn't allowed to grow on the beach - this is to check for that
552 PlantColumnArray2=zeros([size(H,1) 1]);
553 [Hstar, P1, P2, P3, P4]=LandCategorizationUNUSED03312021(Hstar, P1, P2, P3, P4, PlantColumnArray, PlantColumnArray2, ColumnArraySwamp1,
        ColumnArraySwamp2, INum);
554 %***** NOT USING LAND CATEGORIZATION - BUT MAY AGAIN LATER - DO NOT DELETE!!
555
556 if SingleStepICIm==1
557     figure
558     colormap(jet())
559     imagesc(Hstar(:, :, 2), climH)
560     colorbar
561     ('ICs LandCats - After Swamp, Marine Processes and Avalanche');
562 end
563 fprintf('\n')
564 fprintf('Initialization complete')
565
566 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
567 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%END INITIAL PARAMETERIZATION%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
568 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
569 %%
570 tic
571 T=zeros(time, 1);
572 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```

```

573 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
574 %!@##$$                %%%MAIN LOOP%%                $$#@!%!%
575 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
576 %#####
577 tStart=cputime;
578 for t=0:time
579     TimeStep=t
580     Ptot=P1(:,1)+P2(:,1)+P3(:,1)+P4(:,1)+P3d(:,1);
581     if 0==mod(t,26)
582         if t==0
583             fprintf('\n')
584             fprintf('Here we go...')
585         elseif t==26
586             fprintf('\n')
587             fprintf('%d year has passed',t/26)
588         else
589             fprintf('\n')
590             fprintf('%d years have passed',t/26)
591         end
592     end
593     % TimeStep=t
594     %% PC=(P1ErosionCoefficient.*P1(:,1))+(P2ErosionCoefficient.*P2(:,1))+(P3ErosionCoefficient.*P3(:,1))+
595         P4ErosionCoefficient.*P4(:,1));
596     %need to work this into the inside of PlantProcesses
597     for i=1:size(H,1)
598         for j=1:size(H,2)
599             PC1(i,j)=P1(i,j,1)*(P1(i,j,1)>0);
600             PC2(i,j)=P2(i,j,1)*(P2(i,j,1)>0);
601             PC3(i,j)=P3(i,j,1)*(P3(i,j,1)>0);
602             PC4(i,j)=P4(i,j,1)*(P4(i,j,1)>0);
603         end
604     end
605     PC=(P1ErosionCoefficient.*PC1(:,:))+
606         (P2ErosionCoefficient.*PC2(:,:))+
607         (P3ErosionCoefficient.*PC3(:,:))+
608         (P4ErosionCoefficient.*PC4(:,:))+
609         (P3dErosionCoefficient.*P3d(:,1));
610     %% (1) Run Swamp
611     [Hstar,ColumnArraySwamp1,ColumnArraySwamp2,AdjacentLengthSwamp,OppositeLocationSwamp,flag,INum]=SwampProcesses03312021(Hstar,
612         delta,L,flag,PC);
613     %% (1.AV) THIS AVALANCHE RUNS ONCE PER YEAR (every 26 time steps) AND IS THE FIRST AVALANCHE
614     % if (0==mod(t,26)) && t~=0
615     PassCount=1;
616     CellsMoved=zeros(1,1000);
617     [Hstar,flag,CellCt]=AVALANCHEtime03312021(Hstar,delta,L,flag,PC,t);
618     CellsMoved(PassCount)=CellCt;
619     while flag==1
620         PassCount=PassCount+1;
621         [Hstar,flag,CellCt]=AVALANCHEtime03312021(Hstar,delta,L,flag,PC,t);
622         CellsMoved(PassCount)=CellCt;%
623         if CellCt<=5
624             flag=0;
625         end
626     end
627     CellsMoved=CellsMoved(1,1:PassCount);

```

```

626 SwampAvArray=[NaN PassCount CellsMoved];
627 AvIC_Array1=SwampAvArray;
628 AV_ARRAY(TimeStep+1,1:PassCount+2,1)=SwampAvArray;
629
630
631 %***** NOT USING LAND CATEGORIZATION - BUT MAY AGAIN LATER - DO NOT DELETE!!
632 for i=1:size(Hstar,1)
633     for Icount=1:INum
634         if ColumnArraySwamp2(i,Icount)~=0
635             for j=ColumnArraySwamp1(i,Icount):ColumnArraySwamp2(i,Icount)
636                 Hstar(i,j,2)=2;
637             end
638         end
639     end
640 end
641 %***** NOT USING LAND CATEGORIZATION - BUT MAY AGAIN LATER - DO NOT DELETE!!
642
643
644 %%    %(2) Aeolian Transport
645
646 if WindData==0
647     if Windspeed>=6
648         WindCnt=floor(((max(Windspeed-Windmin,0))/(StormThreshold-Windmin))*24);
649         for day=1:14
650             %                for hour =1:WindCnt
651                 [Hstar]=AeolianTransport03212021(Hstar,delta,L,W,Windspeed,Windmin,StormThreshold,WindDir,PC);
652             %                end
653         end%
654     end
655 end
656
657 if WindData==1 && t>0
658     %WindDir=randi([1,8]); can randomize wind or use data-weighted wind directions outlined below. (comment this if you
        randomize)
659     for day=1:14
660         WindSpeedParam=randi(2183);
661         if WindSpeedParam>=1 && WindSpeedParam<=2116
662             Windspeed=randi([0,5]);
663         elseif WindSpeedParam>2116 && WindSpeedParam<=2183
664             ATcntr=ATcntr+1;
665             Windspeed=randi([6,15]);
666         elseif WindSpeedParam>2183
667             Windspeed=16;
668         end
669         WindParam=randi(2107);
670         if WindParam>=1 && WindParam<=250
671             WindDir=1;
672         elseif WindParam>=251 && WindParam<=655
673             WindDir=2;
674         elseif WindParam>=656 && WindParam<=835
675             WindDir=3;
676         elseif WindParam>=836 && WindParam<=1091
677             WindDir=4;
678         elseif WindParam>=1092 && WindParam<=1464
679             WindDir=5;
680         elseif WindParam>=1465 && WindParam<=1717

```



```

681         WindDir=6;
682         elseif WindParam>=1718 && WindParam<=1865
683             WindDir=7;
684         elseif WindParam>=1866 && WindParam<=2107
685             WindDir=8;
686         end
687
688         if Windspeed>=6
689             %           WindCnt=floor(((max(Windspeed-Windmin,0))/(StormThreshold-Windmin))*24);
690             %           for hour=1:WindCnt
691                 [Hstar]=AeolianTransport03312021(Hstar,delta,L,W,Windspeed,Windmin,StormThreshold,WindDir,PC);
692             %           end
693         end
694     end
695 end
696
697 %%      %(2.AV)                THIS AVALANCHE RUNS ONCE PER YEAR (every 26 time steps) AND IS THE SECOND AVALANCHE
698 % if (0==mod(t,26)) && t~=0
699 PassCount=1;
700 CellsMoved=zeros(1,1000);
701 [Hstar,flag,CellCt]=AVALANCHEtime03312021(Hstar,delta,L,flag,PC,t);
702 CellsMoved(PassCount)=CellCt;
703 while flag==1
704     PassCount=PassCount+1;
705     [Hstar,flag,CellCt]=AVALANCHEtime03312021(Hstar,delta,L,flag,PC,t);
706     CellsMoved(PassCount)=CellCt;%
707     %           if CellCt<=5
708     %               flag=0;
709     %           end
710 end
711 CellsMoved=CellsMoved(1,1:PassCount);
712 Av2_Array=[TimeStep PassCount CellsMoved];
713 AV_ARRAY(TimeStep+1,1:PassCount+2,2)=Av2_Array;
714 % end
715
716 %%      %(3) Plant Propogation
717
718 %land categroization used to be required for plant processes - this might be removeable
719 %***** NOT USING LAND CATEGORIZATION - BUT MAY AGAIN LATER - DO NOT DELETE!!
720 [Hstar,P1,P2,P3,P4]=LandCategorizationUNUSED03312021(Hstar,P1,P2,P3,P4,PlantColumnArray,PlantColumnArray2,ColumnArraySwamp1,
721     ColumnArraySwamp2,INum);
722 %***** NOT USING LAND CATEGORIZATION - BUT MAY AGAIN LATER - DO NOT DELETE!!
723
724 if (0==mod(t,13)) % choose 26 for 2-week timesteps; 52 for 1-week timesteps
725     [Hstar,MeanBeachWidth,ESL,WSL,MESL,MWSL,OL]=Shoreline03312021(Hstar,delta,L,BchMax);%get shoreline to use in
726     PlantPropagation
727     [P1,P2,P3,P4,P3d]=PlantPropagation03312021(Hstar,t,P1,P2,P3,P4,W,S,delta,P3d,MaxSwampWidth,PlantRangeArray,alpha,DBE,
728     gdrange1,gdrange2,PctMax,MasterMax,MWSL,MESL,ESL);
729     if P3IC==0 %killing off any morella that was established due to bird activity
730         P3(:,1)=min(P3(:,1),0);
731     end
732 end
733
734 for i=1:size(H,1)
735     for j=1:size(H,2)
736         PC1(i,j)=P1(i,j,1)*(P1(i,j,1)>0);

```

```

734         PC2(i,j)=P2(i,j,1)*(P2(i,j,1)>0); %
735         PC3(i,j)=P3(i,j,1)*(P3(i,j,1)>0);
736         PC4(i,j)=P4(i,j,1)*(P4(i,j,1)>0);
737     end
738 end
739 PC=(P1ErosionCoefficient.*PC1(:,:))+(P2ErosionCoefficient.*PC2(:,:))+(P3ErosionCoefficient.*PC3(:,:))+(P4ErosionCoefficient.*
    PC4(:,:))+(P3dErosionCoefficient.*P3d(:,: ,1));
740 Ptot=PC1(:,: ,1)+PC2(:,: ,1)+PC3(:,: ,1)+PC4(:,: ,1)+P3d(:,: ,1);
741
742 %% (4) Marine Processes - edits
743 %currently every 12 months, moves 2 slabs into ocean
744 if (0==mod(t,26))
745     if MPswitch==1
746         PCmp=-999*ones(ROW,COLUMN);
747         for i=1:ROW
748             for j=1:COLUMN
749                 if P1(i,j,1)>=0 || P2(i,j,1)>=0 || P3(i,j,1)>=0 || P4(i,j,1)>=0
750                     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));
751                 end
752             end
753         end
754         % PCmp=(P1ErosionCoefficient.*P1(:,: ,1))+(P2ErosionCoefficient.*P2(:,: ,1))+(P3ErosionCoefficient.*P3
            (:,: ,1))+(P4ErosionCoefficient.*P4(:,: ,1))+(P3dErosionCoefficient.*P3d(:,: ,1));
755         [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);
756     elseif MPswitch==2
757         [Hstar ,PlantColumnArray ,PlantColumnArray2 ,P3 ,flag ,MeanBeachWidth]=OLDMarineProcesses03312021(Hstar ,delta ,L ,flag ,PC ,P3 ,
            BchMax ,BchW);
758     end
759     if MPswitch~=0
760         MeanBeachWidthVec(1)=MeanBeachWidth;
761         H=Hstar;
762     end
763 end
764 end
765
766 %% (4.AV*)update PC before avalanching again
767 %need to work this into the inside of PlantProcesses
768 for i=1:size(H,1)
769     for j=1:size(H,2)
770         % PC1(i,j)=P1(i,j,1)*(P1(i,j,1)>0);
771         % PC2(i,j)=P2(i,j,1)*(P2(i,j,1)>0); %
772         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
773         % PC4(i,j)=P4(i,j,1)*(P4(i,j,1)>0);
774     end
775 end
776 PC=(P1ErosionCoefficient.*PC1(:,:))+(P2ErosionCoefficient.*PC2(:,:))+(P3ErosionCoefficient.*PC3(:,:))+(P4ErosionCoefficient.*
    PC4(:,:))+(P3dErosionCoefficient.*P3d(:,: ,1));
777 %% (4.AV) Avalanche after Marine Processes THIS AVALANCHE RUNS EVERY TIME STEP AND IS THE FINAL AVALANCHE
778 PassCount=1;
779 CellsMoved=zeros(1,1000);
780 [Hstar ,flag ,CellCt]=AVALANCHEtime03312021(Hstar ,delta ,L ,flag ,PC ,t);
781 CellsMoved(PassCount)=CellCt;
782 while flag==1
783     PassCount=PassCount+1;

```

```

784     [Hstar , flag , CellCt]=AVALANCHEtime03312021(Hstar , delta ,L, flag ,PC, t);
785     CellsMoved(PassCount)=CellCt;
786 end
787 CellsMoved=CellsMoved(1,1:PassCount);
788 Av3_Array=[TimeStep PassCount CellsMoved];
789 AV_ARRAY(TimeStep+1,1:PassCount+2,3)=Av3_Array;
790 %%      %(5) Data and categories in H
791 %***** NOT USING LAND CATEGORIZATION – BUT MAY AGAIN LATER – DO NOT DELETE!!
792 [Hstar ,P1 ,P2 ,P3 ,P4]=LandCategorizationUNUSED03312021(Hstar ,P1 ,P2 ,P3 ,P4 ,PlantColumnArray ,PlantColumnArray2 ,ColumnArraySwamp1 ,
    ColumnArraySwamp2 ,INum);
793 %***** NOT USING LAND CATEGORIZATION – BUT MAY AGAIN LATER – DO NOT DELETE!!
794 [Hstar ,Centroid]=Qdata03312021(Hstar ,P1 ,P2 ,P3 ,P4);% ,ESL ,WSL ,MESL ,MWSL ,OppositeLocation);
795 CentroidVec(t+1,:)=Centroid;
796
797 %calculating mean percent cover for each Pi
798 p1=P1(:, :, 1)>0;p2=P2(:, :, 1)>0;p3=P3(:, :, 1)>0;p4=P4(:, :, 1)>0;
799 P1gt=P1(:, :, 1).*p1;
800 P2gt=P2(:, :, 1).*p2;
801 P3gt=P3(:, :, 1).*p3;
802 P4gt=P4(:, :, 1).*p4;
803
804 MnP1(t+2)=sum(P1gt, 'all')/sum(p1, 'all');
805 MnP2(t+2)=sum(P2gt, 'all')/sum(p2, 'all');
806 MnP3(t+2)=sum(P3gt, 'all')/sum(p3, 'all');
807 MnP4(t+2)=sum(P4gt, 'all')/sum(p4, 'all');
808
809 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
810 %%%/%%/%%                      Sea Level Rise                      %%/%%/%%
811 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
812 if SLRswitch==1
813     if 0==mod(t,156)%0==mod(t,156)    %156 timesteps = 312 weeks = 6 years
814         % if 0==mod(t,260)
815         Hstar=Hstar-1; %working estimate for SLR in VA is 1.66cm/yr = .1m/6yrs
816     end
817 end
818 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
819 %%%/%%/%%                      End Sea Level Rise                      %%/%%/%%
820 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
821
822 H=round(Hstar,0);          %***** THIS IS THE ONLY UPDATE FOR H IN THE MAIN LOOP – OTHERWISE ALL UPDATES ARE MADE TO HSTAR
823
824 if t==1
825     Sv1=H(:, :, 1);
826     Sv1f=round(imgaussfilt(H(:, :, 1),4),0);
827     Sv1Cent=Centroid;
828 elseif t==338
829     Sv2=H(:, :, 1);
830     Sv2f=round(imgaussfilt(H(:, :, 1),4),0);
831     Sv2Cent=Centroid;
832 elseif t==702
833     Sv3=H(:, :, 1);
834     Sv3f=round(imgaussfilt(H(:, :, 1),4),0);
835     Sv3Cent=Centroid;
836 end
837
838 tEnd=cputime-tStart;

