ABSTRACT

Title of Thesis:	USING AN INDIVIDUAL BASED MODEL TO EVALUATE THE EFFECTS OF CLIMATE CHANGE ON THE REPRODUCTIVE PHENOLOGY OF EELGRASS (*ZOSTERA MARINA L.*) ALONG A LATITUDINAL GRADIENT
	Jessica Lynn Foley, Master of Science 2017
Thesis Directed By:	Associate Professor, Dr. Lora A. Harris Marine-Estuarine-Environmental Sciences

I explored the effects of climate change on the reproductive biology of the clonal marine angiosperm *Zostera marina L.* (eelgrass) using an individual-based model. The model captures whole plant ontogeny, morphology, and ecophysiology from seed to reproductive adult to simulate the plasticity of eelgrass in response to environmental variables. Using a latitudinal gradient as a proxy for climate change, virtual seeding experiments were performed in three locations along the East coast of the United States. I simulated the impacts of increased temperatures on *Z. marina's* biomass, reproductive phenology, and life history. Warmer temperatures resulted in a modeled decrease of *Z. marina*'s total biomass, as well as altered reproductive timing and strategy. These results have implications for long term predictions of *Z. marina* persistence in its traditional biogeographic range, and indicate adaptation via shifts in phenology and reproductive strategy may interact to dampen some negative consequences of increased temperatures.

USING AN INDIVIDUAL BASED MODEL TO EVALUATE THE EFFECTS OF CLIMATE CHANGE ON THE REPRODUCTIVE PHENOLOGY OF EELGRASS (*ZOSTERA MARINA L.*) ALONG A LATITUDINAL GRADIENT

by

Jessica Lynn Foley

Thesis submitted to the Faculty of the Graduate School of the
University of Maryland, College Park, in partial fulfillment
of the requirements for the degree of
Master of Science
2017

Advisory Committee:
Associate Professor Dr. Lora Harris, Chair
Professor Dr. Walter Boynton
Associate Professor Dr. Mark Brush

Preface

This thesis presents a new eelgrass model, built upon the Virtual Eelgrass Meadow (VEMv.1) developed by my graduate advisor Dr. Lora Harris (2006). The new model (VEMv.2) incorporates a complete suite of eelgrass life history stages (seed, seedling, adult) as well as both asexual and sexual reproduction. The thesis is presented in one comprehensive chapter with a detailed methods section and full modeling code documented in Appendix A. Included in the chapter is a description of the new model using a modified version of the "ODD [overview, design concepts, and details] protocol" (Grimm et al. 2006). This standardized format for describing individual-based models has helped modelers overcome the challenge of presenting their work in a more complete and consistent manner, especially those using an individual-based approach. In addition to describing the model, the thesis presents results from virtual experiments evaluating the effects of warming temperatures on the reproductive phenology of eelgrass using a latitudinal gradient as a proxy for climate change.

Data used to parameterize, calibrate, and validate the model were compiled from both published and unpublished datasets, mainly those collected in Rhode Island by Dr. Joanne Bintz and in Virginia and North Carolina by Dr. Jessie Jarvis. Seedling biomass samples used to parameterize the model and as initial conditions were collected by Dr. J.J. Orth and his staff, and processed by Dr. Mark Brush at the Virginia Institute of Marine Science (VIMS). Model forcing function data were downloaded from the National Estuarine Research Reserves (NERRs) in North Carolina, Virginia, and New Hampshire. In thesis revisions and as we continue to

refine the model for future use, a data entry error was discovered for the New Hampshire temperature forcing function for years 2007-2009 where air temperature was incorrectly substituted for water temperature in the thesis study. This error is being addressed and corrected for publication and does not affect the model presented or the Virginia and North Carolina simulation output directly. Funding to support my graduate studies and research came from a Maryland Sea Grant Research Fellowship.

Acknowledgements

First, I would like to express my sincere gratitude to my advisor, Dr. Lora A. Harris, for her guidance, patience, and immense knowledge during my thesis study. The continued support she provided me is what got me over the finish line. I could not have managed without her unrelenting efforts and belief in me. I would also like to thank my thesis committee, Dr. Walter Boynton and Dr. Mark Brush, for their insightful comments, warm encouragement, and flexibility.

To the Harris Lab Group, thank you for your camaraderie. I will never forget the great times we spent in the field in pursuit of our research. To the CBL community, thanks for offering your friendship and passion.

I am very grateful to the late Dr. Scott W. Nixon and the late Margaret A. Davidson, two remarkable leaders, who inspire my career focus and my continued passion in coastal research, management and policy. Thank you for your originality and refreshing candor.

I would like to recognize and thank Maryland Sea Grant for funding my thesis study and for providing me with the opportunity to take part in the John A. Knauss Marine Policy Fellowship. I am also grateful to Dr. Jessie Jarvis, my network at NOAA, and Jasper Taylor and his incredible team at Simulistics.

Finally, I must express my very profound gratitude to my family. To my mother, Cindy, thank you for always believing in me and helping me to maintain a positive outlook. To my partner, Dean, thank you for your love and continuous encouragement. This accomplishment would not have been possible without both of you and your unfailing support every step of the way.

Table of Contents

Preface ii
Acknowledgements iv
Table of Contents v
List of Tables vii
List of Figures viii
1. INTRODUCTION 1
2. METHODS 8
2.1 Study Organism 8
2.2 History of the Virtual Eelgrass Meadow (VEMv.1) 10
2.3 General Model Description (VEMv.2) 11
2.3.1 Process Overview and Scheduling 11
2.4 Understanding Eelgrass Ontogeny and Morphology for Model Application 12
2.4.1 Eelgrass Life Stages 12
2.4.2 Understanding Eelgrass Ecophysiology for Model Application 14
2.5 Model Formulation 17
2.6 Modeling Eelgrass Growth 17
2.6.1 The Adult Specific Growth Rate Formulation 18
2.6.2 The Seedling Specific Growth Rate Formulation 19
2.7 Modeling Reproduction 22
2.7.1 Asexual Reproduction 22
2.7.2 Sexual Reproduction 22
2.7 Study Locations 25
2.8 Model Forcing Functions, State Variables, and Parameterization 25
2.8.1 Seedling Specific Growth Rate Parameterization 26
2.9 Model Calibration and Validation 27
2.9.1 Calibration 28
2.9.2 Validation 28
2.10 Sensitivity Analyses 29
2.11 Model Simulations 30
2.11.2 Projected Temperature Simulations 31
2.11.3 Experimental Simulations 32
2.11.1 Model Assumptions 33
3. RESULTS AND DISCUSSION 34
3.1 Overview of Results 34
3.2 Model Calibration and Validation Results 34
3.2.1 Calibration Results 35
3.2.2 Validation of Specific Growth Rates 35
3.2.3 Validation of Seedling Biomass State Variables 35
3.3 Results from Model Sensitivity Analysis 38
3.4 Model Simulation Results and Discussion 39
Impacts to Reproductive Phenology 40
Impacts to Biomass 42
Impacts on Life History Strategies 45
3.5 Model Limitations and Future Work 46
3.5.1 Mortality due to temperature 46

3.5.2 Additional Future Work 47
4. CONCLUSION 49
TABLES 52
FIGURES 64
APPENDIX A – MODEL EQUATIONS 91
APPENDIX B – COMPLETE MODEL PROGRAMMING 134
BIBLIOGRAPHY 247

List of Tables

1-1. List of key eelgrass simulation models 1980 to 2016.

1-2. The seven elements of the original and updated ODD protocol.

1-3. A literature review of the term “eelgrass seedling”.

1-4. Parameter estimates for VEMv.2.

1-5. List of essential equations for eelgrass growth in VEMv.2.

1-6. Forcing functions and initial conditions used in VEMv.2 temperature warming simulations.

1-7. Relationships between *Z. marina* reproductive phenology and temperature.

1-8. Brush and Orth (2015) unpublished data on seedling biomass.

1-9. Results from VEMv.2 sensitivity analyses.

1-10. Biomass and resource allocation results from seeding experiment.

1-11. Results of reproductive phenology from seeding experiment.

1-12. Model temperature anomalies for years 2025, 2050, and 2090 developed using A2 scenarios and averaged over the spatial area of the Chesapeake Bay watershed.

List of Figures

1-1. The theoretical effect of changes in population size on changes in first flowering date.

1-2. Predicted shift in seagrass species based on their resilience as defined by genetics, biochemistry, morphology, life history, and community competition.

1-3. Eelgrass morphology (illustration courtesy M. Fonseca).

1-4. Illustration of eelgrass ontogeny from germinated seed (A, B) to seedling (C, D, E) to adult (F).

1-5. Conceptual diagram of the Virtual Eelgrass Meadow version 2 (VEMv.2).

1-6. In-depth botanical illustration and description of the development of a germinated eelgrass seed into a seedling.

1-7. Visualization of the feedback loop between the leaf length "clock" and the dependency of available light on local shoot density in adult plants.

1-8. VEMv.2 sexual reproduction rules modified by Jarvis et al. (2014).

1-9. Reproductive phenology of *Z. marina* at different locations (with latitude) along the East coast of the United States.

1-10. Model simulation locations along a latitudinal gradient of the East coast of the U.S.: Great Bay, New Hampshire (NH), South Bay, Virginia (VA), and Masonboro, North Carolina (NC).

1-11. Temperature forcing functions used in the model climate scenarios.

1-12. Surface photosynthetically active radiation (PAR) forcing functions used in the model climate scenarios.

1-13. Seedling specific growth rate surface plot showing response to temperature and light.

1-14. Monthly Chesapeake Bay sea surface temperature (SST) anomalies from 1990-2090 (Cole 2008).

1-15. Current and projected temperature forcing functions for model climate simulations.

1-16. VEMv.2 modeled results versus observed seedling biomass from a Rhode Island mesocosm study (Bintz 2002).

1-17. VEMv.2 modeled results versus observed seedling biomass in Virginia's York River.

1-18. VEMv.2 seedling biomass sensitivity analysis.

1-19. Simile software visual environment for VEMv.1 and VEMv.2.

1-20. VEMv.2 Average Seedling vs. Adult Specific Growth Rates in Virginia.

1-21. Reproductive phenology: an individual's probability of flowering versus the first flowering day of the year.

1-22. Reproductive phenology: the effect of latitude on germination day over the model simulations.

1-23. Reproductive phenology results: timing of seed germination and latitude.

1-24. Modeled light reaching eelgrass leaf after attenuating through the water column in South Bay, Virginia.

1-25. Diagram showing the dominant traits among colonizing (C), opportunistic (O), and persistent seagrass genera with respect to shoot turnover, genet persistence, time to reach sexual maturity and seed dormancy (Kilminster 2015).

1-26. Measured seedling biomass approximately two months post germination. Seeds were collected from South Bay, VA and Mobjack Bay, VA. (Orth and Brush 2015).

1-27. Total areal biomass climate simulation results years 2007 and 2090.

1. INTRODUCTION

Understanding how organisms will respond to global climate change is one of the greatest challenges and unknowns facing the scientific community to date (Steffen et al. 2006, see also Root et al. 2003, Williams et al. 2008, Duarte 2014). Migration is one strategy to adapt to adverse conditions, but less mobile organisms with limited dispersal mechanisms (e.g. some plants) will be subject to *in situ* changes, relying on either short term acclimation or evolutionary adaptation or genetic change in the long term (Williams et al. 2008, Anderson et al. 2012). One approach to predicting the response of these organisms is to understand the degree to which phenotypic plasticity, or the ability of an organism to alter or express certain traits, provides for adaptation to changed climatic conditions (Hoffmann and Sgro 2011).

One trait of particular sensitivity to environmental cues in temperate plants is the timing of reproductive events or reproductive phenology (Sherry et al. 2007, Tooke and Battey 2010). Relationships between reproductive phenology and environmental factors, particularly temperature, have been well established in terrestrial systems, but less so for submerged vascular plants in estuarine and coastal ecosystems (Orth et al. 2006, but see Smith and Walker 2002, Diaz-Almela et al. 2006). In terrestrial studies of flowering phenology, some studies support the possibility for plant populations to keep pace with changes in temperature by displaying different reproductive strategies or shifting the timing of key reproductive events such as the flowering day. The conceptual diagram displayed in Figure 1 outlines the theoretical effect of changes in population size on changes in first

flowering date. While an earlier first flowering day does not typically alter the time of peak flowering, it may account for the presence of more flowers during peak bloom, thereby increasing the populations' fecundity. It is also important to note that the term "first flowering date" (FFD) is used in many phenological datasets to track records of flower emergence times, despite some ongoing debate regarding what formally constitutes flower emergence (Miller-Rushing and Primack 2008, Tooke and Battey 2010).

Within the context of changing environmental conditions, an individual plant's ability to continually acclimate to environmental variation (i.e. phenotype or plasticity) may be limited by its own genotype or ability for adaptive evolution (Anderson et al. 2012). Understanding plant productivity and phenology requires an interdisciplinary approach that incorporates information across life history, meteorology, climate seasonality, and physiology (Tooke and Battey 2010). Koch (2016) developed a diagram that predicts how seagrass species will shift to tolerate stress based on their "resilience zone", which includes aspects of genetics, biochemistry, morphology, and life history (Figure 2). Using the temperate marine angiosperm Eelgrass (*Zostera marina* L.) as the study organism, this thesis presents a modeling approach to better understand the ecophysiology that underpins these phenological relationships in addition to the adaptation and acclimation mechanisms deployed within the physically harsh and dynamically changing coastal environment.

Eelgrass meadows offer many important ecosystem services that include keystone habitat for fisheries (Plummer et al. 2012), sinks for carbon and nutrients (Duarte et al. 2005, Fourqurean et al. 2012, Greiner et al. 2013), and coastline

protection from storm surge and erosion (Orth et al. 2006, Koch et al. 2009). It is important to understand the degree to which changes in climate will affect their distribution and abundance. Despite high scientific confidence that global mean sea surface temperatures will continue to rise during the 21st century and affect coastal ecosystems (NOAA 2016), species specific seagrass data on growth, development, and reproduction under these conditions are scarce. Additionally, as Kendrick et al. (2012) emphasize, seagrass researchers have historically focused on clonal reproduction (i.e. Olesen and Sand-Jensen 1994, Olesen 1999, Harris 2006), leaving the role of sexual reproduction in population dynamics largely ignored. Success over the past decade in eelgrass restoration and bed recovery via broadcasting seeds and seedling establishment has helped shift some focus to the role of sexual reproduction and seedlings as an important life-history stage (Orth et al. 1994, 2000, 2006a, Jarvis et al. 2014).

Seagrasses have historically experienced changes in temperature, sea level rise, carbon dioxide, and disease over the past 100 million years. It is the accelerated rates of change for these conditions projected with future climate change that motivates research into seagrass ecology (Orth et al. 2006). The rate of change matters because it takes time for individuals to adapt in order to survive under new, altered conditions (Davis and Shaw 2001). If the conditions change too drastically and multiple environmental thresholds or 'tipping points' are reached at once, then this may lead to a large-scale die-off event and ultimately either a shift or narrowing in the species' biogeographic range. There is broad evidence, however, that *Z. marina* is highly plastic to changes in its environment (Hemminga and Duarte 2000, Duarte et

al. 2006), strategically allocating growth to either the roots or shoots or altering its own reproductive strategies (Jarvis et al.2014). Work to predict its response to a changing environment must consider how rapidly these strategies can effect change to prevent broad changes in distribution and die-offs.

One well-documented example of an altered reproductive strategy used to cope with increased water temperatures is the emergence of *Z. marina* "annual meadows", such as those documented in North Carolina (Jarvis et al. 2012). While the individuals are not genetically annuals, they are able to persist amid stressful summer temperatures and yearly vegetation dieback by shifting their reproductive strategy to survive as seeds during unfavorable times of the year (Kim et al. 2014). Under this strategy, the individuals depend on the yearly production of reproductive shoots, a viable sediment seed bank, and successful seedling establishment (Jarvis et al. 2012). In this temperature stressed environment, seagrass individuals must be able to allocate resources to the formation of reproductive shoots, flower, and set seed for the next year's recruitment before taxing water temperatures lead to vegetative shoot mortality. This contrasts with more northerly populations, where meadows are maintained by a much higher rate of clonal reproduction and expansion, with some sexual reproduction.