```

```

839
840
841 T(t+1)=toc;
842 fprintf('\n')
843 if t~=0
844     Tnew=T(t+1)-T(t);
845 else
846     Tnew=T(1);
847 end
848 fprintf('t=%d time: %0.1f',TimeStep,Tnew)
849 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
850 %%%                %%% During Images %%%
851 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
852
853 % % % % ELEVATION
854
855 if ElevIm.during==1
856     Hfilt=round(imgaussfilt(H(:, :, 1), 4), 0);
857     if (0==mod(t, DurImFreq))
858
859         %
860         figure
861
862         %
863         colormap(jet())
864         imagesc(H(:, :, 1), climH);
865         hold on
866         if MElevCont==1
867             contour(H(:, :, 1), [-5 -5], 'color', 'm', 'linewidth', 1.1)
868         end
869         if SElevCont==1
870             contour(Hfilt(:, :, 1), [-0 -0], 'color', 'k', 'linewidth', 1.1)
871         end
872         if MElevCont==1 && SElevCont==1
873             lgnd=legend('\color{white} marsh contour (-0.5m)', '\color{white} sea level contour (0m)');
874             set(lgnd, 'color', 'none', 'location', 'southeast');
875         elseif MElevCont==1 && SElevCont==0
876             lgnd=legend('\color{white} marsh contour (-0.5m)');
877             set(lgnd, 'color', 'none', 'location', 'southeast');
878         elseif SElevCont==1 && MElevCont==0
879             lgnd=legend('\color{white} sea level contour (0m)');
880             set(lgnd, 'color', 'none', 'location', 'southeast');
881         end
882         colorbar
883         title(sprintf('Elevation at %d years', t/26))
884         hold off
885     end
886 end
887
888 %This is the unsmoothed transect migration image
889 % % % if TransectIm==1
890 % % %     if 0==mod(t, TranImFreq)
891 % % %         Trow=H(TranRow, :, 1);
892 % % %         Trow(Trow<=-5)=-6;
893 % % %         xlim1=find(Trow>=-5, 1, 'first')-100; xlim2=find(Trow>=-5, 1, 'last')+100;

```

```

891 % % %         figure(1000)
892 % % %         hold on
893 % % %         plot(1:size(H,2),Trow,'Linewidth',2)
894 % % %         xlim([xlim1 xlim2])
895 % % %         ylim([-5,max(Trow)+20])
896 % % %         title(sprintf('Island transect at row %d',TranRow))
897 % % %         hold off
898 % % %     end
899 % % % end
900 %%%Close up elev:
901 if ElevImCU_during==1
902     if (0==mod(t,DurCUImFreq))
903         Wndw=250;
904         MidRow=round(Centroid(2));
905         MidCol=round(Centroid(1));
906         figure
907         imagesc(H(MidRow-Wndw:MidRow+Wndw,MidCol-Wndw:MidCol+Wndw,1),climsH);
908         hold on
909         ContourVec=[0:10:max(max(Hfilt))];
910         colorbar
911         contour(Hfilt(MidRow-Wndw:MidRow+Wndw,MidCol-Wndw:MidCol+Wndw,1),ContourVec);
912         contour(Hfilt(MidRow-Wndw:MidRow+Wndw,MidCol-Wndw:MidCol+Wndw,1),ContourVec,'LineColor','k');
913         [C,h]=contour(Hfilt(MidRow-Wndw:MidRow+Wndw,MidCol-Wndw:MidCol+Wndw,1),ContourVec,'LineColor','k');
914         clabel(C,h)
915         title(sprintf('Elevation Close-Up about Centroid at %d years',t/26))
916 % % % old one (lines at 10, 20 close up):
917 %         figure

%
918 %         colormap(jet())
919 %         MidPt=round(CentroidVec(t+1,:));
920 %         MidCol=MidPt(1); MidRow=MidPt(2);
921 %         imagesc(H(MidRow-50:MidRow+50,MidCol-50:MidCol+50,1),climsH);
922 %         hold on
923 %         ContourVec=[0:5:max(max(Hfilt))];
924 %         contour(Hfilt(MidRow-50:MidRow+50,MidCol-50:MidCol+50,1),[10 10],'Color','m','linewidth',1.1);
925 %         contour(Hfilt(MidRow-50:MidRow+50,MidCol-50:MidCol+50,1),[20 20],'Color','k','linewidth',1.1);
926 %         lgnd=legend('\color{white} marsh contour (-0.5m)','\color{white} sea level contour (0m)');
927 %         set(lgnd,'color','none','location','southeast');
928 %         colorbar
929 %         title(sprintf('Elevation Close-Up about Centroid at %d years',t/26))
930 %         hold off
931
932
933         hFig=figure;
934         set(gcf,'PaperPositionMode','auto')
935         set(hFig,'units','inches')
936         hFig.Position(1)=-.5;
937         hFig.Position(2)=-.5;
938         hFig.Position(3)=2*hFig.Position(3);
939         hFig.Position(4)=1.25*hFig.Position(4);
940         subplot(1,2,1)
941         colormap(jet())
942         imagesc(H(MidRow-Wndw:MidRow+Wndw,MidCol-Wndw:MidCol+Wndw,1),climsH);
943         title(sprintf('Elevation Close-Up about Centroid at %d years',t/26))
944         subplot(1,2,2)

```

```

945     [C,h] = contour( Hfilt(MidRow-Wndw:MidRow+Wndw, MidCol-Wndw: MidCol+Wndw,1) ,ContourVec);
946     xlabel(C,h)
947     set(gca, 'YDir', 'reverse')
948     title(sprintf('Contours of Elevation Close-Up about Centroid at %d years',t/26))
949
950
951     figure
952     subplot(2,2,1)
953     colormap bone
954     imagesc(P1(MidRow-Wndw:MidRow+Wndw, MidCol-Wndw: MidCol+Wndw,1) ,climsP);
955     title(sprintf('P.1 about centroid at %d years',t/26))
956     subplot(2,2,2)
957     colormap bone
958     imagesc(P2(MidRow-Wndw:MidRow+Wndw, MidCol-Wndw: MidCol+Wndw,1) ,climsP);
959     title(sprintf('P.2 about centroid at %d years',t/26))
960     subplot(2,2,3)
961     colormap bone
962     imagesc(P3(MidRow-Wndw:MidRow+Wndw, MidCol-Wndw: MidCol+Wndw,1) ,climsP);
963     title(sprintf('P.3 about centroid at %d years',t/26))
964     subplot(2,2,4)
965     colormap bone
966     imagesc(P4(MidRow-Wndw:MidRow+Wndw, MidCol-Wndw: MidCol+Wndw,1) ,climsP);
967     title(sprintf('P.4 about centroid at %d years',t/26))
968     end
969 end
970
971 % % % %Land Category
972
973 if LandCatIm.during==1
974     if (0==mod(t, DurImFreq)) %every 5 years
975
976         %
977         figure
978
979         %
980         colormap(jet())
981         clims=[min(min(H(:, : ,2))) max(max(H(:, : ,2)))]];
982         imagesc(H(:, : ,2) ,clims);
983         colorbar
984         title(sprintf('Land Category at time step = %d',t))
985     end
986 end
987
988 % % % %Plant Percent Cover%
989
990 if PlantPCIm.during==1
991     if (0==mod(t, PlantDurImFreq)) % 2~/mo, choose 26~/year
992
993         %%% AMMOPIILA
994         figure
995         colormap gray
996         %         clims=[-1 max(0,max(max(P1(:, : ,1))))];
997         imagesc(P1(:, : ,1) ,climsP)
998         colorbar
999         title(sprintf('P1-Ammophila at %d years',t/26))
1000         %         title(sprintf('P1-Ammophila at time step = %d',t))

```

```

997         %%% SPARTINA
998         figure
999         colormap gray
1000        %       clims=[-1 max(0,max(max(P2(:,:,1))))];
1001        imagesc(P2(:,:,1),climsP)
1002        colorbar%
1003        title(sprintf('P2 - Spartina patens at %d years',t/26))
1004        %       title(sprintf('P2 - Spartina patens at time step = %d',t))
1005        %%% MORELLA
1006        figure
1007        colormap gray
1008        %       clims=[-1 max(0,max(max(P3(:,:,1))))];
1009        imagesc(P3(:,:,1),climsP)
1010        colorbar%
1011        title(sprintf('P3 - Morella at %d years',t/26))
1012        %       title(sprintf('P3 - Morella at time step = %d',t))
1013        %%% DEAD MORELLA
1014        figure
1015        colormap gray
1016        %       clims=[-1 max(0,max(max(P3d(:,:,1))))];
1017        imagesc(P3d(:,:,1),climsP)
1018        colorbar
1019        title(sprintf('P3d - DEAD Morella at %d years',t/26))
1020        %       title(sprintf('P3d - DEAD Morella at time step = %d',t))
1021        %%% SPARTINA (marsh)
1022        figure
1023        colormap gray
1024        %       clims=[-1 max(1,max(max(P4(:,:,1))))];
1025        imagesc(P4(:,:,1),climsP)
1026        colorbar
1027        title(sprintf('P4 - Spartina alterniflora at %d years',t/26))
1028        %       title(sprintf('P4 - Spartina alterniflora at time step = %d',t))
1029    end
1030
1031    end
1032
1033    end
1034    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
1035    %%% END OF MAIN LOOP %%%
1036    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
1037    %%
1038    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
1039    %%% After Images %%%
1040    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
1041
1042    % % % % Elevation
1043    if ElevIm.after==1
1044        figure
1045        colormap(jet())
1046        imagesc(H(:,:,1),climsH);
1047        hold on
1048        contour(H(:,:,1),[-0.5 -0.5], 'color', 'black')
1049        colorbar
1050        title('After Image')
1051    end
1052

```

```

1053 %Mean Plant Percent Cover Images
1054 if MnPlantCvrIm==1
1055     figure
1056     hold on
1057     plot(MnP1)
1058     plot(MnP2)
1059     plot(MnP3)
1060     plot(MnP4)
1061     legend('P_1','P_2','P_3','P_4')
1062     xlabel('time in two week steps')
1063     ylabel('Mean Percent Cover')
1064     title(sprintf('Mean Percent Cover - %.0f years',time/26))
1065 end
1066
1067 % % % %Land Category
1068
1069 if LandCatIm_.after==1
1070     figure
1071     colormap(jet())
1072     clim=[min(min(H(:, :, 2))) max(max(H(:, :, 2)))]);
1073     imagesc(H(:, :, 2),clim);
1074     colorbar
1075     title(sprintf('Land Category - After'))
1076 end
1077
1078 % % % %PLANTS (percent cover)
1079
1080 if PlantPCIm_.after==1
1081     %%% AMMOPHILA
1082     figure
1083     colormap gray
1084     imagesc(P1(:, :, 1),climP)
1085     colorbar
1086     title(sprintf('P1-Ammophila After %d years',time/26))
1087     %%% SPARTINA
1088     figure
1089     colormap gray
1090     imagesc(P2(:, :, 1),climP)
1091     colorbar
1092     title(sprintf('P2 - Spartina patens after %d years',time/26))
1093     %%% MORELLA
1094     figure
1095     colormap gray
1096     imagesc(P3(:, :, 1),climP)
1097     colorbar%
1098     title(sprintf('P3 - Morella after %d years',time/26))
1099     %%% SPARTINA (marsh)
1100     figure
1101     colormap gray
1102     imagesc(P4(:, :, 1),climP)
1103     colorbar
1104     title(sprintf('P4 - Spartina alterniflora after %d years',time/26))
1105 end
1106
1107 % % Contours at 0, 13, and 27 years
1108 if Contour27==1

```



```

1109 %Contours of the shoreline (0m elevation)
1110 figure

        %
1111 colormap(jet())
1112 hold on
1113 contour(Sv1f(:, :, 1), [1 1], 'Color', 'blue', 'LineWidth', 2);           %beginning contour (after initialization)
1114 contour(Sv2f(:, :, 1), [1 1], 'Color', 'green', 'LineWidth', 2);       %contour at 13 years (t=338)
1115 contour(Sv3f(:, :, 1), [1 1], 'Color', 'red', 'LineWidth', 2);        %contour at 27 years (t=702)
1116 scatter(Sv1Cent(1), Sv1Cent(2), 'o', 'MarkerEdgeColor', 'k', 'MarkerFaceColor', 'b')
1117 scatter(Sv2Cent(1), Sv2Cent(2), 'o', 'MarkerEdgeColor', 'k', 'MarkerFaceColor', 'g')
1118 scatter(Sv3Cent(1), Sv3Cent(2), 'o', 'MarkerEdgeColor', 'k', 'MarkerFaceColor', 'r')
1119 % scatter(Sv1Cent(1), Sv1Cent(2), 'o', 'MarkerEdgeColor', 'w', ...
1120 %             'MarkerFaceColor', 'b', ...
1121 %             'LineWidth', 1.25)
1122 % scatter(Sv2Cent(1), Sv2Cent(2), 'o', 'MarkerEdgeColor', 'w', ...
1123 %             'MarkerFaceColor', 'g', ...
1124 %             'LineWidth', 1.25)
1125 % scatter(Sv3Cent(1), Sv3Cent(2), 'o', 'MarkerEdgeColor', 'w', ...
1126 %             'MarkerFaceColor', 'r', ...
1127 %             'LineWidth', 1.25)
1128 set(gca, 'YDir', 'reverse')
1129 legend('Initial', '13 years', '27 years', 'Location', 'southeast')
1130 title('Island Contours at 0m Elevation')
1131
1132 %contours including the marsh area:
1133 % % % figure

        %
1134 % % % colormap(jet())
1135 % % % hold on
1136 % % % contour(Sv1f(:, :, 1), [-5 -5], 'Color', 'blue', 'LineWidth', 2);           %beginning contour (after initialization)
1137 % % % contour(Sv2f(:, :, 1), [-5 -5], 'Color', 'green', 'LineWidth', 2);       %contour at 13 years (t=338)
1138 % % % contour(Sv3f(:, :, 1), [-5 -5], 'Color', 'red', 'LineWidth', 2);        %contour at 27 years (t=702)
1139 % % % scatter(Sv1Cent(1), Sv1Cent(2), 'o', 'MarkerEdgeColor', 'k', 'MarkerFaceColor', 'b')
1140 % % % scatter(Sv2Cent(1), Sv2Cent(2), 'o', 'MarkerEdgeColor', 'k', 'MarkerFaceColor', 'g')
1141 % % % scatter(Sv3Cent(1), Sv3Cent(2), 'o', 'MarkerEdgeColor', 'k', 'MarkerFaceColor', 'r')
1142 % % % % scatter(Sv3Cent(1), Sv3Cent(2), '*r')
1143 % % % % scatter(Sv2Cent(1), Sv2Cent(2), '*g')
1144 % % % % scatter(Sv1Cent(1), Sv1Cent(2), '*b')
1145 % % % set(gca, 'YDir', 'reverse')
1146 % % % legend('Initial', '13 years', '27 years', 'Location', 'southeast')
1147 % % % title('Island Contours with Marshland (at -0.5m Elevation)')
1148 end
1149
1150 if TransectIm==1
1151     figure
1152     Trow1=Sv1f(TranRow, :, 1);
1153     Trow2=Sv2f(TranRow, :, 1);
1154     Trow3=Sv3f(TranRow, :, 1);
1155     Trow1(Trow1<-5)=-6; Trow2(Trow2<-5)=-6; Trow3(Trow3<-5)=-6;
1156     xlim1=find(Trow1>=-5, 1, 'first')-100; xlim2=find(Trow1>=-5, 1, 'last')+100;
1157     hold on
1158     plot(1:size(H, 2), Trow1, '-','Linewidth', 2)
1159     plot(1:size(H, 2), Trow2, '-','Linewidth', 2)
1160     plot(1:size(H, 2), Trow3, '-','Linewidth', 2)

```

```
1161     xlim([xlim1 xlim2])
1162     ylim([-5,max(Trow1)+20])
1163     title(sprintf('Island transect at row %d',TranRow))
1164     legend('Initial','13 years','27 years','Location','northwest')
1165     hold off
1166 end
1167
1168 fprintf('DONE! ')
1169 toc
1170
1171 %!!!!!!!!!!!!!!!!!!!!!!!!!!!!no, seriously ... it's over!!!!!!!!!!!!!!!!!!!!!!!!!!!!%
```

B.2 Aeolian Transport code

```

1 function [Hstar]=AeolianTransport03312021(Hstar,delta,L,W,Windspeed,Windmin,StormThreshold,WindDir,PC)
2
3 %Aeolian Transport is triggered by sufficient wind speeds chosen in the MainCode.
4 %AT determines the number of cells that the sediment will move based on the
5 %current windspeed:
6 %             windspeed < 6 move 1 cell
7 %             6<= windspeed < 11 move 2 cells
8 %             11<= windspeed < 16 move 3 cells
9
10 %When each cell is polled the plant density on that cell is accounted for,
11 %but it is assumed that after initial saltation the sediment is moving
12 %freely enough so as not to require polling subsequent cells' plant cover.
13 %The angle between the current cell and the next potential cell is checked
14 %for every step 1, 2, and 3.
15
16 %each cell has a probability of moving either in the same direction as the
17 %EindDirection input or one of the two off-directions
18 % (i.e. if WindDir is North: 50% move North, 25% move NorthEast, 25% move NorthWest
19
20
21 MvF=.8; %movement probability factor (turn up for more movement, down for less)
22
23 Hp=padarray(Hstar(:,:,1),[3 3]); %padding Hstar with three extra rows/cols in each direction - avoids index errors
24 %cells moved into padded region are
25 %deleted when Hstar is redeclared
26 %after each loop
27 [n1 n2]=size(Hstar(:,:,1)); %array indexes for loop
28
29 %
30 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
31
32
33 %can move up to three cells based on windspeed
34 MvCnt=1*(Windspeed>=6)+1*(Windspeed>=9)+1*(Windspeed>=12);
35
36 %***about MarshFlag*** - we are currently allowing the wind to affect cells in
37 %the swamp, but previously we were flagging them so that they would not be
38 %moved. If you do not want the wind to move swamp cells then change the
39 %rows labeled below
40
41 for i=4:n1+3
42     Rnow=Hp(i,:);
43     for j=n2+3:-1:4
44         MarshFlag=0; %flagging cells in the marsh
45         if Rnow(j)>=0
46             if (Rnow(j)> 0)&& Rnow(j)<=1
47                 if j==1
48                     if Rnow(j)<Rnow(j+1) && Rnow(j+1)<Rnow(j+2)
49                         MarshFlag=0; %<---change to MarshFlag=1; if wish to ignore marsh cells
50                     end
51                 elseif j==n2

```

```

52         if Rnow(j)>Rnow(j-1) && Rnow(j-1)>Rnow(j-2)
53             MarshFlag=0; %<—change to MarshFlag=1; if wish to ignore marsh cells
54         end
55     else
56         if Rnow(j)<Rnow(j+1) && Rnow(j-1)<Rnow(j)
57             MarshFlag=0; %<—change to MarshFlag=1; if wish to ignore marsh cells
58         end
59     end
60 end
61
62 if Rnow(j)>0 && MarshFlag==0
63     %probability for RNG use
64     P=0.5; %P splits the likelihood of moving in primary wind direction or the two alternative directions
65     Rand=rand; %for probability check
66     MoveChance=(1-PC(i,j))*((max(Windspeed-Windmin,0))/(StormThreshold-Windmin))*MvF; %cells chance of moving,
        gactors in plants and windpseed ranges
67     dist=round(MvCnt*MoveChance,0); %number of cells a slab can potentially move
68
69     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
70     %% DEPOSITION RULES: %%
71     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
72
73     %Wind Direction
74     %1=N 2=NE 3=E 4=SE 5=S 6=SW 7=W 8=NW
75
76     Nhd=Hp(i-3:i+3,j-3:j+3); %7x7 moving window for movement up to three cells
77     PCNhd=PC(i-3:i+3,j-3:j+3);
78 %
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
79
80     %NORTH BLOCK
81
82     %NORIH - NW
83
84     if WindDir==1 && Rand<MoveChance
85         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
86             DirV=[Nhd(4,4), Nhd(3,3), Nhd(2,2), Nhd(1,1)]; %vector of neighborhood Hstar values in the direction
                of chosen movement
87
88             %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
89             PCNE=[PCNhd(4,4), PCNhd(3,3), PCNhd(2,2), PCNhd(1,1)];
90
91             flag=1; %trigger the cell movement
92             k=1; %count for number of cells moved so far in while loop
93             while flag==1
94                 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
95                     DirV(k)=DirV(k)-1; %remove one slab from current cell
96                     DirV(k+1)=DirV(k+1)+1; %add that cell to neighbor
97                     k=k+1; %add one to cell moved count
98                     flag=1*(k<=dist); %repeat until k is equal to distance calculated
99                     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
100 MoveChance=(1-PCNE(k))*((max(Windspeed-Windmin,0))/(StormThreshold-Windmin))*MvF;
101 if Rand >= MoveChance
102     flag=0;
103 end
104 else
105     flag=0; %if too steep an angle is encountered, movement
106     stops
107 end
108 Nhd(4,4)=DirV(1); %replace neighborhood values with the direction
109     vector values
110 Nhd(3,3)=DirV(2);
111 Nhd(2,2)=DirV(3);
112 Nhd(1,1)=DirV(4);
113 %NORTH - N
114 elseif Rand>=((1-P)/2) && Rand<((1+P)/2) %&& (i>1) %move with the wind / North
115 DirV=[Nhd(4,4), Nhd(3,4), Nhd(2,4), Nhd(1,4)];
116 PCNE=[PCNhd(4,4), PCNhd(3,4), PCNhd(2,4), PCNhd(1,4)];
117 flag=1;
118 k=1;
119 while flag==1
120     if atan(((DirV(k+1)-DirV(k))*delta)/L)<(pi/12)
121         DirV(k)=DirV(k)-1;
122         DirV(k+1)=DirV(k)+1;
123         cellcnt=cellcnt+1;
124         MoveChance=(1-PCNE(k+1))*((max(Windspeed-Windmin,0))/(StormThreshold-Windmin));
125         k=k+1;
126         flag=1*(k<=dist);
127         MoveChance=(1-PCNE(k))*((max(Windspeed-Windmin,0))/(StormThreshold-Windmin))*MvF;
128         if Rand >= MoveChance
129             flag=0;
130         end
131     else
132         flag=0;
133     end
134 end
135 Nhd(4,4)=DirV(1);
136 Nhd(3,4)=DirV(2);
137 Nhd(2,4)=DirV(3);
138 Nhd(1,4)=DirV(4);
139
140
141 %NORTH - NE
142 elseif Rand>=((1+P)/2) && (i>1) && (j<size(Hstar,2)) %move to the starboard slab
143 DirV=[Nhd(4,4), Nhd(3,5), Nhd(2,6), Nhd(1,7)];
144 PCNE=[PCNhd(4,4), PCNhd(3,5), PCNhd(2,6), PCNhd(1,7)];
145 flag=1;
146 k=1;
147 while flag==1
148     if atan(((DirV(k+1)-DirV(k))*delta)/L)<(pi/12)
149         DirV(k)=DirV(k)-1;
150         DirV(k+1)=DirV(k)+1;
151         cellcnt=cellcnt+1;
152         MoveChance=(1-PCNE(k+1))*((max(Windspeed-Windmin,0))/(StormThreshold-Windmin));

```

```

153         k=k+1;
154         flag=1*(k<=dist);
155         MoveChance=(1-PCNE(k))*((max(Windspeed-Windmin,0))/(StormThreshold-Windmin))*MvF;
156         if Rand >= MoveChance
157             flag=0;
158         end
159     else
160         flag=0;
161     end
162 end
163 Nhd(4,4)=DirV(1);
164 Nhd(3,5)=DirV(2);
165 Nhd(2,6)=DirV(3);
166 Nhd(1,7)=DirV(4);
167 end
168
169 %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
170
171 %NORTH - EAST BLOCK
172
173 elseif WindDir==2 && rand<MoveChance
174
175 %MOVE N
176 if Rand<((1-P)/2) %&& (i>1)
177     DirV=[Nhd(4,4), Nhd(3,4), Nhd(2,4), Nhd(1,4)];
178     PCNE=[PCNhd(4,4), PCNhd(3,4), PCNhd(2,4), PCNhd(1,4)];
179     flag=1;
180     k=1;
181     while flag==1
182         if atan(((DirV(k+1)-DirV(k))*delta)/L)<(pi/12)
183             DirV(k)=DirV(k)-1;
184             DirV(k+1)=DirV(k+1)+1;
185             cellcnt=cellcnt+1;
186             MoveChance=(1-PCNE(k+1))*((max(Windspeed-Windmin,0))/(StormThreshold-Windmin));
187             k=k+1;
188             flag=1*(k<=dist);
189             MoveChance=(1-PCNE(k))*((max(Windspeed-Windmin,0))/(StormThreshold-Windmin))*MvF;
190             if Rand >= MoveChance
191                 flag=0;
192             end
193         else
194             flag=0;
195         end
196     end
197     Nhd(4,4)=DirV(1);
198     Nhd(3,4)=DirV(2);
199     Nhd(2,4)=DirV(3);
200     Nhd(1,4)=DirV(4);
201
202 %MOVE NE
203 elseif Rand>=(1-P)/2 && Rand<((1+P)/2)% && (i>1) && (j<size(Hstar,2))
204     DirV=[Nhd(4,4), Nhd(3,5), Nhd(2,6), Nhd(1,7)];
205     PCNE=[PCNhd(4,4), PCNhd(3,5), PCNhd(2,6), PCNhd(1,7)];
206     flag=1;

```

```

207         k=1;
208         while flag==1
209             if atan(((DirV(k+1)-DirV(k))*delta)/L)<(pi/12)
210                 DirV(k)=DirV(k)-1;
211                 DirV(k+1)=DirV(k+1)+1;
212             %         cellcnt=cellcnt+1;
213             %         MoveChance=(1-PCNE(k+1))*((max(Windspeed-Windmin,0))/(StormThreshold-Windmin));
214             k=k+1;
215             flag=1*(k<=dist);
216             MoveChance=(1-PCNE(k))*((max(Windspeed-Windmin,0))/(StormThreshold-Windmin))*MvF;
217             if Rand >= MoveChance
218                 flag=0;
219             end
220         else
221             flag=0;
222         end
223     end
224     Nhd(4,4)=DirV(1);
225     Nhd(3,5)=DirV(2);
226     Nhd(2,6)=DirV(3);
227     Nhd(1,7)=DirV(4);
228
229     %MOVE E
230     elseif Rand>=((1+P)/2)% && (j<size(Hstar,2))
231         DirV=[Nhd(4,4), Nhd(4,5), Nhd(4,6), Nhd(4,7)];
232         PCNE=[PCNhd(4,4), PCNhd(4,5), PCNhd(4,6), PCNhd(4,7)];
233         flag=1;
234         k=1;
235         while flag==1
236             if atan(((DirV(k+1)-DirV(k))*delta)/L)<(pi/12)
237                 DirV(k)=DirV(k)-1;
238                 DirV(k+1)=DirV(k+1)+1;
239             %         cellcnt=cellcnt+1;
240             %         MoveChance=(1-PCNE(k+1))*((max(Windspeed-Windmin,0))/(StormThreshold-Windmin));
241             k=k+1;
242             flag=1*(k<=dist);
243             MoveChance=(1-PCNE(k))*((max(Windspeed-Windmin,0))/(StormThreshold-Windmin))*MvF;
244             if Rand >= MoveChance
245                 flag=0;
246             end
247         else
248             flag=0;
249         end
250     end
251     Nhd(4,4)=DirV(1);
252     Nhd(4,5)=DirV(2);
253     Nhd(4,6)=DirV(3);
254     Nhd(4,7)=DirV(4);
255     end
256
257     %
258
259     %EAST BLOCK
260

```

```

261         elseif WindDir==3 && rand<MoveChance
262
263             %MOVES NE
264             if Rand<((1-P)/2)% && (i>1) && (j<size(Hstar,2))
265                 DirV=[Nhd(4,4), Nhd(3,5), Nhd(2,6), Nhd(1,7)];
266                 PCNE=[PCNhd(4,4), PCNhd(3,5), PCNhd(2,6), PCNhd(1,7)];
267                 flag=1;
268                 k=1;
269                 while flag==1
270                     if atan(((DirV(k+1)-DirV(k))*delta)/L)<(pi/12)
271                         DirV(k)=DirV(k)-1;
272                         DirV(k+1)=DirV(k+1)+1;
273                     %
274                     %
275                         cellcnt=cellcnt+1;
276                         MoveChance=(1-PCNE(k+1))*((max(Windspeed-Windmin,0))/(StormThreshold-Windmin));
277                         k=k+1;
278                         flag=1*(k<=dist);
279                         MoveChance=(1-PCNE(k))*((max(Windspeed-Windmin,0))/(StormThreshold-Windmin))*MvF;
280                         if Rand >= MoveChance
281                             flag=0;
282                         end
283                     else
284                         flag=0;
285                     end
286                 end
287                 Nhd(4,4)=DirV(1);
288                 Nhd(3,5)=DirV(2);
289                 Nhd(2,6)=DirV(3);
290                 Nhd(1,7)=DirV(4);
291
292             %MOVES E
293             elseif Rand>=((1-P)/2) && Rand<((1+P)/2) %&& (j<size(Hstar,2))
294                 DirV=[Nhd(4,4), Nhd(4,5), Nhd(4,6), Nhd(4,7)];
295                 PCNE=[PCNhd(4,4), PCNhd(4,5), PCNhd(4,6), PCNhd(4,7)];
296                 flag=1;
297                 k=1;
298                 while flag==1
299                     if atan(((DirV(k+1)-DirV(k))*delta)/L)<(pi/12)
300                         DirV(k)=DirV(k)-1;
301                         DirV(k+1)=DirV(k+1)+1;
302                     %
303                     %
304                         cellcnt=cellcnt+1;
305                         MoveChance=(1-PCNE(k+1))*((max(Windspeed-Windmin,0))/(StormThreshold-Windmin));
306                         k=k+1;
307                         flag=1*(k<=dist);
308                         MoveChance=(1-PCNE(k))*((max(Windspeed-Windmin,0))/(StormThreshold-Windmin))*MvF;
309                         if Rand >= MoveChance
310                             flag=0;
311                         end
312                     else
313                         flag=0;
314                     end
315                 end
316                 Nhd(4,4)=DirV(1);
317                 Nhd(4,5)=DirV(2);
318                 Nhd(4,6)=DirV(3);
319                 Nhd(4,7)=DirV(4);

```



```

317
318 %MOVES SE
319 elseif Rand>=((1+P)/2)% && (i<size(Hstar,1)) && (j<size(Hstar,2))
320 DirV=[Nhd(4,4), Nhd(5,5), Nhd(6,6), Nhd(7,7)];
321 PCNE=[PCNhhd(4,4), PCNhhd(5,5), PCNhhd(6,6), PCNhhd(7,7)];
322 flag=1;
323 k=1;
324 while flag==1
325     if atan(((DirV(k+1)-DirV(k))*delta)/L)<(pi/12)
326         DirV(k)=DirV(k)-1;
327         DirV(k+1)=DirV(k+1)+1;
328 %         cellcnt=cellcnt+1;
329 %         MoveChance=(1-PCNE(k+1))*((max(Windspeed-Windmin,0))/(StormThreshold-Windmin));
330         k=k+1;
331         flag=1*(k<=dist);
332         MoveChance=(1-PCNE(k))*((max(Windspeed-Windmin,0))/(StormThreshold-Windmin))*MvF;
333         if Rand >= MoveChance
334             flag=0;
335         end
336     else
337         flag=0;
338     end
339 end
340 Nhd(4,4)=DirV(1);
341 Nhd(5,5)=DirV(2);
342 Nhd(6,6)=DirV(3);
343 Nhd(7,7)=DirV(4);
344
345
346 end
347
348 %

```

```

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

```

```

349
350 %SOUTH-EAST BLOCK
351
352 elseif WindDir==4 && rand<MoveChance
353
354 %MOVES E
355 if Rand<((1-P)/2)% && (j<size(Hstar,2))
356 DirV=[Nhd(4,4), Nhd(4,5), Nhd(4,6), Nhd(4,7)];
357 PCNE=[PCNhhd(4,4), PCNhhd(4,5), PCNhhd(4,6), PCNhhd(4,7)];
358 flag=1;
359 k=1;
360 while flag==1
361     if atan(((DirV(k+1)-DirV(k))*delta)/L)<(pi/12)
362         DirV(k)=DirV(k)-1;
363         DirV(k+1)=DirV(k+1)+1;
364 %         cellcnt=cellcnt+1;
365 %         MoveChance=(1-PCNE(k+1))*((max(Windspeed-Windmin,0))/(StormThreshold-Windmin));
366         k=k+1;
367         flag=1*(k<=dist);
368         MoveChance=(1-PCNE(k))*((max(Windspeed-Windmin,0))/(StormThreshold-Windmin))*MvF;
369         if Rand >= MoveChance
370             flag=0;