Warmer water temperatures have already been documented along the East coast of the U.S. and are projected to increase (NOAA National Data Buoy Center, Cole 2008). I hypothesize that eelgrass beds may begin to respond with reproductive strategies similar to those documented in North Carolina populations by increasing the rates of sexual reproduction and a heavier reliance on annual growth, recruitment,

and establishment of seedlings. According to an investigation by Furman (2015), the need to link recruitment mechanisms to coverage changes (meadow development and small-scale disturbance recovery) occurring at landscape scales continues to remain a challenge for seagrass ecologists and managers alike. This example further emphasizes the need to gain a better understanding of the impacts of reproductive biology, phenology, and whole plant ontogeny on the maintenance of existing meadows in the face of a changing climate.

A complication to assessing eelgrass meadow development and individual species response to climate change is that long-term, base-line observations of seagrass meadows are rare. Similarly, relatively little information exists regarding the effects of climate change over the appropriate spatial (local, individual lagoons/estuaries) and temporal scales (years and decades) to support adequate species-specific climate assessments (Hemminga and Duarte 2000). It is also difficult to attribute changes to climate effects alone as coastal habitats have been impacted by anthropogenic activities, such as eutrophication, for decades (Nixon 1995, Hemminga and Duarte 2000, Kemp et al. 2005, Ralph et al. 2007, Duarte et al. 2015). These populations have already been impacted by changing climate, so an argument can be made that our ability to determine cause of changing population distributions is needed now to de-couple the interacting effects of climate and eutrophication on distribution patterns.

To help address some of the gaps and unknowns in *Z. marina* ecophysiology, I explored the use of an eelgrass mathematical model. While there have been several beneficial efforts to numerically synthesize seagrass knowledge into varied modeling

frameworks since the 1980s (Table 1), the majority of these studies allocate very little discussion towards *Z. marina*'s ontogeny, morphology, phenology, and ecophysiology from seed to seedling to reproductive adult. Seagrass productivity and growth has also often been modeled as static" black boxes" of carbon. The limitation in this fixed assumption is that it disregards many of the key internal interactions and driving forces within such a dynamic ecosystem and with such a plastic species (Harris 2006). As Fazlioglu (2016) points out, "the interplay between growth form, competitive ability, and reproduction is fundamental in understanding how plants cope with and evolve in competitive environments." These within system interactions are what help to explain why *Z. marina* responds the way it does to changing conditions.

One seagrass modeling platforms that moved beyond the static "black box" assumptions was developed by Harris (2006) and links eelgrass population dynamics to a mechanistic, physiological formulation of eelgrass growth within an Individual Based Model (IBM). Grimm et al. (2006) explains that IBMs are important for both theory and management since "they allow for scientists to evaluate aspects usually ignored in analytical models: variability among individuals, local interactions, complete life cycles, and in particular individual behavior adapting to the individual's changing internal and external environment". While the Harris (2006) model begins to understand these relationships by using a hybrid IBM/rules based approach and highlighting eelgrass' clonal plant strategy and individual ramet plasticity, there are still some missing pieces to address, namely the juvenile life stages of *Z. marina* and recruitment by sexual reproduction.

Noting the literature gap in modeling *Zostera marina* sexual reproduction and recruitment, Jarvis et al. (2014) provides a robust example of a reproduction model that includes formulations for reproductive shoot production, seed production, seed-bank density, seed viability, and seed germination. The model by Jarvis et al. (2014) offers estimates for key parameters, which are derived from several sources and model calibrations (Silberhorn et al. 1983, Fishman and Orth 1996, Harwell 2002, Bintz and Nixon 2001). The Jarvis model does not, however, track seedling above and belowground biomass through the first year of growth citing a lack of information on seedling parameters. This is one area that my thesis expands upon.

In this study I present a new model, developed from the Harris (2006) eelgrass modeling platform which is referred to as the Virtual Eelgrass Meadow (VEMv.1). This new model (VEMv.2), developed using Simile v5.97 software (Simulistics Ltd.), incorporates both asexual (Harris 2006) and sexual reproduction (Jarvis et al. 2014), a complete suite of life history stages (seed, seedling, adult), a sediment seed bank, a new seedling growth formulation and set of biomass allocation rules that distinguish the juvenile seedlings from the adult ramets. Using an individual-based approach allows the model to incorporate these features in great detail. The VEMv.2 is, therefore, a suitable tool to use in this study to perform virtual experiments evaluating the effects of climate change on the reproductive phenology of eelgrass. The new model is presented using the "ODD [overview, design concepts, and details] protocol" (Grimm et al. 2006). This standardized format for describing individual-based models helps modelers overcome the challenge of finding a way to present and publish their work in a more consistent and complete framework. The development of

this protocol signifies a large step forward in the study of individual based ecology as it allows ecologists to adopt common modeling concepts and terminology (Grimm and Railsback 2005). Specific elements included in the ODD protocol are outlined in Table 2.

Using the VEMv.2, we investigate the phenotypic plasticity of *Z. marina* under model scenarios of current and increased temperature within three geographically distinct locations using latitudinal gradients, as described by Frenne et al. (2013), as a proxy for climate change. The objective of this study is to use the model to test the impacts to eelgrass biology and reproductive phenology under current and future warming temperature scenarios. Model results are focused on changes in total and individual biomass and the timing of major reproductive events within the context of a 4-year "seeding" experiment bounded spatially within a 1.6m^2 mesocosm. Findings from the virtual seeding experiments will help scientists better understand the physiological ecology of seagrasses and help coastal managers to better anticipate eelgrass' response and resilience under current and future scenarios of climate change.

2. METHODS

2.1 Study Organism

Zostera marina L., commonly referred to as eelgrass, is a clonal marine hydrophyte adapted to temperate climates and coastal estuarine environments worldwide (Short et al. 2001). Subject to both intense seasonal and interannual dynamics (Cabello-Pasini et al. 2002), eelgrass grows in shallow water, typically less than 3 meters in depth in the U.S., and requires minimum salinities of 10 and sandy to

partially muddy substrates (Murphy et al. 2011). Eelgrass morphology is characterized by a creeping underground rhizome with nodes, internodes, roots, and aboveground vertical ribbon-like leaves that arise from the leaf sheath just above the basal meristem (Figure 3). The meristems are where active cell division (i.e. growth) takes place in the plant. In addition to the basal meristem located below the leaf sheath, the apical meristem is responsible for horizontal growth of the underground rhizome leading to the development of a new clone (Hemminga and Duarte 2000).

The perennial angiosperm reproduces both asexually by rhizome and sexually by seed to form monospecific meadows consisting of daughter ramets and seedlings (Les 1988, Hemminga and Duarte 2000). Eelgrass inflorescences are monoecious and pollen is distributed by water currents (Cox et al. 1992, Ackerman 2006). Light availability (Dennison et al. 1993) and water temperature are the two main direct drivers of species productivity, growth, and reproduction (Hemminga and Duarte 2000, Lee et al. 2007).

Z. marina meadows function as ecosystem engineers (Bouma et al. 2005, van der Heide et al. 2012) where bio-physical feedbacks enhance local sedimentation, resulting in improved water quality, and stabilizing the seabed (Christianen et al. 2013). Eelgrass meadows are also highly productive ecosystems that store most of their production in the sediments as organic carbon, known as “blue carbon” (Mcleod et al. 2011, Greiner et al. 2013), or export to neighboring ecosystems. *Z. marina* is recognized as a valuable coastal resource deserving of conservation and restoration (Hemminga and Duarte 2000, recent others).

2.2 History of the Virtual Eelgrass Meadow (VEMv.1)

The first numerical eelgrass model was developed in 1975 and published by Short (1980). This pioneering model paralleled the traditional approach used to model phytoplankton growth where the production rate relies on a widely used formulation representing a photosynthesis-light curve (Brush et al. 2002, Kremer and Nixon 1978, Steele 1962). Several eelgrass models (i.e. Harris 2006, see also Table 1) with a similar empirically driven limitation factors combined with mechanistic equations for growth were further developed beyond after the Short (1980) efforts.