```

```

371         end
372     else
373         flag=0;
374     end
375 end
376 Nhd(4,4)=DirV(1);
377 Nhd(4,5)=DirV(2);
378 Nhd(4,6)=DirV(3);
379 Nhd(4,7)=DirV(4);
380
381
382 %MOVES SE
383 elseif Rand>=((1-P)/2) && Rand<((1+P)/2)% && (i<size(Hstar,1)) && (j<size(Hstar,2))
384     DirV=[Nhd(4,4), Nhd(5,5), Nhd(6,6), Nhd(7,7)];
385     PCNE=[PCNhhd(4,4), PCNhhd(5,5), PCNhhd(6,6), PCNhhd(7,7)];
386     flag=1;
387     k=1;
388     while flag==1
389         if atan(((DirV(k+1)-DirV(k))*delta)/L)<(pi/12)
390             DirV(k)=DirV(k)-1;
391             DirV(k+1)=DirV(k+1)+1;
392             cellcnt=cellcnt+1;
393             %
394             MoveChance=(1-PCNE(k+1))*((max(Windspeed-Windmin,0))/(StormThreshold-Windmin));
395             k=k+1;
396             flag=1*(k<=dist);
397             MoveChance=(1-PCNE(k))*((max(Windspeed-Windmin,0))/(StormThreshold-Windmin))*MvF;
398             if Rand >= MoveChance
399                 flag=0;
400             end
401         else
402             flag=0;
403         end
404     end
405     Nhd(4,4)=DirV(1);
406     Nhd(5,5)=DirV(2);
407     Nhd(6,6)=DirV(3);
408     Nhd(7,7)=DirV(4);
409
410 %MOVES S
411 elseif Rand>=((1+P)/2) && (i<size(Hstar,1))
412     DirV=[Nhd(4,4), Nhd(5,4), Nhd(6,4), Nhd(7,4)];
413     PCNE=[PCNhhd(4,4), PCNhhd(5,4), PCNhhd(6,4), PCNhhd(7,4)];
414     flag=1;
415     k=1;
416     while flag==1
417         if atan(((DirV(k+1)-DirV(k))*delta)/L)<(pi/12)
418             DirV(k)=DirV(k)-1;
419             DirV(k+1)=DirV(k+1)+1;
420             cellcnt=cellcnt+1;
421             %
422             MoveChance=(1-PCNE(k+1))*((max(Windspeed-Windmin,0))/(StormThreshold-Windmin));
423             k=k+1;
424             flag=1*(k<=dist);
425             MoveChance=(1-PCNE(k))*((max(Windspeed-Windmin,0))/(StormThreshold-Windmin))*MvF;
426             if Rand >= MoveChance
427                 flag=0;
428             end
429         else
430             flag=0;
431         end
432     end

```

```

427         else
428             flag=0;
429         end
430     end
431     Nhd(4,4)=DirV(1);
432     Nhd(5,4)=DirV(2);
433     Nhd(6,4)=DirV(3);
434     Nhd(7,4)=DirV(4);
435
436     end
437
438 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
439
440 %SOUTH BLOCK
441
442 elseif WindDir==5 && rand<MoveChance
443
444     %MOVES SE
445     if Rand<((1-P)/2) %&& (i<size(Hstar,1) && (j<size(Hstar,2)))
446         DirV=[Nhd(4,4), Nhd(5,5), Nhd(6,6), Nhd(7,7)];
447         PCNE=[PCNhd(4,4), PCNhd(5,5), PCNhd(6,6), PCNhd(7,7)];
448         flag=1;
449         k=1;
450         while flag==1
451             if atan(((DirV(k+1)-DirV(k))*delta)/L)<(pi/12)
452                 DirV(k)=DirV(k)-1;
453                 DirV(k+1)=DirV(k+1)+1;
454                 cellcnt=cellcnt+1;
455                 MoveChance=(1-PCNE(k))*((max(Windspeed-Windmin,0))/(StormThreshold-Windmin));
456                 k=k+1;
457                 flag=1*(k<=dist);
458                 MoveChance=(1-PCNE(k))*((max(Windspeed-Windmin,0))/(StormThreshold-Windmin))*MvF;
459                 if Rand >= MoveChance
460                     flag=0;
461                 end
462             else
463                 flag=0;
464             end
465         end
466         Nhd(4,4)=DirV(1);
467         Nhd(5,5)=DirV(2);
468         Nhd(6,6)=DirV(3);
469         Nhd(7,7)=DirV(4);
470
471     %MOVES S
472     elseif Rand>=(1-P)/2 && Rand<((1+P)/2) %&& (i<size(Hstar,1))
473         DirV=[Nhd(4,4), Nhd(5,4), Nhd(6,4), Nhd(7,4)];
474         PCNE=[PCNhd(4,4), PCNhd(5,4), PCNhd(6,4), PCNhd(7,4)];
475         flag=1;
476         k=1;
477         while flag==1
478             if atan(((DirV(k+1)-DirV(k))*delta)/L)<(pi/12)
479                 DirV(k)=DirV(k)-1;
480                 DirV(k+1)=DirV(k+1)+1;

```

```

481 %           cellcnt=cellcnt+1;
482 %           MoveChance=(1-PCNE(k+1))*((max(Windspeed-Windmin,0))/(StormThreshold-Windmin));
483           k=k+1;
484           flag=1*(k<=dist);
485           MoveChance=(1-PCNE(k))*((max(Windspeed-Windmin,0))/(StormThreshold-Windmin))*MvF;
486           if Rand >= MoveChance
487               flag=0;
488           end
489       else
490           flag=0;
491       end
492   end
493   Nhd(4,4)=DirV(1);
494   Nhd(5,4)=DirV(2);
495   Nhd(6,4)=DirV(3);
496   Nhd(7,4)=DirV(4);
497
498   %MOVES SW
499   elseif Rand>=((1+P)/2) %&& (i<size(Hstar,1)) %&& (j>1)
500       DirV=[Nhd(4,4), Nhd(5,3), Nhd(6,2), Nhd(7,1)];
501       PCNE=[PCNhd(4,4), PCNhd(5,3), PCNhd(6,2), PCNhd(7,1)];
502       flag=1;
503       k=1;
504       while flag==1
505           if atan(((DirV(k+1)-DirV(k))*delta)/L)<(pi/12)
506               DirV(k)=DirV(k)-1;
507               DirV(k+1)=DirV(k+1)+1;
508 %           cellcnt=cellcnt+1;
509 %           MoveChance=(1-PCNE(k+1))*((max(Windspeed-Windmin,0))/(StormThreshold-Windmin));
510           k=k+1;
511           flag=1*(k<=dist);
512           MoveChance=(1-PCNE(k))*((max(Windspeed-Windmin,0))/(StormThreshold-Windmin))*MvF;
513           if Rand >= MoveChance
514               flag=0;
515           end
516       else
517           flag=0;
518       end
519   end
520   Nhd(4,4)=DirV(1);
521   Nhd(5,3)=DirV(2);
522   Nhd(6,2)=DirV(3);
523   Nhd(7,1)=DirV(4);
524
525
526 %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
527
528 %SOUTH-WEST BLOCK
529
530 elseif WindDir==6 && rand<MoveChance
531
532 %MOVES S
533 if Rand<((1-P)/2) %&& (i<size(Hstar,1))
534     DirV=[Nhd(4,4), Nhd(5,4), Nhd(6,4), Nhd(7,4)];

```

```

535 PCNE=[PCNhhd(4,4), PCNhhd(5,4), PCNhhd(6,4), PCNhhd(7,4)];
536 flag=1;
537 k=1;
538 while flag==1
539     if atan(((DirV(k+1)-DirV(k))*delta)/L)<(pi/12)
540         DirV(k)=DirV(k)-1;
541         DirV(k+1)=DirV(k+1)+1;
542     %         cellcnt=cellcnt+1;
543     %         MoveChance=(1-PCNE(k+1))*((max(Windspeed-Windmin,0))/(StormThreshold-Windmin));
544         k=k+1;
545         flag=1*(k<=dist);
546         MoveChance=(1-PCNE(k))*((max(Windspeed-Windmin,0))/(StormThreshold-Windmin))*MvF;
547         if Rand >= MoveChance
548             flag=0;
549         end
550     else
551         flag=0;
552     end
553 end
554 Nhd(4,4)=DirV(1);
555 Nhd(5,4)=DirV(2);
556 Nhd(6,4)=DirV(3);
557 Nhd(7,4)=DirV(4);
558
559
560 %MOVES SW
561 elseif Rand>=((1-P)/2) && Rand<((1+P)/2) %&& (i<size(Hstar,1)) && (j>1)
562     DirV=[Nhd(4,4), Nhd(5,3), Nhd(6,2), Nhd(7,1)];
563     PCNE=[PCNhhd(4,4), PCNhhd(5,3), PCNhhd(6,2), PCNhhd(7,1)];
564     flag=1;
565     k=1;
566     while flag==1
567         if atan(((DirV(k+1)-DirV(k))*delta)/L)<(pi/12)
568             DirV(k)=DirV(k)-1;
569             DirV(k+1)=DirV(k+1)+1;
570         %         cellcnt=cellcnt+1;
571         %         MoveChance=(1-PCNE(k+1))*((max(Windspeed-Windmin,0))/(StormThreshold-Windmin));
572             k=k+1;
573             flag=1*(k<=dist);
574             MoveChance=(1-PCNE(k))*((max(Windspeed-Windmin,0))/(StormThreshold-Windmin))*MvF;
575             if Rand >= MoveChance
576                 flag=0;
577             end
578         else
579             flag=0;
580         end
581     end
582     Nhd(4,4)=DirV(1);
583     Nhd(5,3)=DirV(2);
584     Nhd(6,2)=DirV(3);
585     Nhd(7,1)=DirV(4);
586
587
588 %MOVES W
589 elseif Rand>=((1+P)/2) %&& (j>1)
590     DirV=[Nhd(4,4), Nhd(4,3), Nhd(4,2), Nhd(4,1)];

```

```

591     PCNE=[PCNhd(4,4), PCNhd(4,3), PCNhd(4,2), PCNhd(4,1)];
592     flag=1;
593     k=1;
594     while flag==1
595         if atan(((DirV(k+1)-DirV(k))*delta)/L)<(pi/12)
596             DirV(k)=DirV(k)-1;
597             DirV(k+1)=DirV(k+1)+1;
598             cellcnt=cellcnt+1;
599             MoveChance=(1-PCNE(k+1))*((max(Windspeed-Windmin,0))/(StormThreshold-Windmin));
600             k=k+1;
601             flag=1*(k<=dist);
602             MoveChance=(1-PCNE(k))*((max(Windspeed-Windmin,0))/(StormThreshold-Windmin))*MvF;
603             if Rand >= MoveChance
604                 flag=0;
605             end
606         else
607             flag=0;
608         end
609     end
610     Nhd(4,4)=DirV(1);
611     Nhd(4,3)=DirV(2);
612     Nhd(4,2)=DirV(3);
613     Nhd(4,1)=DirV(4);
614
615     end
616
617     %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
618
619     %WEST BLOCK
620
621     elseif WindDir==7 && rand<MoveChance
622
623     %MOVES SW
624     if Rand<((1-P)/2) && (i<size(Hstar,1)) && (j>1)
625         DirV=[Nhd(4,4), Nhd(5,3), Nhd(6,2), Nhd(7,1)];
626         PCNE=[PCNhd(4,4), PCNhd(5,3), PCNhd(6,2), PCNhd(7,1)];
627         flag=1;
628         k=1;
629         while flag==1
630             if atan(((DirV(k+1)-DirV(k))*delta)/L)<(pi/12)
631                 DirV(k)=DirV(k)-1;
632                 DirV(k+1)=DirV(k+1)+1;
633                 cellcnt=cellcnt+1;
634                 MoveChance=(1-PCNE(k+1))*((max(Windspeed-Windmin,0))/(StormThreshold-Windmin));
635                 k=k+1;
636                 flag=1*(k<=dist);
637                 MoveChance=(1-PCNE(k))*((max(Windspeed-Windmin,0))/(StormThreshold-Windmin))*MvF;
638                 if Rand >= MoveChance
639                     flag=0;
640                 end
641             else
642                 flag=0;
643             end
644         end

```

```

645         Nhd(4,4)=DirV(1);
646         Nhd(5,3)=DirV(2);
647         Nhd(6,2)=DirV(3);
648         Nhd(7,1)=DirV(4);
649
650         %MOVES W
651     elseif Rand>=((1-P)/2) && Rand<((1+P)/2) %&& (j>1)
652         DirV=[Nhd(4,4), Nhd(4,3), Nhd(4,2), Nhd(4,1)];
653         PCNE=[PCNhd(4,4), PCNhd(4,3), PCNhd(4,2), PCNhd(4,1)];
654         flag=1;
655         k=1;
656         while flag==1
657             if atan(((DirV(k+1)-DirV(k))*delta)/L)<(pi/12)
658                 DirV(k)=DirV(k)-1;
659                 DirV(k+1)=DirV(k+1)+1;
660                 cellcnt=cellcnt+1;
661                 %
662                 MoveChance=(1-PCNE(k+1))*((max(Windspeed-Windmin,0))/(StormThreshold-Windmin));
663                 k=k+1;
664                 flag=1*(k<=dist);
665                 MoveChance=(1-PCNE(k))*((max(Windspeed-Windmin,0))/(StormThreshold-Windmin))*MvF;
666                 if Rand >= MoveChance
667                     flag=0;
668                 end
669             else
670                 flag=0;
671             end
672         end
673         Nhd(4,4)=DirV(1);
674         Nhd(4,3)=DirV(2);
675         Nhd(4,2)=DirV(3);
676         Nhd(4,1)=DirV(4);
677
678         %MOVES NW
679     elseif Rand>=((1+P)/2) %&& (i>1) && (j>1)
680         DirV=[Nhd(4,4), Nhd(3,3), Nhd(2,2), Nhd(2,1)];
681         PCNE=[PCNhd(4,4), PCNhd(3,3), PCNhd(2,2), PCNhd(1,1)];
682         flag=1;
683         k=1;
684         while flag==1
685             if atan(((DirV(k+1)-DirV(k))*delta)/L)<(pi/12)
686                 DirV(k)=DirV(k)-1;
687                 DirV(k+1)=DirV(k+1)+1;
688                 cellcnt=cellcnt+1;
689                 %
690                 MoveChance=(1-PCNE(k+1))*((max(Windspeed-Windmin,0))/(StormThreshold-Windmin));
691                 k=k+1;
692                 flag=1*(k<=dist);
693                 MoveChance=(1-PCNE(k))*((max(Windspeed-Windmin,0))/(StormThreshold-Windmin))*MvF;
694                 if Rand >= MoveChance
695                     flag=0;
696                 end
697             else
698                 flag=0;
699             end
700         end
701         Nhd(4,4)=DirV(1);
702         Nhd(3,3)=DirV(2);

```

```

701         Nhd(2,2)=DirV(3);
702         Nhd(1,1)=DirV(4);
703
704     end
705
706 %
707
708     %NORTH-WEST
709
710     elseif WindDir==8 && rand<MoveChance
711
712         %MOVES W
713         if Rand<((1-P)/2) %&& (j>1)
714             DirV=[Nhd(4,4), Nhd(4,3), Nhd(4,2), Nhd(4,1)];
715             PCNE=[PCNhd(4,4), PCNhd(4,3), PCNhd(4,2), PCNhd(4,1)];
716             flag=1;
717             k=1;
718             while flag==1
719                 if atan(((DirV(k+1)-DirV(k))*delta)/L)<(pi/12)
720                     DirV(k)=DirV(k)-1;
721                     DirV(k+1)=DirV(k+1)+1;
722                     cellcnt=cellcnt+1;
723                     MoveChance=(1-PCNE(k+1))*((max(Windspeed-Windmin,0))/(StormThreshold-Windmin));
724                     k=k+1;
725                     flag=1*(k<=dist);
726                     MoveChance=(1-PCNE(k))*((max(Windspeed-Windmin,0))/(StormThreshold-Windmin))*MvF;
727                     if Rand >= MoveChance
728                         flag=0;
729                     end
730                 else
731                     flag=0;
732                 end
733             end
734             Nhd(4,4)=DirV(1);
735             Nhd(4,3)=DirV(2);
736             Nhd(4,2)=DirV(3);
737             Nhd(4,1)=DirV(4);
738
739             %MOVES NW
740             elseif Rand>=((1-P)/2) && Rand<((1+P)/2) %&& (i>1) && (j>1)
741                 DirV=[Nhd(4,4), Nhd(3,3), Nhd(2,2), Nhd(1,1)];
742                 PCNE=[PCNhd(4,4), PCNhd(3,3), PCNhd(2,2), PCNhd(1,1)];
743                 flag=1;
744                 k=1;
745                 while flag==1
746                     if atan(((DirV(k+1)-DirV(k))*delta)/L)<(pi/12)
747                         DirV(k)=DirV(k)-1;
748                         DirV(k+1)=DirV(k+1)+1;
749                         cellcnt=cellcnt+1;
750                         MoveChance=(1-PCNE(k+1))*((max(Windspeed-Windmin,0))/(StormThreshold-Windmin));
751                         k=k+1;
752                         flag=1*(k<=dist);
753                         MoveChance=(1-PCNE(k))*((max(Windspeed-Windmin,0))/(StormThreshold-Windmin))*MvF;
754                         if Rand >= MoveChance

```



```

755         flag=0;
756     end
757     else
758         flag=0;
759     end
760 end
761 Nhd(4,4)=DirV(1);
762 Nhd(3,3)=DirV(2);
763 Nhd(2,2)=DirV(3);
764 Nhd(1,1)=DirV(4);
765
766 %MOVES N
767 elseif Rand>=((1+P)/2) %&& (i>1)
768     DirV=[Nhd(4,4), Nhd(3,4), Nhd(2,4), Nhd(1,4)];
769     PCNE=[PCNhd(4,4), PCNhd(3,4), PCNhd(2,4), PCNhd(1,4)];
770     flag=1;
771     k=1;
772     while flag==1
773         if atan(((DirV(k+1)-DirV(k))*delta)/L)<(pi/12)
774             DirV(k)=DirV(k)-1;
775             DirV(k+1)=DirV(k+1)+1;
776             cellcnt=cellcnt+1;
777             MoveChance=(1-PCNE(k+1))*((max(Windspeed-Windmin,0))/(StormThreshold-Windmin));
778             k=k+1;
779             flag=1*(k<=dist);
780             MoveChance=(1-PCNE(k))*((max(Windspeed-Windmin,0))/(StormThreshold-Windmin))*MvF;
781             if Rand >= MoveChance
782                 flag=0;
783             end
784         else
785             flag=0;
786         end
787     end
788     Nhd(4,4)=DirV(1);
789     Nhd(3,4)=DirV(2);
790     Nhd(2,4)=DirV(3);
791     Nhd(1,4)=DirV(4);
792 %
793 %     flag=1;
794 %     cnt=0;
795 %     k=1;
796 %     while flag==1
797 %         if atan(((Nhd(4-k,4)-Nhd(4-k+1,4))*delta)/L)<(pi/12)
798 %             Nhd(4-k+1,4)=Nhd(4-k+1,4)-1;%4-k+1=4-(k-1)
799 %             Nhd(4-k,4)=Nhd(4-k,4)+1;
800 %             MoveChance=(1-PCNhd(4-k,4))*((max(Windspeed-Windmin,0))/(StormThreshold-Windmin));%(
calculating, but not using yet)
801 %             k=k+1;
802 %             cnt=cnt+1;
803 %             flag=1*(k<=dist);%(Nhd(4,4)>=0)
804 %         else
805 %             flag=0;
806 %         end
807 %     end
808 end
809 end

```

```

810             end
811             Hp(i-3:i+3,j-3:j+3,1)=Nhd;
812         end
813     else
814     end
815 end
816 end
817
818 Hstar(:,1)=Hp(4:n1+3,4:n2+3);
819
820 end %end of AT function
821
822 %COMMENT: Old way of polling may have made it easier to change number of
823 %steps in each direction that a cell can possible move:
824
825 %it used this neighborhood function to declare a local region of 3 cells in
826 %each direction. If we used a variablelike D=#cells wished to move
827
828 %then ...
829
830 %
831 % function Nl=NewNhd(R,C,H)
832 % if R<=(size(H,1)-1)
833 %     R=floor(R);
834 %     Rp=R+1;
835 % else
836 %     Rp=floor(R);
837 % end
838 % if R>=2
839 %     R=floor(R);
840 %     Rm=R-1;
841 % else
842 %     Rm=floor(R);
843 % end
844 % if C<=(size(H,2)-1)
845 %     C=floor(C);
846 %     Cp=C+1;
847 % else
848 %     Cp=floor(C);
849 % end
850 % if C>=2
851 %     C=floor(C);
852 %     Cm=C-1;
853 % else
854 %     Cm=floor(C);
855 % end
856 % Nl=zeros(3,3);           %<-----CHANGED THIS TO zeros(D,D) *****
857
858 % Nl=[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)];
859 %
860 % end
861
862 % *****and used the old method of checking:
863
864 % *****All of these 4's would need to be replaced with D+1
865

```

```

866 %         flag=1;
867 %         cnt=0;
868 %         k=1;
869 %         while flag==1
870 %             if atan(((Nhd(4-k,4)-Nhd(4-k+1,4))*delta)/L)<(pi/12)
871 %                 Nhd(4-k+1,4)=Nhd(4-k+1,4)-1;%4-k+1=4-(k-1)
872 %                 Nhd(4-k,4)=Nhd(4-k,4)+1;
873 %                 MoveChance=(1-PCNhd(4-k,4))*((max(Windspeed-Windmin,0))/(StormThreshold-Windmin));%(
calculating, but not using yet)
874 %                 k=k+1;
875 %                 cnt=cnt+1;
876 %                 flag=1*(k<=dist);%*(Nhd(4,4)>=0)
877 %             else
878 %                 flag=0;
879 %             end
880 %         end

```

B.3 Avalanche code

```
1 function [Hstar,flag,CellCt] = AVALANCHEtime03312021(Hstar,delta,L,flag,PC,t)
2 %% This version of avalanche:
3 % ~weights the probability of avalanching to favor the direction of the
4 % greatest violation of the angle of repose.
5 % ~takes t as input to track if avalanching is required over whole domain
6 % (once per 5 years)
7 % ~most recent edit code was ATTEMPT2newAVALANCHEtime0928
8 %
9 AoR=pi/6;
10 % AoR=pi/9;
11
12 %%
13 % tic
14
15 Hstarflag=Hstar;
16 CellCt=0;
17 flag=0;
18 FLAG=1;
19 alpha=1;
20 % fprintf('I am running AV! ')
21 Nhd=4; %size of neighborhood check (4 or 8)
22 beta1=zeros(1,Nhd); %angle of repose
23 beta2=zeros(1,Nhd); %
24 check1=zeros(1,Nhd);
25 check2=zeros(1,Nhd);
26
27 %once per 5 year we check every cell in domain, both above and below sea level
28
29 if (0==mod(t,26)) || exist('t','var')==0
30 for R=1:size(Hstar,1)
31 for C=1:size(Hstar,2)
32 Erosioncheck=1;
33 FLAG=1;
34 while FLAG==1
35 RAND=rand;
36 %FLAG=1;
37 %if Hstar(R,C)>0
38 if R<=(size(Hstar,1)-1)
39 R=floor(R);
40 Rp=R+1;
41 else
42 Rp=floor(R);
43 end
44 if R>=2
45 R=floor(R);
46 Rn=R-1;
47 else
48 Rn=floor(R);
49 end
50 if C<=(size(Hstar,2)-1)
51 C=floor(C);
52 Cp=C+1;
53 else
54 Cp=floor(C);
```

```

55     end
56     if C>=2
57         C=floor(C);
58         Cm=C-1;
59     else
60         Cm=floor(C);
61     end
62
63     Hx=Hstar(R,C); %current cell value
64     N=[Hstar(Rm,C),Hstar(R,Cp),Hstar(Rp,C),Hstar(R,Cm)]; %neighborhood
65
66     beta1=atan(((Hx-N)*delta)/L);
67     check1=(beta1>=(AoR));
68     %Amdir=find(beta1>=AoR);
69     beta2=alpha*((beta1/(AoR))*(1-PC(R,C)));
70     check2=(RAND<beta2);
71
72
73     if sum(check2)~=0
74         while sum(check1)~=0 && sum(check2)~=0 %if I use a while loop, there's no point to using probability
75             %posbeta1=beta1;
76             %posbeta1(find(beta1<0))=0;
77             %Nprob=posbeta1/sum(posbeta1);
78             AVbeta1=zeros(1,4);
79             AVbeta1(check1)=beta1(check1);
80             AVbeta1(check2)=AVbeta1(check2);
81             Nprob=AVbeta1/sum(AVbeta1);
82             prob=rand;
83             Hcheck=Hx;
84             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));
85             % if Hx-Hcheck==0
86             %     FLAG=0;
87             %     break
88             % end
89             N(1)=N(1)+check1(1)+check2(1)*(prob<Nprob(1));
90             N(2)=N(2)+check1(2)+check2(2)*(prob<Nprob(2));
91             N(3)=N(3)+check1(3)+check2(3)*(prob<Nprob(3));
92             N(4)=N(4)+check1(4)+check2(4)*(prob<Nprob(4));
93             beta1=newbeta1(N,Hx);
94             beta2=newbeta2(N,beta1);
95             check1=newcheck1(N,beta1);
96             check2=newcheck2(N,beta2);
97             %flag=1;
98             if Hx-Hcheck~=0
99                 CellCt=CellCt+1;
100                flag=1;
101            end
102            if sum(check1)==0
103                Hstar(R,C)=Hx;
104                Hstar(Rm,C)=N(1);
105                Hstar(R,Cp)=N(2);
106                Hstar(Rp,C)=N(3);
107                Hstar(R,Cm)=N(4);
108                %fprintf('Changes have been made')
109                FLAG=0;

```

```

110             if Hx >=0
111                 break
112             end
113         end
114         if check1+check2'==0
115             break
116         end
117     end
118     if sum(check1)==0
119         FLAG=0;
120         %no avalanching needed
121     end
122     elseif sum(check2)==0 && sum(check1)~=0
123         FLAG=0;
124         %fprintf('Too many plants!')
125     elseif sum(check1)==0
126         FLAG=0; %only other case should be if both sum to 0
127     end
128
129 %     else%{if Hstar(R,C)<=0}
130 %         Erosioncheck=0;
131 %         FLAG=0;
132 %     end
133     FLAG=0;
134 end
135 end
136 end
137 %for other instances of avalanche we only check subaerial portions of island
138 else
139 %     ISLANDindex=Hstar(:, : ,1)>0; %all cells with land
140 %     Hstarflag=cumsum(ISLANDindex,2) == 1 & ISLANDindex; %this outputs an array same size as H but with a 1 in the first cell>0
141 %     IslandCheck=sum(Hstarflag '); %outputs a row vector with 0^ no col>0, 1^col>0 (ie yes/no land in that row)
142 %     COLindex1 = Hstarflag*(1:size(Hstar,2))'; %the index of the first positive value in a row
143 %     COLindex2 = zeros(size(COLindex1));
144 %     for ii=1:length(COLindex1)
145 %         if ii == 101
146 %             fprintf('yay')
147 %         end
148 %
149 %         if COLindex1(ii)~=0
150 %             RowNow=ISLANDindex(ii, :);
151 %             Icount=0;
152 %             for jj = COLindex1(ii):1:size(RowNow)
153 %                 if RowNow(jj)==1 && RowNow(jj+1)==0
154 %                     Icount=Icount+1;
155 %                     COLindex2(ii, Icount)=jj;
156 %                 end
157 %             end
158 %         end
159 %     end
160     IslandColumnArray1=zeros([size(Hstar,1) 1]);
161
162     for i=1:size(Hstar,1)
163         ElevCheck1=Hstar(i, : ,1)>=0; %first cell above water
164         Icount=0;%
165         for j=2:size(Hstar,2)

```

```

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

           %
169         %break;

           %
170     end

           %
171     end

           %
172 end
173 INum=size(IslandColumnArray1,2);
174 IslandColumnArray2=zeros(size(IslandColumnArray1(:,:)));
175 for i=1:size(Hstar,1)
176     ElevCheck2=Hstar(i,:,1)<=0;
177     if sum(IslandColumnArray1(i,1:INum)~=0)>0
178         for Icount=1:INum
179             if IslandColumnArray1(i,Icount)~=0
180                 for j=IslandColumnArray1(i,Icount):size(Hstar,2)
181                     if ElevCheck2(j)==1 && IslandColumnArray2(i,Icount)==0
182                         IslandColumnArray2(i,Icount)=j;
183                         break;
184                     %             elseif sum(ElevCheck2)==0
185                     %                 ColumnArraySwamp1(i,Icount)=0;
186                 end
187             end
188         end
189     end
190 end
191 end
192
193 for R=1:size(Hstar,1)
194     for Icount=1:INum
195         if IslandColumnArray2(R,Icount)~=0
196             for C=IslandColumnArray1(R,Icount):IslandColumnArray2(R,Icount)
197                 Erosioncheck=1;
198                 FLAG=1;
199                 while FLAG==1
200                     RAND=rand;
201                     %FLAG=1;
202                     if Hstar(R,C)>0
203                         if R<=(size(Hstar,1)-1)
204                             R=floor(R);
205                             Rp=R+1;
206                         else
207                             Rp=floor(R);
208                         end
209                         if R>=2
210                             R=floor(R);
211                             Rm=R-1;
212                         else

```

```

213         Rm=floor(R);
214     end
215     if C<=(size(Hstar,2)-1)
216         C=floor(C);
217         Cp=C+1;
218     else
219         Cp=floor(C);
220     end
221     if C>=2
222         C=floor(C);
223         Cm=C-1;
224     else
225         Cm=floor(C);
226     end
227
228     Hx=Hstar(R,C); %current cell value
229     N=[Hstar(Rm,C),Hstar(R,Cp),Hstar(Rp,C),Hstar(R,Cm)]; %neighborhood
230
231     beta1=atan(((Hx-N)*delta)/L);
232     check1=(beta1>=AoR);
233     %Amdir=find(beta1>=AoR);
234     beta2=alpha*((beta1/AoR)*(1-PC(R,C)));
235     check2=(RAND<beta2);
236
237
238     %         if sum(check1)~=0
239     %             if sum(check2)~=0
240     %                 end
241     %             end
242     %         if R==241
243     %             if C==492
244 %             fprintf('STOP!')
245     %             end
246     %         end
247
248     if sum(check2)~=0
249         if check1*check2~=0
250             while sum(check1)~=0 && sum(check2)~=0 %if I use a while loop, there's no point to
                using probability
251                 %posbeta1=beta1;
252                 %posbeta1(find(beta1<0))=0;
253                 %Nprob=posbeta1/sum(posbeta1);
254                 AVbeta1=zeros(1,4);
255                 AVbeta1(check1)=beta1(check1);
256                 AVbeta1(check2)=AVbeta1(check2);
257                 Nprob=AVbeta1/sum(AVbeta1);
258                 prob=rand;
259                 Hcheck=Hx;
260                 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))
                    ;
261                 %             if Hx-Hcheck==0
262                 %                 FLAG=0;
263                 %                 break
264                 %             end
265                 N(1)=N(1)+check1(1)*check2(1)*(prob<Nprob(1));

```



```

266 N(2)=N(2)+check1(2)*check2(2)*(prob<Nprob(2));
267 N(3)=N(3)+check1(3)*check2(3)*(prob<Nprob(3));
268 N(4)=N(4)+check1(4)*check2(4)*(prob<Nprob(4));
269 beta1=newbeta1(N,Hx);
270 beta2=newbeta2(N,beta1);
271 check1=newcheck1(N,beta1);
272 check2=newcheck2(N,beta2);
273 %flag =1;
274 if Hx-Hcheck~=0
275     CellCt=CellCt+1;
276     flag =1;
277 end
278 if sum(check1)==0
279     Hstar(R,C)=Hx;
280     Hstar(Rm,C)=N(1);
281     Hstar(R,Cp)=N(2);
282     Hstar(Rp,C)=N(3);
283     Hstar(R,Cm)=N(4);
284     %fprintf('Changes have been made')
285     FLAG=0;
286     if Hx >=0
287         break
288     end
289 end
290 if check1*check2'==0
291     break
292 end
293 end
294 end
295 if sum(check1)==0
296     FLAG=0;
297     %no avalanching needed
298 end
299 elseif sum(check2)==0 && sum(check1)~=0
300     FLAG=0;
301     %fprintf('Too many plants!')
302 elseif sum(check1)==0
303     FLAG=0;    %only other case should be if both sum to 0
304 end
305
306 else%{ if Hstar(R,C)<=0}
307     Erosioncheck=0;
308     FLAG=0;
309 end
310 FLAG=0;
311 end
312 end
313 end
314 end
315
316 end
317 end
318 % end
319
320 %this portion is for elevation maps with multiple full sized islands within the domain
321 %we rarely need this, but it's here just in case

```

```

322
323 % for R=1:size(Hstar,1)
324 %     %ISLANDindex=(Hstar(R,:,1)>0);
325 %     if sum(ISLANDindex)~=0 %skip whole rows of water
326 %         for C=1:size(Hstar,2)
327 %             Erosioncheck=1;
328 %             FLAG=1;
329 %             while FLAG==1
330 %                 RAND=rand;
331 %                 %FLAG=1;
332 %                 if Hstar(R,C)>0
333 %                     if R<=(size(Hstar,1)-1)
334 %                         R=floor(R);
335 %                         Rp=R+1;
336 %                     else
337 %                         Rp=floor(R);
338 %                     end
339 %                     if R>=2
340 %                         R=floor(R);
341 %                         Rm=R-1;
342 %                     else
343 %                         Rm=floor(R);
344 %                     end
345 %                     if C<=(size(Hstar,2)-1)
346 %                         C=floor(C);
347 %                         Cp=C+1;
348 %                     else
349 %                         Cp=floor(C);
350 %                     end
351 %                     if C>=2
352 %                         C=floor(C);
353 %                         Cm=C-1;
354 %                     else
355 %                         Cm=floor(C);
356 %                     end
357 %
358 %                     Hx=Hstar(R,C); %current cell value
359 %                     N=[Hstar(Rm,C),Hstar(R,Cp),Hstar(Rp,C),Hstar(R,Cm)]; %neighborhood
360 %
361 %                     beta1=atan(((Hx-N)+delta)/L);
362 %                     check1=(beta1>=(AoR));
363 %                     %Amdir=find(beta1>=AoR);
364 %                     beta2=alpha+((beta1/(AoR))*(1-PC(R,C)));
365 %                     check2=(RAND<beta2);
366 %
367 %
368 % %         if sum(check1)~=0
369 % %             if sum(check2)~=0
370 % %                 end
371 % %             end
372 % %             if R==24
373 % %                 fprintf('STOP!')
374 % %             end
375 %
376 %         if sum(check2)~=0
377 %             if check1*check2'~=0