The Harris (2006) development of the VEMv.1 is a product of a thorough review and a purposeful representation of the eelgrass ecosystem linked together by several modeling techniques. Most notably, the VEM is unique in that it combines aspects of a traditional system dynamics model with the programming structure of an IBM, a model type that has grown in prevalence over the last two decades (Grimm et al. 2006, DeAngelis and Mooij 2005). IBMs are recognized for their ability to provide scientists with a virtual platform to model emergent properties from processes defined at the individual level (Harfoot et al. 2014, Railsback 2001). The VEM centers on an IBM approach with mechanistic formulations based on first principles combined with empirical limitation functions and “rules” to dictate resource allocation. Given the model’s blended structure, it allows the user to examine how individual plant physiology relates to landscape level processes (Harris 2006) in a manner that former eelgrass models have been unable to provide.

2.3 General Model Description (VEMv.2)

The VEMv.2, is a mathematical simulation model, in which an eelgrass meadow emerges from the recruitment, establishment, growth, and death of individual ramets and seedlings. The updated model is built upon the VEMv.1 (Harris 2006) and includes a modified sexual reproduction model developed from Jarvis et al. (2014), and a new seedling growth rate formulation and associated biomass parameterization. The VEMv.2 also consists of renewed programming rules that govern biomass resource allocation, colonization, neighbor interactions, and the incidence of asexual and sexual reproduction. Since the VEMv.2 simulates sexual reproduction, it tracks seed production, germination rates, and incorporates a sediment seed bank as well. The resulting model encompasses the variability of growth rates, morphology, physiology, and phenotypic plasticity observed across the ontogenetic life stages of *Z. marina* from seed to seedling to adult (Figure 4).

2.3.1 Process Overview and Scheduling

There are eight fundamental processes accounted for in the VEMv.2, which will be described in detail, and scheduled in the following order: (1) distinct seedling and adult ramet specific growth rates are computed, (2) these growth rates are translated into biomass, and (3) subsequently partitioned separately to either aboveground (leaves) or belowground (roots and rhizomes) biomass. Concurrently, the model dictates (4) the occupation of space and location of new shoots along growing rhizomes and in an expanding meadow landscape. As individual densities increase, (5) the model limits population size via a set of self-thinning rules. Finally, (6) the production of offspring via asexual reproduction (i.e. the creation of daughter

ramets) is tracked in addition to (7) the incidence of sexual reproduction (flowering, seed production, seed dormancy and germination) and subsequent (8) seedling recruitment and establishment in the model. These processes are also visually represented by the conceptual diagram (Figure 5).

2.4 Understanding Eelgrass Ontogeny and Morphology for Model Application

2.4.1 Eelgrass Life Stages

Prior to model formulation, I needed to define the three main ontogenetic life stages of *Z. marina* represented in the model (seed, seedling, adult), both botanically and functionally, so that operational model definitions were explicitly characterized in our current and in future growth models. One immediate challenge with this exercise was that in botany the terms “seed”, “seedling”, and “adult” carry no hard botanical description. According to Webster's Seventh New Collegiate Dictionary (1967), for example, a seedling (botany) is “a young plant grown from seed”. For the purposes of model application, this definition was neither specific nor informative enough to be effective. Furthermore, a literature review (Table 3) of the term “eelgrass seedling” confirmed multiple inconsistencies in defining the seedling life stage using both chronological age and biomass measurements as indicators.

In-depth botanical descriptions of *Z. marina* development and structure by Taylor (1957) are shown in Figure 6. The precise morphological and cellular descriptions of the species’ life stages occurs at a level of resolution that is too fine for the scale at which the VEMv.2 was formulated, and associated calibration and validation data are not available to distinguish these ontogenetic phases. A seagrass seedling is the earliest part of the juvenile phase; typically, no more than a few weeks

old, usually where evidence of attachment to the seed is still present (personal communication D. H. Les, August 2014).

For this thesis, I chose to designate the ramet plastochron interval (PI), or the first instance of clonal reproduction (i.e. fecundity) after an individual had acquired four nodes, as the point of transition or biological threshold separating an eelgrass seedling from adulthood. Fecundity is under both genetic and environmental control and is a measure of fitness, providing a clear biological threshold separating the two life stages in the model. Furthermore, the PI threshold is intimately connected to the rate and magnitude of leaf production (Harris 2006) as opposed to chronological age. Terrestrial botanists have also long understood the principle that a plant's ontogenetic life stage or "biological age" (Robbins 1957) is a better measure than calendar age (Gatsuk et al. 1980). For these reasons, the subsequent creation of new internodes in the model appropriately allows it to act as a biological age clock. The biological feedback loop is visualized in Figure 7. For model purposes, once a growing seedling accumulates four or more nodes (the average eelgrass ramet PI; Harris 2006), then the "seedling" is considered an "adult" in the model and subject to the adult specific growth rate formulation and adult biomass allocation rules.

Lastly, the operational model definition of "seed" used in the VEMv.2 is a state variable produced from the product of the flowering shoots variable and a parameter setting the number of seeds produced per flowering shoot. The flowering shoots variable is calculated by summing how many total shoots flowered on the first flowering day. To obtain the total number of seeds produced each year, the seeds are summed and stored in a sediment seed bank until the seeds lose their viability (i.e.

rot) or germinate. While stored in the seed bank, only a portion of the original seeds survive through the summer dormancy period until temperatures fall below 20°C and trigger the remaining stored seeds to germinate and recruit into the population as new seedling individuals. A list of the VEMv.2 sexual reproduction model rules, which were formed using a modified version of the sexual reproduction model developed by Jarvis et al. (2014), are shown in Figure 8. Specific details of the reproduction model and the work by Jarvis et al. (2014) are explained in Section 2.7 *Modeling Reproduction*.

2.4.2 Understanding Eelgrass Ecophysiology for Model Application

In the context of eelgrass ontogeny and morphology, ecophysiology - the manner in which an individual functions in its environment is an important underlying mechanism to understand prior to model development and parameterization. Generally, plant ecophysiology describes the relationships of the essential resources of light, water, temperature, and nutrients in the context of varied climate conditions. This field of study dates far back to the work of Theophrastus, an ancient Greek native c. 300 BC, who took extensive notes of how seed nutrition, germination, and plant growth are affected by the growing environment. For the purpose of this thesis, I incorporated known relationships regarding eelgrass ecophysiology into the model development and parameterization, laying the groundwork for evaluating the species physiological adaptation and acclimation mechanisms related to changing conditions in its environment.

Similar to terrestrial angiosperms, three essential resources that affect eelgrass growth and survival are light, temperature, and nutrients. Light and temperature are

two major factors affecting growth. We know that photosynthetic rates in seagrasses increase linearly with light to a saturating level (Falkowski and Raven 2007) and that light limitation determines the depth limit of seagrass growth in addition to shifting biomass allocation to leaves from the rhizomes (Hemminga and Duarte 2000). This reduces the maintenance and development of non-photosynthetic tissues and maximizes the formation of leaves, which in turn enhances light absorption. This relationship was incorporated into biomass allocation rules in the model.

Temperature has a significant effect on eelgrass reproductive phenology (Moore et al. 2003, Moore and Orth 1982, Setchell 1929). Under rising temperatures scenarios, *Z. marina* is known to increase its metabolism in addition to its rates of leaf respiration (R) relative to photosynthesis (P) (Short and Neckles 1999). The decreasing P:R ratio and high metabolic state will continue as the plant responds to thermal stress until the upper limit of stress tolerance is reached where productivity rapidly declines, frequently resulting in mortality.

Rates of flowering in clonal seagrasses also typically increase with increased disturbance or stressful conditions, such as high water temperature (Kim et al. 2014). For this reason, I incorporated this relationship during parameterization of sexual reproduction in the model using a temperature threshold for accumulated “thermal stress days”. The threshold was set at 20 degrees Celsius, the average temperature with which respiration begins to increase beyond productivity in *Z. marina* individuals (Marsh et al. 1986). Another way of relating this type relationship is through one of the oldest and most studied ecological concepts, “growing degree days” (GDD), where an accumulated degree-day phenological interaction (largely

temperature and/or light) drives a particular developmental event (Idso et al. 1978). This relationship has also been applied outside the field of plant ecology, such as with blue crab growth in the Chesapeake Bay (Brylawski and Miller).