```

```

378 %           while sum(check1)~=0 && sum(check2)~=0 %if I use a while loop, there's no point to using
probability
379 %           %posbeta1=beta1;
380 %           %posbeta1(find(beta1<0))=0;
381 %           %Nprob=posbeta1/sum(posbeta1);
382 %           AVbeta1=zeros(1,4);
383 %           AVbeta1(check1)=beta1(check1);
384 %           AVbeta1(check2)=AVbeta1(check2);
385 %           Nprob=AVbeta1/sum(AVbeta1);
386 %           prob=rand;
387 %           Hcheck=Hx;
388 %           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));
389 %           %           if Hx-Hcheck==0
390 %           %           FLAG=0;
391 %           %           break
392 %           %           end
393 %           N(1)=N(1)+check1(1)*check2(1)*(prob<Nprob(1));
394 %           N(2)=N(2)+check1(2)*check2(2)*(prob<Nprob(2));
395 %           N(3)=N(3)+check1(3)*check2(3)*(prob<Nprob(3));
396 %           N(4)=N(4)+check1(4)*check2(4)*(prob<Nprob(4));
397 %           beta1=newbeta1(N,Hx);
398 %           beta2=newbeta2(N,beta1);
399 %           check1=newcheck1(N,beta1);
400 %           check2=newcheck2(N,beta2);
401 %           %flag=1;
402 %           if Hx-Hcheck~=0
403 %           flag=1;
404 %           end
405 %           if sum(check1)==0
406 %           Hstar(R,C)=Hx;
407 %           Hstar(Rm,C)=N(1);
408 %           Hstar(R,Cp)=N(2);
409 %           Hstar(Rp,C)=N(3);
410 %           Hstar(R,Cm)=N(4);
411 %           %fprintf('Changes have been made')
412 %           FLAG=0;
413 %           if Hx >=0
414 %           break
415 %           end
416 %           end
417 %           if check1*check2'==0
418 %           break
419 %           end
420 %           end
421 %           end
422 %           if sum(check1)==0
423 %           FLAG=0;
424 %           %no avalanching needed
425 %           end
426 %           elseif sum(check2)==0 && sum(check1)~=0
427 %           FLAG=0;
428 %           %fprintf('Too many plants!')
429 %           elseif sum(check1)==0
430 %           FLAG=0; %only other case should be if both sum to 0
431 %           end

```

```

432 %
433 %           else%{if Hstar(R,C)<=0}
434 %               Erosioncheck=0;
435 %               FLAG=0;
436 %           end
437 %           FLAG=0;
438 %       end
439 %   end
440 % end
441 %
442 % end
443 % end
444
445
446 %Local functions for finding angle of repose and probability WRT plant cover
447 function B1=newbeta1(N,Hx)
448     %B1=zeros(1,length(N));
449     B1=atan(((Hx-N)*delta)/L);
450
451
452 %     for k=1:length(N)
453 %         B1(k)=atan(((Hx-N(k))*delta)/L);
454 %     end
455 end
456
457 function B2=newbeta2(N,beta1)
458     %B2=zeros(1:length(N));
459     B2=alpha*((beta1/(AoR))*(1-PC(R,C)));
460
461 %     for k=1:length(N)
462 %         B2(k)=alpha*((beta1(k)/(AoR))*(1-PC(R,C)));
463 %     end
464 end
465
466 function C1=newcheck1(N,beta1)
467     %C1=zeros(1,length(N));
468     C1=(beta1>=(AoR));
469 %     for k=1:length(N)
470 %         C1(k)=(beta1(k)>=(AoR));
471 %     end
472 end
473
474 function C2=newcheck2(N,beta2)
475     %C2=zeros(1,length(N));
476     C2=(RAND<beta2);
477 %     for k=1:length(N)
478 %         C2(k)=(RAND<beta2(k));
479 %     end
480 end
481
482 %FLAGCHECK=Hstarflag-Hstar;
483 %if sum(sum(Hstarflag-Hstar))~=0
484     %flag=1;
485 %end
486
487

```

488 % toc
489 end

B.4 Plant Propagation code

```
1 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)
2 % function[P1,P2,P3,P4,P3d]=PlantPropagation03312021(Hstar,P1,P2,P3,P4,W,S,delta,P1Burial,P2Burial,P3Burial,P4Burial,P3d,
   MaxSwampWidth,t)
3
4 test=zeros(size(P1,1),size(P2,2));
5 clims=[0 0];
6
7 P1B4=P1(:, :, 1);
8 P2B4=P2(:, :, 1);
9 P3B4=P3(:, :, 1);
10 P4B4=P4(:, :, 1);
11
12 % first loop does all propagating and death of all populations,
13 % second loop is death by competition for cells > MasterMAX
14
15
16 % To run this file you will need to specify:
17 % H - elevation matrix that is continually used in main code
18 % P1 - the plant matrix for Ammophila (GRASS)
19 % P2 - the plant matrix for Spartina (GRASS)
20 % P3 - the plant matrix for Morella (SHRUB)
21 % W - the elevation matrix for the water table
22 % S - the matrix which determines available salinity at each cell
23 % delta - the thickness of each slab
24 %
25 % The routine will return the matrix for each of the plant species after
26 % propagating.
27
28
29 % fprintf('I am running PP! ')
30 %recently moved to main code:
31 % PlantRangeArray=[1 5;0.75 3;1.5 2.5;-0.5 1]; %all of the elevation ranges for p1-p4
32 % alpha=.01; %propagation rate for each populated cell
33 % DBE=.3; %death by elevation rate for each populated cell outside of plant's elevation range
34 % gdrange1=[-.02:.01:.08]; %range of percent values for growth/death for plant populations at (0, 50)% cover
35 % gdrange2=[-.02:.01:.08];%[-.04:.01:.04]; %range of percent values for growth/death for plant pops greater than 50% cover
36 % P1PctMax=.6; %largest percentage we will allow any plant population on a given cell to attain
37 % P2PctMax=.6;
38 % P3PctMax=.8;
39 % P4PctMax=.8;
40 % PctMax=[P1PctMax P2PctMax P3PctMax P4PctMax];
41 % MasterMax=0.8;% The most any cell can permit - 80% plant coverage
42
43
44 SwampWidth=MaxSwampWidth; % number of cells wide that the swamp should be - should find a better way of establishing, 10 'looks
   right' for now
45 WesternCells=zeros(1,SwampWidth);
46
47 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
48
49
```

```

50
51
52 %%
53
54 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
55 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%DEATH BY WATER TABLE STUFF%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
56 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
57 %not using—we have(?) data, but I've never seen us use water table concept
58
59
60 % HWcheck=H(:,1)-W(:,);
61 % HWcheck1=(HWcheck>0);
62 % HWcheck2a(:,)=(delta+HWcheck>=-0.5);
63 % HWcheck2b=(delta+HWcheck<=1);
64
65 % for i=1:size(H,1)
66 %     for j=1:size(H,2)
67 %         if H(i,j,2)~= -1
68 %             if HWcheck2a(i,j)==1
69 %                 if HWcheck2b(i,j) ==1
70 %                     if H(i,j,2)==2
71 %                         if P4(i,j,1)==-999
72 %                             P4(i,j,1)=0;
73 %                         end
74 %                     end
75 %                 end
76 %             end
77 %
78 %             if HWcheck1(i,j)==1
79 %                 if P1(i,j,1)==-999
80 %                     P1(i,j,1)=0;
81 %                 end
82 %
83 %                 if P2(i,j,1)==-999
84 %                     P2(i,j,1)=0;
85 %                 end
86 %
87 %                 if P3(i,j,1)==-999
88 %                     P3(i,j,1)=0;
89 %                 end
90 %             end
91 %             if HWcheck1(i,j)==0
92 %                 P1(i,j,1)=-999;
93 %                 P1(i,j,2)=0;
94 %                 P2(i,j,1)=-999;
95 %                 P2(i,j,2)=0;
96 %                 P3(i,j,1)=-999;
97 %                 P3(i,j,2)=0;
98 %             end
99 %         end
100 %     end
101 % end
102 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
103
104 %%
105

```

```

106
107 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
108 %%%GROWTH/DEATH/PROP%%
109 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
110 %splitting P_i into arrays for P_i(:,1) and P_i(:,2)
111 PA=zeros(size(Hstar,1),size(Hstar,2),4); %(P)lant (A)rray for tracking plant pct cover in growth/death/prop loop
112 PA(:,1)=P1(:,1);
113 PA(:,2)=P2(:,1);
114 PA(:,3)=P3(:,1);
115 PA(:,4)=P4(:,1);
116 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
117 PHA(:,1)=P1(:,2);
118 PHA(:,2)=P2(:,2);
119 PHA(:,3)=P3(:,2);
120 PHA(:,4)=P4(:,2);
121
122 % plants do not propagate during initialization, skip straight to beath by competition
123 if isnan(t)==0
124 % Beginning of the year, plants use full gdrange (death and growth)
125 if mod(t,26)==0
126 for i=1:size(PA,3)
127 PAB4=PA(:,i);
128 Px=PA(:,i);
129 Ph=PHA(:,i);
130 for R=1:size(Hstar,1)
131 for C=1:size(Hstar,2)
132 if dH(R,C)<-0.5
133 Px(R,C)=-999;
134 Ph(R,C)=0;
135 elseif ESL(R)~=0 && C>ESL(R)
136 Px(R,C)=-999;
137 Ph(R,C)=0;
138 else
139 if i==3
140 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
141 % if P3d(R,C,2)==0
142 % P3(R,C,1)=0;
143 % end
144 if dH(R,C)<0 && P3d(R,C,1)>0 %if elev. goes below zero we get rid of dead morella
145 P3d(R,C,1)=0;
146 P3d(R,C,2)=0;
147 end
148 end
149 if i~=4 %work on all Pi arrays except p4 first
150 if (dH(R,C)<PlantRangeArray(i,1) || (dH(R,C)>PlantRangeArray(i,2)) %if we are outside of the
% elevation range for this plant...
151 if Px(R,C)~= -999 %but we aren't underwater...
152 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
153 PxB4=Px(R,C); %store p3 %cover before death by elev.
154 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
155 if PxB4>0.01 && Px(R,C)<0.01 %if newly below 1% we create a p3d cell and kill
% off remaining p3
156 Px(R,C)=-999;

```



```

157         Ph(R,C)=0;
158         P3d(R,C,1) = .05;
159         P3d(R,C,2)=10;
160     end
161 elseif Px(R,C) >0 && dH(R,C)>=0 %if it's p1 or p2
162     outside of elevation we remove a percentage
163     Px(R,C)=Px(R,C)-DBE*Px(R,C);%-999; %changed on 9/21 see above
164     if Px(R,C)<0.01 %if that percentage drops below .1% we kill it
165         Px(R,C) = 0; %since it is outside of it's elev. range we
166         negate the cell for future popln.
167         Ph(R,C)=0;
168     end
169 else %should only have cells now underwater which we will
170     negate
171     Ph(R,C)=0;
172 end
173 if (dH(R,C)>=PlantRangeArray(i,1)) && (dH(R,C)<=PlantRangeArray(i,2)) %if we are inside of the plant
174     elevation range
175     % resetting negated cells which were previously submerged
176     if Px(R,C)==-999
177         Px(R,C)=0;
178         Ph(R,C)=Hstar(R,C,1);
179     end
180     % find the neighborhood, calc new popln for
181     % growth/propgatr :
182     % some % * (how many neighbors are
183     % populated)
184     Nhd=NewNhd(R,C,PAB4);
185     if Px(R,C) <.5
186         PxB4=Px(R,C);
187         kk=randi([1,length(gdrange1)]);
188         beta=gdrange1(kk);
189         Px(R,C)=min(Px(R,C)+sum(beta*Nhd)+beta*(beta>0)*(i==3),PctMax(i)); %grow by sum of
190         neighbors* beta, if it's morella grow a little more?
191     if i==3 && PxB4>=0.05 && Px(R,C)<=.05 %
192         add's chance of spread by birds anywhere inside of p3 range (same below)
193         Px(R,C)=0;
194         Ph(R,C)=0;
195         P3d(R,C,1) = .05;
196         P3d(R,C,2)=5;
197     end
198 else
199     kk=randi([1,length(gdrange2)]);
200     beta=gdrange2(kk);
201     Px(R,C)=min(Px(R,C)+sum(beta*Nhd)+beta*(beta>0)*(i==3),PctMax(i)); %grow by sum of
202     neighbors* beta, if it's morella grow a little more?
203 end
204 % record elevations for all cells inside the range,
205 % elevation=zero if no popln exists on cell currently
206 if Px(R,C)==0
207     Ph(R,C)=0;
208 elseif Px(R,C)>0
209     Ph(R,C)=Hstar(R,C,1);

```

```

206         elseif Px(R,C)<0
207             Px(R,C)=0;
208             Ph(R,C)=0;
209         end
210     end
211     elseif i==4
212         if (dH(R,C)<PlantRangeArray(i,1)) %if less than min height (-0.5)
213             if Px(R,C)~= -999 %negate (should already be done from first loop after R,C declared
214
215                 Ph(R,C)=0;
216             end
217         elseif (dH(R,C)>PlantRangeArray(i,2)) %elseif greater than max height, make 0 (no death by elev. just
218             kill)
219             Px(R,C)=0;
220             Ph(R,C)=0;
221         end
222         if (dH(R,C)>=PlantRangeArray(i,1)) && (dH(R,C)<=PlantRangeArray(i,2))
223             if ESL(R)~=0
224                 if MWSL(R)<=C && MESL(R)>=C
225                     Nhd=NewNhd(R,C,Px);
226                     if Px(R,C) <.5
227                         kk=randi([1,length(gdrange1)]);
228                         beta=gdrange1(kk);
229                         Px(R,C)=min(Px(R,C)+sum(beta*Nhd)+beta*(beta>0)*(i==3),PctMax(i)); %this also kills
230                             by a
231                     else
232                         kk=randi([1,length(gdrange2)]);
233                         beta=gdrange2(kk);
234                         Px(R,C)=min(Px(R,C)+sum(beta*Nhd)+beta*(beta>0)*(i==3),PctMax(i));
235                     end
236                 else
237                     end
238             end
239             Nhd=NewNhd(R,C,Px);
240             if Px(R,C) <.5
241                 kk=randi([1,length(gdrange1)]);
242                 beta=gdrange1(kk);
243                 Px(R,C)=min(Px(R,C)+sum(beta*Nhd)+beta*(beta>0)*(i==3),PctMax(i)); %this also kills by a
244             else
245                 kk=randi([1,length(gdrange2)]);
246                 beta=gdrange2(kk);
247                 Px(R,C)=min(Px(R,C)+sum(beta*Nhd)+beta*(beta>0)*(i==3),PctMax(i));
248             end
249             % record elevations for all cells inside the range,
250             % elevation=zero if no popln exists on cell currently
251             if Px(R,C)==0
252                 Ph(R,C)=0;
253             elseif Px(R,C)>0
254                 Ph(R,C)=Hstar(R,C,1);
255             elseif Px(R,C)<0 && Px(R,C)~= -999
256                 Px(R,C)=0;
257                 Ph(R,C)=0;
258             end
259         end