Despite some conflicting evidence regarding the effects of nutrients on eelgrass in the literature (Harlin & Thorne-Miller 1981, Dennison et al. 1987, Zimmerman et al. 1987), we know that both leaves and roots/rhizomes can uptake nutrients from the water column and the sediments (Hemminga and Duarte 2000). In young seedlings the main mechanism for nutrient uptake is through the roots (Hemminga et al. 1994). Eelgrass also has an impressive tolerance to high porewater nutrients (Hemminga and Duarte 2000). There are secondary effects of nutrient enrichment of the water column on eelgrass, which include increased macroalgae and epiphytic leaf algae shading the available light to the individual, especially as temperatures increase (Harline & Thorne-Miller 1981). In terms of eelgrass reproductive phenology, there is some conflicting evidence on the effects of nutrients. Sediment nutrients have been found to be inversely correlated (Short 1983) with the percentage of flowering shoots, while concentrations of pore water nutrients (ammonium) have been found to be positively correlated (Johnson et al. 2017).

In previous eelgrass enrichment experiments, it has been shown that eelgrass elongates its leaves in response to higher nitrogen concentrations (Roberts et al. 1987). For this reason, we included a multiplicative factor that takes the predicted weight of the longest leaf and increases or decreases these limits under high or low nitrogen conditions, respectively. The regression equation used to capture this relation in the model was taken from work performed by Roberts et al. (1987).

2.5 Model Formulation

The model was developed using Simile v5.97 software (Simulistics Ltd.), similar to STELLA, but with added capabilities to construct object-oriented programs (see visual: Figure 19) as implemented in its base language C++. The model code can be found in Appendix A. Parameter estimates were selected using values from the literature and eelgrass field calibration data for adult ramets and seedlings displayed in Table 4. Governing equations for eelgrass growth included two separate specific growth rate formulations (Table 5), one for adults and one for seedlings. Resource allocation of biomass was specified through programmed "rules" that apportion growth to above and belowground plant structures. Additionally, the rules governing ramet branching and bed expansion paralleled architectural rules governing the plant in nature and were adopted from Harris (2006). Asexual and sexual reproduction occurred in the model via clonal growth of new ramets and anthesis, seed development, and germination. When individuals transitioned to adulthood, they were subsequently subject to the self-thinning phenomenon where the model used an individual's local shoot density within a 10 cm^2 area to create a feedback that affected the down welling light availability of that individual (Harris 2006). This density dependent rule captures the competition for light resources during crowded conditions and limits production with some cases resulting in mortality.

2.6 Modeling Eelgrass Growth

Eelgrass growth in the VEMv.2 is driven by two specific growth rate formulations, which are constructed and parameterized separately for adult ramets and seedlings. Specific growth rate (μ) in the context of the model is defined as the

mass increase per gram of biomass per day in units of grams dry weight per gram dry weight per day (gDW gDW^{-1} d^{-1}). The growth rate equations include a specific growth rate maximum (μmax), parameterized from eelgrass literature values (see Table 4 for data sources), and limitation functions of light (I), temperature (T), and sediment sulfide (S). The resulting relationships are generally represented by governing equation 1.

$$\text{(eq. 1)}\ \mu = \mu_{max} * f(I) * f(T) * f(S)$$

The following formulations focus primarily on those pertaining to seedling growth, which this thesis was focused on and while some general information is provided for the adult formulations, specifics can be found in Harris (2006).

2.6.1 The Adult Specific Growth Rate Formulation

The adult specific growth rate formulation (eq.1a) is described in Harris (2006) and adapts the rectangular hyperbola model typically used to model photosynthetic rates from Photosynthesis-Irradiance (PI) curves (e.g. Baly (1935)) with parameters developed from Olesen and Sand-Jensen's (1993) Growth-Irradiance curves. Each of the input parameters (μmax, alpha, I, and ro) were correlated with temperature using eelgrass data collected by Olesen and Sand-Jensen (1993). The adult sediment sulfide (S) limitation was formulated using an "if-then" statement from data published in Goodman (1992) in addition to Olesen and Sand-Jensen (1993).

$$\text{(eq. 1a)}\ \mu Adult = \left(\left(\frac{\mu_{max}\alpha I}{\mu_{max}+\alpha I}\right) + ro\right) * f(S)$$

(eq. 2) $Adult\ S\ Limitation = If\ S \leq 55.45\ then\ 1\ elseif\ S \geq 2000\ then\ 0\ else\ 13.6 * S^{(-0.65)}$

2.6.2 The Seedling Specific Growth Rate Formulation

The new seedling specific growth rate model was developed separately from the adults using a Jassby-Platt formulation (Jassby & Platt 1976). This formulation was developed originally as a photosynthesis-light (PI) curve for phytoplankton, and parameterized with eelgrass seedling data by Abe et al. (2008). Given that associated seedling growth data was already parameterized to this Jassby-Platt formulation, it was more suitable to use for the seedlings than the adult formulation. The equation includes a maximum specific growth rate (μmax) designated from the literature, which describes the rate at which seedlings will grow under optimal conditions. The μmax is then limited by a derived and normalized interactive function of light and temperature, and a sediment sulfide toxin limitation factor (eq. 3).

(eq. 3) $\mu Seedling(I,\ T, S) =\ \mu_{max} * f(I,T) * f(S)$

In equation 3, μ denotes the specific growth rate, a measure relating the relative change in biomass over time, I represents underwater irradiance at the canopy depth (μmol m^{-2} s^{-1}) and T denotes temperature (°C). Irradiance at depth was calculated using Beer-Lambert's Law where a modeled attenuation coefficient, k, was applied to the water depth (z) and measurements of incident photosynthetically active radiation (PAR) as I_o:

(eq. 4) $I_z = I_o * e^{-kz}$

The time step in the model (and therefore the growth formulation) is one day, allowing for daily changes in photoperiod and average light intensity and temperature to alter the growth rate of seedlings each day out of the year. The limitation function within the governing equation (eq.5) is dependent on variables of light and temperature and is derived using published seedling data (Abe et al. 2008) fitted to a hyperbolic tangent function originally described by Jassby & Platt (1976).

$$\text{(eq. 5)}\quad P(I) = P_{max\,g} * \tanh\left(\frac{I}{I_k}\right) + R$$

This is an averaged photosynthesis-irradiance curve, similar to those used to model phytoplankton production, where P(I) indicates net photosynthesis at irradiance (μL O_2/cm^2 per h), $P_{max\,g}$ is the maximum gross photosynthesis (μL O_2/cm^2 per h), I_k is light saturation (μmol m^{-2} s^{-1}), I is irradiance (μmol m^{-2} s^{-1}) and R is respiration (units). Temperature proxies in the form of linear regressions were derived to represent the values of I_k, R, and P_{max} to fit the Abe et al. (2008) seedling photosynthetic rate data across changing temperatures of 5°C-35°C. Only the seedling photosynthetic rate data from the final day 6 of the Abe et al. (2008) experiment were used as measurements prior to this day suggested that the plants were still undergoing thermal acclimation to the temperature treatments.

The resulting limitation equation (eq.6) was normalized between 0 and 1 by dividing the modified Jassby & Platt (1976) equation by the net photosynthesis maximum, which occurred at the seedling's presumed physiological optimum temperature of 25 °C and at a saturating irradiance value of 250 μmol m^{-2} s^{-1}(Abe et al. 2008). The normalized and unit less limitation function was then multiplied by the selected μmax of 0.03 gDW gDW^{-1} day^{-1} to determine the specific growth rate of an

individual seedling for a particular day. This normalization permits the use of photosynthesis data to formulate the light limitation function, with the assumption that net primary productivity is similarly limited in the form as net growth. The limitation equation uses an "if, else" statement in the model due to an optimum temperature threshold at 25°C, which substitutes into the f(I,T) portion of the seedling specific growth rate formulation (eq. 3) above.

(eq. 6) $If\ T \leq 25\ then$

$$\frac{\left(0.97T - 0.75) * \tanh(\frac{I_z}{4.59T + 3.34}\right) + (-0.949T - 1.0503\right)}{20.02}$$

$$else$$

$$\frac{((-2.8T + 94.3) * \tanh(\frac{I_z}{4.5978T + 3.347}) + (-0.949T - 1.0503))}{20.02}$$

The specific growth rate maximum (μmax) in the governing equation (eq. 3) is an example of one important input parameter requiring careful selection due to its potential influence on model output since μmax describes the rate at which *Z.marina* will grow under optimal light and temperature conditions. Parameterization of this value and others will be discussed in greater detail later in the methods.

The seedling sediment sulfide (S) limitation displayed in equation 7 was calculated using data from field and laboratory experiments performed by Dooley et al. (2013) on eelgrass seedlings and normalized between 0 and 1.

(eq. 7) $If\ S \leq 1\ then\ 1\ elseif\ S \geq 2000\ then\ 0\ else\ 1.0239 * e^{(-0.002*S)}$

In this equation, S represents porewater sediment sulfide concentrations in units of µM of hydrogen sulfide (H_2S).

2.7 Modeling Reproduction

2.7.1 Asexual Reproduction

Asexual reproduction or clonal growth is linked to the rate and magnitude of leaf and node production. Each time an old leaf is sloughed off in the model a new node is “born”. In order for this process to occur, the model checks to see if the compartment for "Oldest Leaf Mass" was larger on the last time step than on the current time step. If it is lower than the last time step, then that compartment has been emptied and the model is instructed to create a new node. Once four nodes have been acquired since the last lateral shoot, the model creates a new lateral shoot, hence a new ramet individual (Harris 2006) which signifies adulthood.

2.7.2 Sexual Reproduction

Sexual reproduction was programmed using a rules based approach (Figure 8) with procedures and parameters modified from the eelgrass reproduction model developed by Jarvis et al. (2014). The Jarvis et al. model (2014) highlighted the need to include sexual reproduction within traditional *Z. marina* production models as model output could more accurately predict loss and recovery processes (Jarvis et al. 2014). The reproduction model influences the timing and magnitude of sexual reproductive events, including the production of flowering shoots, seeds, germination, and seedling establishment. Silberhorn (1983) published a valuable synthesis documenting reproductive phenology at a number of locations along a latitudinal gradient (Figure 9). Despite differences in the time of year and latitude, the observance of key reproductive events (inflorescence primordia, anthesis, and mature fruit) across four different investigations were all intimately tied to temperature

(Silberhorn 1983). These sexual reproduction-temperature relationships were also confirmed and documented by Moore and Orth (1982; Table 7).

One aspect not detailed in the Jarvis et al. (2014) model but incorporated in the VEMv.2 is the "growing degree days" (GDD) concept of thermal stress day accumulation. Referenced earlier in section 2.4.2, the VEMv.2, counts how many days in the growing season are above the 20°C stress parameter. The model then uses the number of days in the year above 20°C to set the biological threshold and temperature proxy to determine the probability that an individual will flower. This conditional flowering probability is a hybrid mechanistic-stochastic parameter that determines if an individual should flower and is linked to temperature, time of year, number of nodes, and overall probability. Each year, individuals are assigned either a 30%, 50%, or 90% chance of flowering based on the number of thermal stress days accumulated over that year. If there were <=100 days with temperatures over the 20°C then an individual's probability of flowering is 30%. If there were >= 150 days then the probability is 90% and if there was any amount in between then the probability of flowering is 50%. The probability of flowering is a value that affects the individual ramet and should not be confused with the percent of total individuals who flowered within the meadow during a given year.

Overall, setting these threshold parameters, both percent and days, were challenging. The 100 day and 150 day temperature stress thresholds were selected based on a literature review of average temperatures against observed rates of reproductive shoots. Information supporting these thresholds, however, are still lacking and there are inconsistencies in literature data with respect to eelgrass

flowering and temperature. Still, several studies have shown that locations with higher water temperatures typically experience higher rates of flowering (i.e. North Carolina “annual” meadows) (Jarvis et al. 2012, Kim et al. 2014). Percentages of reproductive shoots found within eelgrass meadows have been reported between 0-28% of total shoots (Jacobs & Pierson 1981, Silberhorn et al. 1983, Thayer et al. 1984, Olesen 1999), all the way up to 100% of total shoots flowering (Keddy & Patriquin 1978, Robertson & Mann 1984, Meling-López & Ibarra-Obando 1999, Jarvis 2012).

The flowering probability is applied once the optimal anthesis temperature is reached, a condition that is met when spring water temperatures increase from a daily average of 14°C to 15° C. This temperature threshold identifies the first flowering day (FFD), a time point that is declared in the model and at which time eligible shoots flower. It is important to note that a seedling with three or more nodes also has the ability to flower in the model, even if it is less than one-year-old, as there is evidence that older seedlings can flower in their first year of growth (Jarvis et al. 2012, Johnson et al. 2017). The number of shoots that flower on the first flowering day are then multiplied by the average number of seeds per shoot (10) and placed in the model’s sediment seed bank. While in the seed bank, the seeds remain dormant through the hot summer months until water temperatures decrease below 20° C and near 15 ° C (Moore et al. 1993). Only 40% of the original seeds that enter the seed bank remain viable to germinate after the period of dormancy (Jarvis et al. 2014). Once germinated, the individuals are considered newly “born” seedlings or individual recruits into the model and are assigned spatial coordinates. These young seedlings

grow according to the seedling specific growth rate formulation and seedling biomass allocation rules until transition to adulthood.

2.7 Study Locations

Model simulations were run in three locations along a latitudinal gradient of the East coast of the United States (Figure 10). The sites were located at Great Bay, New Hampshire (NH), South Bay, Virginia (VA), and Phillips Island, North Carolina (NC). Frenne et al. (2014) describe using latitudinal gradients as a proxy for climate change: "[latitudinal gradients are] an excellent natural laboratory to investigate the role of temperature and the potential impacts of climate warming". Many studies, including previous eelgrass investigations, have taken this approach as well (Silberhorn 1983 and Clausen et al. 2014). Specific simulation sites were selected based on availability of water quality and meteorological data, nearby presence of eelgrass, and location. Model calibration and validation sites were located in Narragansett Bay, Rhode Island, and the York River, Virginia (see below).

2.8 Model Forcing Functions, State Variables, and Parameterization

The model was forced with data controlling the physical setting (nutrients, water depth, sediment), the light setting (surface PAR, k, Iz) and the temperature setting (Table 6) all averaged by day during 2007-2009, with one substitution using 2013 temperature data in South Bay, VA from the Virginia Estuarine and Coastal Observing System (VECOS) due to 2007-2009 missing data (Figures 11 and 12). The model simulation sites in NH and NC were forced with water quality and meteorological data from the National Estuarine Research Reserve System (NERRS)

System Wide Monitoring Program (SWMP), notably water temperature and surface PAR, which are key model input forcing functions. For Virginia's South Bay site, adjacent NERRS Taskinas Creek surface photosynthetically active radiation (PAR) measurements were used since light data in South Bay were unavailable. Water temperature was recorded in degrees Celsius and averaged by day for model assimilation.

For the light setting, 15-minute interval PAR measurements were averaged by day using the known daily photoperiod and converted into units of Photosynthetic Photon Flux Density (PPFD) or uMol m^{-2} day^{1}. PPFD is a unit of measurement often used in plant biology to quantify the number of photons in the 400-700 nm range experienced by a surface over a specified amount of time. Light attenuation coefficient (k_d) values were also forced using modeled output from the Lagoon Ecosystem Model (LEM) developed by Brush 2014 and applied in Hog Island Bay, VA.

The key state variables accounted for in the model include the above and belowground biomass characteristics, the computed specific growth rates, and the plant traits modeled to change, such as an individual's number of nodes, and reproductive qualities, such as the total number of seeds produced. Model parameters were selected from the literature and field data and evaluated during model calibration (Table 4 and 6) to produce the best fit within ecological limits.

2.8.1 Seedling Specific Growth Rate Parameterization

The seedling specific growth rate maximum (μmax) in the seedling specific growth rate governing equation (eq. 2) is an example of one important parameter

requiring careful selection due to its potential influence on state variables in the model. The challenge with selecting model parameters such as μmax, in this case, is that empirical measurements of seedling biomass specific growth rates are not frequently measured. Additionally, when specific growth rates are measured they are not always reported in standardized or easily comparable units. Converting across units and measurement methods in an attempt to estimate biomass output further decreases our confidence in its selection.