```

```

260         end
261     end
262 end
263 end
264 PA(:, :, i)=Px;
265 PHA(:, :, i)=Ph;
266 end
267
268
269 % Half year, plants only grow (no death, simulate spring time growth)
270 else
271     Spr.gdrange1=gdrange1(gdrange1>0);
272     Spr.gdrange2=gdrange2(gdrange2>0);
273     for i=1:size(PA,3)
274         PAB4=PA(:, :, i);
275         Px=PA(:, :, i);
276         Ph=PHA(:, :, i);
277         for R=1:size(Hstar,1)
278             for C=1:size(Hstar,2)
279                 if dH(R,C)<-.5
280                     Px(R,C)=-999;
281                     Ph(R,C)=0;
282                 elseif ESL(R)>0 && C>ESL(R)
283                     Px(R,C)=-999;
284                     Ph(R,C)=0;
285                 else
286                     if i==3
287                         if dH(R,C)<0 && P3d(R,C,1)>0 %if elev. goes below zero we get rid of dead morella
288                             P3d(R,C,1)=0;
289                             P3d(R,C,2)=0;
290                         end
291                     end
292                     if i~=4 %work on all Pi arrays except p4 first
293                         if (dH(R,C)<PlantRangeArray(i,1) || (dH(R,C)>PlantRangeArray(i,2))) %if we are outside of the
294                             elevation range for this plant...
295                             if Px(R,C)~= -999 %but we aren't underwater...
296                                 if i==3 && Px(R,C)>0 && dH(R,C)>0 && t~=0 %if it's p3 outside of elevation range we
297                                     have to create a dead morella value
298                                     PxB4=Px(R,C); %store p3 %cover before death by elev.
299                                     Px(R,C)=Px(R,C) - DBE*Px(R,C); %-999; %changed on 9/21 so that death by elevation
300                                     is not a sudden drop to 0
301                                     if PxB4>0.01 && Px(R,C)<0.01 %if newly below 1% we create a p3d cell and kill
302                                         off remaining p3
303                                         Px(R,C)=0;
304                                         Ph(R,C)=0;
305                                         P3d(R,C,1)=.05;
306                                         P3d(R,C,2)=5;
307                                     end
308                                 elseif Px(R,C) >0 && dH(R,C)>=0 %if it's p1 or p2
309                                     outside of elevation we remove a percentage
310                                     Px(R,C)=Px(R,C)-DBE*Px(R,C); %-999; %changed on 9/21 see above
311                                     if Px(R,C) <0.001 %if that percentage drops below .1% we kill it
312                                         Px(R,C) = 0; %since it is outside of it's elev. range we drop
313                                         to 0, will negate after elev drops below -0.5.
314                                     Ph(R,C)=0;
315                                 end
316                             end
317                         end
318                     end
319                 end
320             end
321         end
322     end
323 end

```

```

310         else                                     %should only have cells now underwater which we will
311             negate
312             Px(R,C)=0;
313             Ph(R,C)=0;
314         end
315     end
316     if i~=3 && (dH(R,C)>=PlantRangeArray(i,1)) && (dH(R,C)<=PlantRangeArray(i,2)) %if we are inside of
317         the plant elevation range, don't do morella this time
318         % resetting negated cells which were previously submerged
319         if Px(R,C)==-999
320             Px(R,C)=0;
321             Ph(R,C)=Hstar(R,C,1);
322         end
323         % find the neighborhood, calc new popln for
324         % growth/propgatr :
325         % some % * (how many neighbors are
326         % populated)
327         Nhd=NewNhd(R,C,PAB4);
328         if Px(R,C) <.5
329             PxB4=Px(R,C);
330             kk=randi([1,length(Spr_gdrange1)]);
331             beta=Spr_gdrange1(kk);
332             Px(R,C)=min(Px(R,C)+sum(beta*Nhd)+beta*(beta>0)*(i==3),PctMax(i)); %grow by sum of
333             neighbors* beta, if it's morella grow a little more?
334         if i==3 && PxB4>=0.05 && Px(R,C)<=.05 %
335             add's chance of spread by birds anywhere inside of p3 range (same below)
336             Px(R,C)=0;
337             Ph(R,C)=0;
338             P3d(R,C,1)=.05;
339             P3d(R,C,2)=10;
340         end
341     else
342         kk=randi([1,length(Spr_gdrange2)]);
343         beta=Spr_gdrange2(kk);
344         Px(R,C)=min(Px(R,C)+sum(beta*Nhd)+beta*(beta>0)*(i==3),PctMax(i)); %grow by sum of
345         neighbors* beta, if it's morella grow a little more?
346     end
347     % record elevations for all cells inside the range,
348     % elevation=zero if no popln exists on cell currently
349     if Px(R,C)==0
350         Ph(R,C)=0;
351     elseif Px(R,C)>0
352         Ph(R,C)=Hstar(R,C,1);
353     elseif Px(R,C)<0
354         Px(R,C)=0;
355         Ph(R,C)=0;
356     end
357     end
358     elseif i==4
359         if (dH(R,C)<PlantRangeArray(i,1)) %if less than min height (-0.5)
360             if Px(R,C)~-999 %negate (should already be done from first loop after R,C declared
361                 Px(R,C)=-999;
362                 Ph(R,C)=0;
363             end
364         elseif (dH(R,C)>PlantRangeArray(i,2)) %elseif greater than max height, make 0 (no death by elev. just

```

```

361         kill)
362         Px(R,C)=0;
363         Ph(R,C)=0;
364     end
365     if (dH(R,C)>=PlantRangeArray(i,1) && (dH(R,C)<=PlantRangeArray(i,2))
366         if ESL(R)~=0
367             if MWSL(R)<C && MESL(R)>=C
368                 Nhd=NewNhd(R,C,Px);
369                 if Px(R,C) <.5
370                     kk=randi([1,length(gdrange1)]);
371                     beta=gdrange1(kk);
372                     Px(R,C)=min(Px(R,C)+sum(beta*Nhd)+beta*(beta>0)*(i==3),PctMax(i)); %this also kills
373                     by a
374                 else
375                     kk=randi([1,length(gdrange2)]);
376                     beta=gdrange2(kk);
377                     Px(R,C)=min(Px(R,C)+sum(beta*Nhd)+beta*(beta>0)*(i==3),PctMax(i));
378                 end
379             else
380                 Px(R,C)=0;
381             end
382         end
383         % record elevations for all cells inside the range,
384         % elevation=zero if no popln exists on cell currently
385         if Px(R,C)==0
386             Ph(R,C)=0;
387         elseif Px(R,C)>0
388             Ph(R,C)=Hstar(R,C,1);
389         elseif Px(R,C)<0 && Px(R,C)~= -999
390             Px(R,C)=0;
391             Ph(R,C)=0;
392         end
393     end
394 end
395 end
396 end
397     PA(:, :, i)=Px;
398     PHA(:, :, i)=Ph;
399 end
400 end
401 end
402
403
404 %now updating changes in original plant matrices
405 P1(:, :, 1)=PA(:, :, 1);
406 P2(:, :, 1)=PA(:, :, 2);
407 P3(:, :, 1)=PA(:, :, 3);
408 P4(:, :, 1)=PA(:, :, 4);
409
410 P1(:, :, 2)=PHA(:, :, 1);
411 P2(:, :, 2)=PHA(:, :, 2);
412 P3(:, :, 2)=PHA(:, :, 3);
413 P4(:, :, 2)=PHA(:, :, 4);
414

```

```

415 %new temp plant matrices removing the -999 values to get an accurate total
416 %of all poplins on cells
417 Pt1=max(P1(:,1),0);
418 Pt2=max(P2(:,1),0);
419 Pt3=max(P3(:,1),0);
420 Pt4=max(P4(:,1),0);
421
422 Ptot=Pt1+Pt2+Pt3+Pt4;
423 %Death by Comp
424 fprintf('\n')
425 fprintf('The plants are killing each other!!!')
426 for i =1:size(Hstar,1)
427     for j =1:size(Hstar,2)
428         if Ptot(i,j)>MasterMax
429             Px1=Pt1(i,j,1);
430             Px2=Pt2(i,j,1);
431             Px3=Pt3(i,j,1);
432             Px4=Pt4(i,j,1);
433             if Px3==MasterMax
434                 P1(i,j,1)=0;
435                 P2(i,j,1)=0;
436                 P4(i,j,1)=0;
437             elseif (Px3 < MasterMax) && (Px3>0)
438                 k=(MasterMax-Px3)/(Px1+Px2+Px4);
439                 P1(i,j,1)=Px1*k;
440                 P2(i,j,1)=Px2*k;
441                 P4(i,j,1)=Px4*k;
442             elseif Px3==0
443                 %if Px1>=(k/4) && Px2>=(k/4) && Px3>=(k/4) && Px4>=(k/4)
444                 k=(Px1+Px2+Px4)-MasterMax;
445                 if Px1>=(k/3) && Px2>=(k/3) && Px4>=(k/3)
446                     P1(i,j,1)=Px1-(k/3);
447                     P2(i,j,1)=Px2-(k/3);
448                     P4(i,j,1)=Px4-(k/3);
449                     %Only two are bigger than k/3
450                     %%1 2%%
451                 elseif Px1>=(k/3) && Px2>=(k/3) && Px4<(k/3)
452                     P1(i,j,1)=Px1-((k/3)+(((k/3)-Px4)/2));
453                     P2(i,j,1)=Px2-((k/3)+(((k/3)-Px4)/2));
454                     P4(i,j,1)=0;
455                     %%1 4%%
456                 elseif Px1>=(k/3) && Px2<(k/3) && Px4>=(k/3)
457                     P1(i,j,1)=Px1-((k/3)+(((k/3)-Px4)/2));
458                     P2(i,j,1)=0;
459                     P4(i,j,1)=Px4-((k/3)+(((k/3)-Px2)/2));
460                     %%2 4%%
461                 elseif Px1<(k/3) && Px2>=(k/3) && Px4>=(k/3)
462                     P1(i,j,1)=0;
463                     P2(i,j,1)=Px2-((k/3)+(((k/3)-Px1)/2));
464                     P4(i,j,1)=Px4-((k/3)+(((k/3)-Px1)/2));
465                     %Only one is bigger than k/3%
466                     %%1%
467                 elseif Px1>=(k/3) && Px2<(k/3) && Px4<(k/3)
468                     P1(i,j,1)=Px1-((k/3)+((k/3)-Px2)+((k/3)-Px4));
469                     P2(i,j,1)=0;
470                     P4(i,j,1)=0;

```

```

471         %/2/%
472         elseif Px1<(k/3) && Px2>=(k/3) && Px4<(k/3)
473             P1(i,j,1)=0;
474             P2(i,j,1)=Px2-((k/3)+((k/3)-Px1)+((k/3)-Px4));
475             P4(i,j,1)=0;
476         %/4/%
477         elseif Px1<(k/3) && Px2<(k/3) && Px4>=(k/3)
478             P1(i,j,1)=0;
479             P2(i,j,1)=0;
480             P4(i,j,1)=Px4-((k/3)+((k/3)-Px1)+((k/3)-Px2));
481         end
482     end
483 end
484 end
485 end
486
487
488 %currently run this outside of PlantProcesses
489 for i=1:size(Hstar,1)
490     for j=1:size(Hstar,2)
491         PC1(i,j)=P1(i,j,1)*(P1(i,j,1)>0);
492         PC2(i,j)=P2(i,j,1)*(P2(i,j,1)>0);
493         PC3(i,j)=P3(i,j,1)*(P3(i,j,1)>0);
494         PC4(i,j)=P4(i,j,1)*(P4(i,j,1)>0);
495
496     end
497 end
498 Ptot=PC1+PC2+PC3+PC4;
499
500
501 function N1=NewNhd(R,C,PAB4)
502     if R<=(size(Hstar,1)-1)
503         R=floor(R);
504         Rp=R+1;
505     else
506         Rp=floor(R);
507     end
508     if R>=2
509         R=floor(R);
510         Rm=R-1;
511     else
512         Rm=floor(R);
513     end
514     if C<=(size(Hstar,2)-1)
515         C=floor(C);
516         Cp=C+1;
517     else
518         Cp=floor(C);
519     end
520     if C>=2
521         C=floor(C);
522         Cm=C-1;
523     else
524         Cm=floor(C);
525     end
526     N1=zeros(1,8);

```

```
527     %     PB4=PA(:, :, i);
528     N1=[PAB4(Rm,Gm) ,PAB4(Rm,C) ,PAB4(Rm,Cp) ,PAB4(R,Cp) ,PAB4(Rp,Cp) ,PAB4(Rp,C) ,PAB4(Rp,Gm) ,PAB4(R,Gm) ];
529     for k=1:8
530         if N1(k)==-999
531             N1(k)=0;
532         elseif N1(k)<0.01
533             N1(k)=0;
534         end
535     end
536 end
537
538 end
```


B.5 Shoreline code

```

1 function [Hstar, MeanBeachWidth, ESL, WSL, MESL, MWSL, OL]=Shoreline03312021(Hstar, delta, L, BchMax)
2 %this function is just to return the eastern shoreline for plant initialization
3 %P4 not permitted to grow east of ESL - (E)astern (S)hore (L)ine
4
5 [n1 n2]=size(Hstar(:, :, 1));
6 AL=zeros(n1,1); %AdjacentLength~width of beach
7 OL=zeros(n1,1); %OppositeLocation~innermost reach of beach/index of first cell>BchMax
8 ESL=zeros(n1,1); %EAST shoreline (column value per row;
9 WSL=zeros(n1,1); %formerly "WesternShore"
10 MESL=zeros(n1,1); %formerly "SlimeSwamp"
11 MWSL=zeros(n1,1); %formerly "WBdrySwamp"
12
13 beta=zeros([n1 1]); %this gets called but isn't currently used, stores profile slope
14 Beta=zeros([n1 1]); %bruun rule beta
15
16 R=zeros(n1,1);
17 DC=zeros(n1,2);
18
19
20 dH=delta*Hstar(:, :, 1); %making new array of Hstar values in terms of meters instead of slabs (ease of use)
21
22 for i=1:n1
23     Rnow=dH(i, :);
24     if Rnow(n2)<0
25         for j1=n2-1:-1:2 %declaring eastern shoreline by looking for first cell with positive elevation west of a
26             cell with negative elevation
27                 if Rnow(j1)>=0 && Rnow(j1+1)<=0 && ESL(i)==0
28                     if Rnow(j1-1)~=0
29                         ESL(i)=j1;
30                     end
31                 end
32                 if j1==2 && ESL(i)==0
33                     ESL(i)=ESL(i)+1*(Rnow(1)>=0); %special condition for first column - avoid index errors
34                 end
35                 if ESL(i)~=0 && ESL(i)~=1 %if we found a shoreline (that wasn't in col 1) need to declare AL and OL
36                     DC(i, 2)=max(Rnow); %will use max of current row if no cell satisfies being greater than
37                     BchMax
38                     DC(i, 1)=find(Rnow==DC(i, 2), 1, 'last');
39                     flag=0;
40                     while flag==0
41                         for j2=j1-1:-1:1 % looking for dune crest starting with shoreline and moving
42                             west
43                             if Rnow(j2)>=BchMax % if we find a cell >= BchMax
44                                 OL(i)=j2; % then that cell is the dune crest
45                                 AL(i)=j1-j2; % width of the beach is shoreline-dunecrest (indexes)
46                                 flag=1;
47                                 break
48                             elseif Rnow(j2)<0 % if we go below water
49                                 m=1;
50                                 while j2-m>=1
51                                     if Rnow(j2-m)>0
52                                         j2=j2-m; %if we get back above water, change j2 and keep looking for
53                                         swamp/dune crest
54                                         m=1;

```

```

51         elseif Rnow(j2-m) <=-0.5
52             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
53             AL(i)=j1-OL(i);           %width of beach is shoreline index - the index of max height
                    of subaerial island
54             flag=1;
55             break
56         elseif m=j2-1
57             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
58             AL(i)=j1-OL(i);
59             m=j2;
60             flag=1;
61         else
62             m=m+1;
63         end
64     end
65     if j2==1 %j2==2           %if we
66         OL(i)=DC(i,1);           % make the dune crest the max height of the positive elevation portion
                    of the island
67         AL(i)=j1-OL(i);           %width of beach is shoreline - the max height of subaerial island
68         flag=1;
69     end
70     elseif j2==1
71         OL(i)=DC(i,1);           % make the dune crest the max height of the positive elevation portion of
                    the island
72         AL(i)=j1-OL(i);           %width of beach is shoreline - the max height of subaerial island
73         flag=1;
74     end
75     if OL(i)>0 && AL(i)>0
76         break
77     end
78 end
79 end
80 end
81 end
82 if OL(i)>0 && AL(i)>0
83     break
84 end
85 end
86 end
87 end
88
89 for ii=1:n1
90     if ESL(ii)~=0
91         Rnow=dH(ii,:);
92         MWSL(ii)=find(Rnow>=-0.5,1,'first');
93         DC(ii)=find(Rnow==max(Rnow),1,'first');
94         SwmpChk=Rnow(MWSL(ii)+1:DC(ii));
95         j=find(SwmpChk>1,1,'first');
96         if isempty(j)==1
97             MESL(ii)=DC(ii);
98         else
99             MESL(ii)=MWSL(ii)+j;
100     end
101 end