Despite only a few values reported for seedlings (Bintz and Nixon 2001, Harris 2006, Rasmussen et al. 2012), μmax rates for seedlings and adult eelgrass individuals are shown to be fairly similar in literature studies. I selected a μmax of 0.3 gdw gdw^{-1} day^{-1} for the seedling growth rate sub model within the VEMv.2 (Bintz and Nixon 2001, Rasmussen et al. 2012). The surface plot displayed in Figure 13 shows the effect of changing light availability and temperature from model output on changes in a seedling's specific growth rate. The graph clearly shows the optimal temperature zone for growth at 25°C and the point at which light reaches saturating levels, approximately 250 μMol m^{-2} s^{-1}. The graphs also depict the rapid decline in growth rates with a temperature increase from 25°C to 30°C.

2.9 Model Calibration and Validation

Since the adult specific growth rate and clonal reproduction components in the model were previously parameterized and tested against field data by Harris (2006), the focus of this thesis is on eelgrass sexual reproduction and seedling parameterization, calibration, and validation. With IBMs as large and as complex as the VEMv.2, it is simply not possible to calibrate and validate all parameter values

within a realistic time frame. In addition, the availability of data is also a limiting factor in performing additional calibration and validation exercises. For this reason, calibration and model verification focuses on the key parameters and processes that control growth and biomass allocation.

2.9.1 Calibration

Prior to performing the initial model runs, calibration was considered as a possible step in model testing. Calibration determines if certain key parameters need adjusting before proceeding to model validation. This was important since I wanted to ensure the model was robust enough to be applied at varying latitudinal gradients, which called for an iterative model review process. Upon testing preliminary calibration results, it was evident that comparisons with empirical data did not require significant changes to model parameters or formulations. Instead, validating model results against available data from different latitudes seemed critical given the interest in testing effects of temperature differences on eelgrass. For this reason, most of my efforts in testing the model were spent evaluating the validation results and not on model calibration.

2.9.2 Validation

Seedling specific growth rate and biomass state variables in the VEMv.2 were directly compared against growth rate and biomass data collected by Bintz (2002) from a 12 week seedling mesocosm experiment in Narragansett, Rhode Island and Jarvis (2007) from Virginia's York River. The comparisons are displayed and discussed in the model validation results section, which intends to instill confidence

and provide transparency in the model's performance. Model forcing functions, parameter values, and initial conditions were defined in these simulation scenarios to replicate Rhode Island and York River conditions as provided in Bintz (2002) and Jarvis (2007), respectively. Since comprehensive seedling biomass and specific growth rate data are rare and limited in the literature, I had to rely on very limited data for model validation. I did, however, obtain morphological measurements of very young seedlings from Virginia's coastal lagoons (Table 8 and Figure 26, Orth and Brush 2015) to use as a reference for setting the model's initial seedling biomass conditions. In an ideal scenario, the model would be calibrated and validated using multiple inputs and independently verified by data from varying sites.

2.10 Sensitivity Analyses

With the VEMv.2 as large and as complex as it is, it was unrealistic to test all state variables against changes to key model parameters. For this reason, I chose to test the sensitivity of two state variables, total biomass and the first day of adulthood, to a +/- 5%, +/-10%, and +/-20% change in the seedling maximum specific growth rate or μmax parameter. I chose the variables and the μmax parameter as they are both essential components in the model and for the simulation trials. The goal for performing the analysis was to assess the degree to which the model output changed per changes in μmax. All of the sensitivity simulation runs began on January 1, 2007, continued for 120 days, and were forced with conditions representing Virginia's South Bay.

2.11 Model Simulations

Model simulations of the VEMv.2 were performed to carry out virtual seeding experiments isolating *Z. marina's* response under warming temperatures that would otherwise take years to simulate in field conditions. Incorporating a latitudinal gradient approach as a proxy for climate change, I selected the three simulation sites, detailed previously, in Great Bay, New Hampshire (NH), South Bay, Virginia (VA), and Masonboro, North Carolina (NC).

Two years of forcing function data for the three sites were looped through twice to obtain a total of four years of forcing function data. I focus on results from the first half of these results. Four different temperature simulation scenarios, lasting four years each, were performed at each of the three locations. The temperature scenarios included one representing ambient water temperatures from 2007 and three under projected water temperatures representative of years 2025, 2050, and 2090. For the total areal biomass simulation results, year 2025 was omitted due to its proximity to year 2007 (ambient conditions). Standard to many simulation models, the VEMv.2 benefits from model initiation run up time, and in this context, only 1.5 years of model output is used for interpretive purposes.

The model simulations were run using Simulistics Ltd. software v5.97. They were performed on a personal computer with an AMD Athlon™ II X2 250 Processor, 3.00 GHz clock frequency and 16.0 GB RAM. Technical assistance in debugging the model and cutting down on model run time was provided by Jasper Taylor (Simulistics Ltd.).

2.11.2 Projected Temperature Simulations

The projected temperature forcing functions were derived using Victoria Cole's (2008) monthly Chesapeake Bay sea surface temperature (SST) anomalies added onto observed temperature data specific to the local sites chosen (Figure 14). The monthly anomalies were averaged over the spatial area of the Chesapeake Bay watershed by Cole (2008) and reported for years 2025, 2050, and 2090 using the 20th century data as the baseline. In order to compute the scenario anomalies, Cole (2008) performed regressions between the IPCC AR4 model surface air temperature and ambient Chesapeake Bay sea surface temperatures. The resulting climate projections were based on 20 of the best performing climate models freely available in the IPCC AR4 archive (Solomon et al. 2007) maintained by the Program for Climate Model Diagnosis and Intercomparison (PCMDI).

For application in our model simulation runs, Cole's monthly temperature anomalies for 2025, 2050, and 2090 were added to the 2007-2009 ambient water temperature forcing functions to represent the effect that climate change would have on the three simulation sites along the East coast (Cole 2008) (Figure 15). To maintain consistency across the three sites during the simulations, all other forcing functions and input parameters remained constant (Figure 12). The assumption in applying the anomalies in this way is that locations outside of the Chesapeake Bay region will experience similar trends and rates of projected warming. With a lack of downscaled, regional climate and, specifically water temperature, projections for New Hampshire and North Carolina in order to confirm this, the assumption is open to criticism. Please note that in final revisions of this thesis, an error was noted in

parameterization of the New Hampshire temperature data that is being corrected for peer-review manuscripts.

2.11.3 Experimental Simulations

To carry out the simulations, I chose to set up virtual seeding or restoration experiments by initializing only seeds at the beginning of a four-year period at each of the three locations and over the four temperature regimes (current, 2025, 2050, 2090). 100 viable seeds were added to the model's sediment seed bank and restricted to a 1.68 m^2 area, the same size of the Rhode Island experimental mesocosm tanks whose data was used during model validation. Given that the total sediment area of the virtual mesocosm was 1.68m^2, the initial starting seed bank density was 60 seeds m^{-2}. Seeds, however, were not assigned spatial characteristics in the model. Individuals only obtained spatial coordinates upon successful seed germination, a recruitment process in the model.

As one of the model's sexual reproduction rules (Figure 8), 40% of viable seeds in the seed bank germinate. This means that 40 of the 100 viable seeds in the seed bank would germinate in the 1.68m^2 plot, accounting for an approximate germinated seedling density of 24 seeds m^{-2} in the model simulations. These values are in line with those recommended for large scale eelgrass restoration efforts (Orth et al. 2007) and are representative of densities (i.e. 12.5-50 seeds m^{-2}) that have formed successful plots in previous restoration projects, namely the work in Virginia's coastal bays (Orth et al. 2006d).

Upon germination in the simulations, new seedlings emerged and grew according to the seedling specific growth rate formulation and seedling biomass rules

limited and controlled by the model's forcing functions and site parameterization. Once the seedlings amassed four nodes they transitioned to adulthood. The seedlings could also transition to adulthood by flowering when they had amassed 3 nodes and satisfied the conditional flowering probability. Field studies have validated this choice by showing that some seedlings will flower during their first year of growth (Jarvis 2012, Johnson et al. 2017). Once again, the flowering conditional probability is a hybrid mechanistic-stochastic parameter that decides if an individual should flower and is inked to temperature, time of year, number of nodes, and overall probability. As adult individuals, the ramets were then subject to the adult specific growth rate formulation and adult biomass allocation rules. Adulthood also initiated the density-driven "self-thinning" rule, which, as a local biomass control, limits the available light to adult ramets as population densities increase.