```

```
102 end
103
104 MBWfactor=(AL~=0); %calculating changes in beach width – does nothing unless used outside fcn in an image
105 MeanBeachWidth=(sum(AL(MBWfactor)))/sum(MBWfactor); %same
106
107 end
```

B.6 Marine Processes code

```

1 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)
2 alpha=1;
3 AoR=pi/6;
4
5
6 T=t/26;
7 AccelCheck=MigYr*(Ma)^T
8 if floor(AccelCheck)>MigYr
9     Mig=floor(AccelCheck);
10 elseif AccelCheck==0
11     Mig=AccelCheck;
12 else
13     Mig=MigYr;
14 end
15
16 nn=5; %number of rows above and below current row to check plant density when calculating migration
17
18 if isnan(t)==1
19     Tflag=1;
20     t=0;
21 else
22     Tflag=0;
23 end
24
25 test=0;
26 SLR=0.00635; %was for Bruun Rule testing - not currently used
27 Htest=0; %handy in editing to just declare this
28 Ptest=0; %handy in editing to just declare this
29
30 [n1 n2]=size(Hstar(:, :, 1));
31 AL=zeros(n1,1); %AdjacentLength~width of beach
32 OL=zeros(n1,1); %OppositeLocation~innermost reach of beach/index of first cell>BchMax
33 ESL=zeros(n1,1); %EAST shoreline (column value per row;
34 WSL=zeros(n1,1); %formerly "WesternShore"
35 MESL=zeros(n1,1); %formerly "SlineSwamp"
36 MWSL=zeros(n1,1); %formerly "WBdrySwamp"
37
38 beta=zeros([n1 1]); %this gets called but isn't currently used, stores profile slope
39 Beta=zeros([n1 1]); %bruun rule beta
40
41 R=zeros(n1,1);
42 DC=zeros(n1,2);
43
44
45 dH=delta*Hstar(:, :, 1); %making new array of Hstar values in terms of meters instead of slabs (ease of use)
46
47 for i=1:n1
48     Rnow=dH(i, :);
49     Pnow=P3(i, :, 1);
50     PHnow=P3(i, :, 2);
51     if Rnow(n2)<0
52         for j1=n2-1:-1:2 %declaring eastern shoreline by looking for first cell with positive elevation west of a
            cell with negative elevation

```

```

53     if Rnow(j1)>=0 && Rnow(j1+1)<=0 && ESL(i)==0
54         if Rnow(j1-1)~=0
55             ESL(i)=j1;
56         end
57     end
58     if j1==2 && ESL(i)==0
59         ESL(i)=ESL(i)+1*(Rnow(1)>=0);           %special condition for first column - avoid index errors
60     end
61     if ESL(i)~=0 && ESL(i)~=1                   %if we found a shoreline (that wasn't in col 1) need to declare AL and OL
62         DC(i,2)=max(Rnow);                     %will use max of current row if no cell satisfies being greater than
63         BchMax
64         DC(i,1)=find(Rnow==DC(i,2),1,'last');
65         flag=0;
66         while flag==0
67             for j2=j1-1:-1:1                     % looking for dune crest starting with shoreline and moving
68                 west
69                 if Rnow(j2)>=BchMax              % if we find a cell >= BchMax
70                     OL(i)=j2;                  % then that cell is the dune crest
71                     AL(i)=j1-j2;              % width of the beach is shoreline-dunecrest (indexes)
72                     flag=1;
73                     break
74                 elseif Rnow(j2)<0              % if we go below water
75                     m=1;
76                     while j2-m>=1
77                         if Rnow(j2-m)>0
78                             j2=j2-m;          %if we get back above water, change j2 and keep looking for
79                             swamp/dune crest
80                             m=1;
81                             elseif Rnow(j2-m)<=-0.5
82                                 OL(i)=(j2)+find(Rnow(j2+1:j1)==max(Rnow(j2+1:j1)),1,'last'); % make the dune
83                                 crest the max height of the positive elevation portion of the island
84                                 AL(i)=j1-OL(i); %width of beach is shoreline index - the index of max height
85                                 of subaerial island
86                                 flag=1;
87                                 break
88                             elseif m==j2-1
89                                 OL(i)=DC(i,1); %if we search the rest of the row and don't find swamp/dune
90                                 crest use max(row) as dune crest
91                                 AL(i)=j1-OL(i);
92                                 m=j2;
93                                 flag=1;
94                             else
95                                 m=m+1;
96                             end
97                         end
98                     if j2==1 %j2==2           %if we
99                         OL(i)=DC(i,1);        % make the dune crest the max height of the positive elevation portion
100                        of the island
101                        AL(i)=j1-OL(i);        %width of beach is shoreline - the max height of subaerial island
102                        flag=1;
103                    end
104                elseif j2==1
105                    OL(i)=DC(i,1);            % make the dune crest the max height of the positive elevation portion of
106                    the island
107                    AL(i)=j1-OL(i);          %width of beach is shoreline - the max height of subaerial island

```

```

101         flag=1;
102     end
103     if OL(i)>0 && AL(i)>0
104         break
105     end
106 end
107 end
108 end
109 if OL(i)>0 && AL(i)>0
110     break
111 end
112 end
113 if ESL(i)>1           %if the island has been found in this row
114     for m=0:AL(i)     %removing P3 from the beach
115         j=OL(i)+m;
116         if Pnow(j)>0
117             Pnow(j)=0;
118             PHnow(j)=0;
119         end
120     end
121     P3(i, :, 1)=Pnow;
122     P3(i, :, 2)=PHnow;
123 elseif ESL(i)==1     %if shoreline is first cell make OL, AL first cell
124     OL(i)=1;
125     AL(i)=1;
126     if Pnow(1)>0
127         Pnow(1)=0;
128         PHnow(1)=0;
129     end
130 end
131 end
132 end
133
134 MBWfactor=(AL~=0); %calculating changes in beach width – does nothing unless used outside fcn in an image
135 MeanBeachWidth=(sum(AL(MBWfactor)))/sum(MBWfactor); %same
136
137
138
139 %now calculating the foreshore slope and resetting cells
140 cnt=0;
141 DofCi=zeros(n1,1);
142 xdoc=sym('xdoc');
143
144 for i=1:n1
145     if ESL(i)>0
146         cnt=cnt+1;
147         Rnow=dH(i, :);
148         if AL(i)~=0 %if we have a shoreline and the adjacent length is not zero
149             beta(i)=tan(Rnow(OL(i))/AL(i)); %calculate the slope of the shoreline
150             R(i)=1/beta(i); %part of Bruun rule – unused
151         elseif AL(i)==0
152             beta(i)=.0003; %if there is a shore that is one cell wide (special condition above) use common shore slope
153             R(i)=1/beta(i);
154         end
155     end
156 end

```

```

157 cnt=0;
158 shortR=0; %for removing NaN and infinity slopes - possible with holes and ponds, but unlikely (pretty sure I debugged this issue)
159 for i=1:n1
160     if ESL(i)>0
161         cnt=cnt+1;
162         shortR(cnt)=R(i);%*(isnan(R(i))==0); %uncomment this if NaN issue (see above) comes up
163     end
164 end
165 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
166 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%for-Vector-of-Migration-Years%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
167 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
168 %below is for using a vector of predetermined years to trigger migration
169 % YrsPast=sum(SLRyrs(SLRyrs<0)); %any negative years in vector are summed
170 % YrCnt=(t/26)+YrsPast; %current number of years since last migration
171 % if YrCnt==max(SLRyrs(SLRyrs>0)) %if current #yrs is the next number of years in vector of vaues to trigger
migration
172 % MigChk=1; %trigger migration
173 % kk=find(SLRyrs==max(SLRyrs(SLRyrs>0))) %find that yr value in vector
174 % SLRyrs(kk)=-SLRyrs(kk); %replace with negative year so it will be summed as years past
175 % else
176 % MigChk=0;
177 % end
178 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
179 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%END%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
180 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
181
182 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
183 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%For_Yearly_Migration%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
184 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
185
186 Plx=zeros(n1,1);
187 if 0==mod(t,26) %&& t~=0
188     MigChk=1;
189     if MigChk==1
190         for i=1:n1
191             if i<nn
192                 PIChkArea=PCmp(1:1+nn,: ,1); %plant cover for rows near current row
193                 Plx(i)=mean(PIChkArea(PIChkArea>=0));
194                 if sum(dH(i,:) >=0)==0
195                     MigCnt(i)=MigCnt(i)+Mig;
196                 else
197                     if Plx(i) >= 0.3
198                         MigCnt(i)=MigCnt(i)+round(0.00*Mig); %reduce migration by 70% if nearby weighted plant cover exceeds 50%
199                     elseif Plx(i) >= 0.2 && Plx(i) < 0.3
200                         MigCnt(i)=MigCnt(i)+round(0.3*Mig); %reduce migration by 50% if nearby weighted plant cover exceeds 35%
but not 50%
201                     elseif Plx(i) >= 0.1 && Plx(i) < 0.1
202                         MigCnt(i)=MigCnt(i)+round(0.5*Mig); %reduce migration by 30% if nearby weighted plant cover exceeds 10%
but not 35%
203                     else
204                         MigCnt(i)=MigCnt(i)+Mig; %full effect of migration if nearby plant cover less than 10%
205                     end
206                 end
207                 if MigCnt(i)>ScaleFactor
208                     % MigR=1; %set MigR=1 if using scaled version of
migration

```

```

209     MigR=floor(MigCnt(i)/ScaleFactor);
210     fprintf('\n')
211     fprintf('THE SHORELINE AT ROW %d IS MIGRATING EAST BY %d COLUMNS!',i,MigR)
212     RnowDummy=zeros(1,n2);
213     P1RnowDummy=RnowDummy;
214     P2RnowDummy=RnowDummy;
215     P3RnowDummy=RnowDummy;
216     P4RnowDummy=RnowDummy;
217     RnowDummy(1:n2-MigR)=dH(i,MigR+1:n2);
218     P1RnowDummy(1:n2-MigR)=P1(i,MigR+1:n2);
219     P2RnowDummy(1:n2-MigR)=P1(i,MigR+1:n2);
220     P3RnowDummy(1:n2-MigR)=P1(i,MigR+1:n2);
221     P4RnowDummy(1:n2-MigR)=P1(i,MigR+1:n2);
222     dH(i,1:n2-MigR)=RnowDummy(1:n2-MigR);
223     P1(i,1:n2-MigR)=P1RnowDummy(1:n2-MigR);
224     P2(i,1:n2-MigR)=P1RnowDummy(1:n2-MigR);
225     P3(i,1:n2-MigR)=P1RnowDummy(1:n2-MigR);
226     P4(i,1:n2-MigR)=P1RnowDummy(1:n2-MigR);
227     MigCnt(i)=MigCnt(i)-MigR*ScaleFactor;
228     end
229     elseif nn<i && i<n1-nn
230     %
231     %     fprintf('oo')
232     %     end
233     P1ChkArea=PCmp(i-nn:i+nn,: ,1); %plant cover for rows near current row
234     Plx(i)=mean(P1ChkArea(P1ChkArea>=0));
235     if sum(dH(i,:) >=0)==0
236         MigCnt(i)=MigCnt(i)+Mig;
237     else
238         if Plx(i) >= 0.3
239             MigCnt(i)=MigCnt(i)+round(0.00*Mig); %reduce migration by 70% if nearby weighted plant cover exceeds 50%
240         elseif Plx(i) >= 0.2 && Plx(i) < 0.3
241             MigCnt(i)=MigCnt(i)+round(0.3*Mig); %reduce migration by 50% if nearby weighted plant cover exceeds 35%
242                 but not 50%
243         elseif Plx(i) >= 0.1 && Plx(i) < 0.2
244             MigCnt(i)=MigCnt(i)+round(0.5*Mig); %reduce migration by 30% if nearby weighted plant cover exceeds 10%
245                 but not 35%
246         else
247             MigCnt(i)=MigCnt(i)+Mig; %full effect of migration if nearby plant cover less than 10%
248         end
249     end
250     if MigCnt(i)>ScaleFactor
251     %
252     %     MigR=1; %set MigR=1 if using scaled version of
253     %     migration
254     MigR=floor(MigCnt(i)/ScaleFactor);
255     fprintf('\n')
256     fprintf('THE SHORELINE AT ROW %d IS MIGRATING EAST BY %d COLUMNS!',i,MigR)
257     RnowDummy=zeros(1,n2);
258     P1RnowDummy=RnowDummy;
259     P2RnowDummy=RnowDummy;
260     P3RnowDummy=RnowDummy;
261     P4RnowDummy=RnowDummy;
262     RnowDummy(1:n2-MigR)=dH(i,MigR+1:n2);
263     P1RnowDummy(1:n2-MigR)=P1(i,MigR+1:n2);
264     P2RnowDummy(1:n2-MigR)=P1(i,MigR+1:n2);
265     P3RnowDummy(1:n2-MigR)=P1(i,MigR+1:n2);

```



```

262 P4RnowDummy(1:n2-MigR)=P1(i,MigR+1:n2);
263 dH(i,1:n2-MigR)=RnowDummy(1:n2-MigR);
264 P1(i,1:n2-MigR)=P1RnowDummy(1:n2-MigR);
265 P2(i,1:n2-MigR)=P1RnowDummy(1:n2-MigR);
266 P3(i,1:n2-MigR)=P1RnowDummy(1:n2-MigR);
267 P4(i,1:n2-MigR)=P1RnowDummy(1:n2-MigR);
268 MigCnt(i)=MigCnt(i)-MigR*ScaleFactor;
269 end
270 elseif i>n1-nn
271 PIChkArea=PCmp(n1-nn:n1,,: ,1); %plant cover for nows near current row
272 Plx(i)=mean(PIChkArea(PIChkArea>=0));
273 if sum(dH(i, :)>=0)==0
274 MigCnt(i)=MigCnt(i)+Mig;
275 else
276 if Plx(i) >= 0.5
277 MigCnt(i)=MigCnt(i)+round(0.00*Mig); %reduce migration by 70% if nearby weighted plant cover exceeds 50%
278 elseif Plx(i) >= 0.35 && Plx(i) < 0.5
279 MigCnt(i)=MigCnt(i)+round(0.3*Mig); %reduce migration by 50% if nearby weighted plant cover exceeds 35%
but not 50%
280 elseif Plx(i) >= 0.1 && Plx(i) < 0.35
281 MigCnt(i)=MigCnt(i)+round(0.5*Mig); %reduce migration by 30% if nearby weighted plant cover exceeds 10%
but not 35%
282 else
283 MigCnt(i)=MigCnt(i)+Mig; %full effect of migration if nearby plant cover less than 10%
284 end
285 end
286 if MigCnt(i)>ScaleFactor
287 % MigR=1; %set MigR=1 if using scaled version of
migration
288 MigR=floor(MigCnt(i)/ScaleFactor);
289 fprintf('\n')
290 fprintf('THE SHORELINE AT ROW %d IS MIGRATING EAST BY %d COLUMNS!',i,MigR)
291 RnowDummy=zeros(1,n2);
292 P1RnowDummy=RnowDummy;
293 P2RnowDummy=RnowDummy;
294 P3RnowDummy=RnowDummy;
295 P4RnowDummy=RnowDummy;
296 RnowDummy(1:n2-MigR)=dH(i,MigR+1:n2);
297 P1RnowDummy(1:n2-MigR)=P1(i,MigR+1:n2);
298 P2RnowDummy(1:n2-MigR)=P1(i,MigR+1:n2);
299 P3RnowDummy(1:n2-MigR)=P1(i,MigR+1:n2);
300 P4RnowDummy(1:n2-MigR)=P1(i,MigR+1:n2);
301 dH(i,1:n2-MigR)=RnowDummy(1:n2-MigR);
302 P1(i,1:n2-MigR)=P1RnowDummy(1:n2-MigR);
303 P2(i,1:n2-MigR)=P1RnowDummy(1:n2-MigR);
304 P3(i,1:n2-MigR)=P1RnowDummy(1:n2-MigR);
305 P4(i,1:n2-MigR)=P1RnowDummy(1:n2-MigR);
306 MigCnt(i)=MigCnt(i)-MigR*ScaleFactor;
307 end
308 end
309 end
310 end
311 end
312 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
313 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%END%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
314 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```

```

315
316 % *****RESETTING EQUILIBRIUM PROFILE OF SHORELINE HAS BEEN REMOVED FOR THIS VERSION*****
317 % *****see Fix2MarineProcesses02202021 for most recent version of resetting profile***
318
319 dH=round(dH,1); %need to round to 1 dec. place if redeclared shoreline
320 Hstar(:,1)=(1/delta)*dH; %convert back to slabs when redeclaring Hstar
321
322 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
323 if Tflag==1
324     t=NaN; %reset t if this is initialization MP so it doesn't throw of loop in MainCode
325 end
326
327 end

```