2.11.1 Model Assumptions

The experimental simulation runs included several simplifying assumptions. One main assumption we made during parameterization is that the eelgrass genotypes across the three sites all have the same morphological characteristics. We also assumed that the only forcing function that would change between the yearly simulations was temperature while we know that aspects of light availability, nutrient loads, sediment type, and even water depth would likely change in any of these locations between now and 2090. There were also other climate change related aspects not represented in the model, such as the effect of rising levels of carbon dioxide in the water column (i.e. ocean acidification). With the goal of applying the VEMv.2 over such a large geographical area (East coast of the U.S.) and the VEMv.2

as large and as complex as it is, I had to make these types of assumptions as to not misinterpret the results from too many changing environmental variables at once. Additionally, I recognize that with the VEMv.2 being able to make these types of broad scale comparisons comes the costs of more specific site parameterizations and model accuracy.

3. RESULTS AND DISCUSSION

3.1 Overview of Results

Results from the model calibration and validation indicate that the VEMv.2 does reasonably well in reproducing seedling specific growth rates and biomass. Results from the sensitivity analysis confirmed that the model's total biomass was sensitive to changes in μmax. Findings from the modeled climate simulations indicated an effect of temperature on the reproductive phenology and biomass of eelgrass, showing different latitudinal responses down the East coast of the U.S. as well as a shift towards reliance on sexual reproductive strategies for necessary meadow maintenance and survival under warmer temperatures. Understanding that IBMs are often difficult to analyze, understand and communicate specific results, I focused and organized results from the seeding experiment under three main sub-headings: impacts to biomass, impacts on reproductive phenology, and impacts on life history strategies.

3.2 Model Calibration and Validation Results

3.2.1 Calibration Results

Referenced previously in the methods, comparisons with empirical data did not require significant changes to model parameters or formulations ahead of model validation due to general agreement between model output and observed data. This agreement built confidence in the VEMv.2 and did not necessitate calibration of model parameters. Instead, validating model results against available data from different latitudes seemed critical given the interest in testing effects of temperature differences on eelgrass.

3.2.2 Validation of Specific Growth Rates

During the 12-week Rhode Island mesocosm validation run, the modeled average VEMv.2 seedling biomass output fit well against the observed average biomass values recorded in the Rhode Island experiment (Figure 16). A small divergence in the average biomass results was seen in June, although the VEMv.2 caught up with the Rhode Island observed biomass results by the end of the 12 week mesocosm experiment with an average of 0.268 gDW $shoot^{-1}$; nearly identical to the Rhode Island biomass of 0.267 gDW $shoot^{-1}$. Despite a small decline in average biomass in early August by the VEMv.2 due to stressful water temperatures, the average specific growth rates over the 12 week calibration in the VEMv.2 and in Rhode Island were also very similar, 0.0283 gDW gDW^{-1} d^{-1} and 0.027 gDW gDW^{-1} d^{-1}, respectively.

3.2.3 Validation of Seedling Biomass State Variables

Seedling biomass state variables were directly compared with seedling biomass data collected by Jarvis (2007) in Virginia's York River from April-July 2007. Results from the model validation are shown in Figure 17, which display the 2007 VEMv.2 average seedling biomass state variables against the 2007 observed averages and sampling day averages. The VEMv.2 underestimates the average biomass compared to the observed values but begins to more rapidly increase biomass in June once the seedlings transition to adults in the model. It is also worth noting the great variability among the individuals in the observed data representing the seedlings from that sampling day. This may be due to the fact that the York River seedlings were purposely sampled from the shallow-bed, mid-bed, and deep-bed locations within the meadow. That would likely have contributed to the larger variability in biomass sizes due to the availability of light.

By early June the York River seedlings were about a month ahead in terms of the average biomass weight than in the Rhode Island calibration. This was expected given the warmer temperatures in Virginia. There were no data reported on specific growth rates for the York River seedlings for model validation.

While no parameters were altered during model calibration, the importance of setting the correct initial conditions was highlighted by the Rhode Island (2002) and Virginia (2007) validation runs. Initial seedling biomass conditions, which included the aboveground and belowground biomass compartment values set at time=(0), were originally assigned using the young seedling biomass measurements (Table 8) taken by Brush and Orth (2015) and representative of seedling growth as of January 1 in a given simulation year.

In order for the VEMv.2 validation results to directly compare with the two observed datasets, I had to make sure that the starting biomass conditions in the model were similar to the starting conditions of the seedlings used in both Rhode Island and Virginia. For this reason, I manually updated the initial total seedling biomass values at time= (0) from 0.0023 gDW $shoot^{-1}$ to 0.041 gDW $shoot^{-1}$ in the model. This adjustment to the biomass initial conditions was made since the methods from both the Rhode Island mesocosm experiment and the York River study implied that the seedlings used in their 3-month investigations were neither young nor small like those reported by Brush and Orth (2015) and initialized in the VEMv.2.

In the Rhode Island study, Bintz states that "we thinned the plants [on May 15] to leave one seedling per pot before moving them to outdoor tank..." (Bintz 2002 Chapter 4 pg. 130). This thinning likely indicates that the so-called "seedlings" and associated data in Bintz (2002) are actually young adult plants as thinning would only be necessary if clonal growth had occurred. Per the VEMv.2 definition of a seedling, lateral shoot production is a process that signifies a plant's transition to adulthood. Secondly, under the Rhode Island mesocosm initial conditions and forcing functions, the VEMv.2 seedlings transitioned to adulthood on June 3, early on in the experiment, suggesting that it would have been entirely plausible for the Rhode Island seedlings to also have been adults by May 15. In fact, if I did not allow for the seedlings to transition to adulthood in the VEMv.2, the modeled biomass output would have significantly underestimated the biomass at the end of the 12 week calibration. This is because the specific growth rates of seedlings and adults are different, as I model in

the different formulations describing these critical rate processes as defined in equations 1 and 3 shown in Table 5.

Using both the Rhode Island data to validate growth rates and biomass, and the York River data to validate biomass, our results concluded general confidence in the model. It is likely that further simulations that adjusted initial conditions for seedling biomass, or attempted to represent some of the variability seen in field conditions in the simulated ramets by varying their initial individual biomass, might result in greater agreement to field conditions. In general, this validation exercise also underscores the lack of sufficient empirical data and studies focused on eelgrass seedling measurements and ecophysiology on top of frequent mischaracterizations of the seedling life stage in the seagrass literature. Both of these factors limit ability to further test the model, which would be improved by testing against a greater diversity of field measurements for both growth rates and individual biomass values.

Finally, it is entirely feasible that eelgrass individuals may be better represented by parameters specific to their local simulation sites and genotypes. As Hughes et al. (2009) report, *Zostera* genotypes showed significant differences to a variety of "ecologically relevant and morphological measures", including shoot production, belowground biomass, allocation of growth to leaves or roots, rhizome length, and rooting depth. It may be too difficult to apply the VEMv.2 broadly, across sites all along the Atlantic coast, with the same parameterization and expect the results to also be very site specific.

3.3 Results from Model Sensitivity Analysis

Results show that seedling biomass is sensitive to changes in the seedling specific growth rate maximum (μmax) parameter but not as sensitive as originally hypothesized (Figure 18). With total biomass (gDW shoot^{-1}) as the state variable, I tested its sensitivity to +/- 5%, +/-10%, and +/-20% changes in μmax. The simulations were performed over 120 days, the time at which all individuals were still seedlings, using forcing functions and input data from Virginia's South Bay. The percent change or sensitivity was relative to the parameterized 0.03 gDW gDW^{-1} day^{-1} μmax.

Complete results are displayed in Table 9, including those that detail changes in the day a seedling transitions to adulthood, which is slightly more sensitive to changes in μmax than total biomass. Results show that with a +/-20% change in μmax there is a 4% change in total biomass. This was a surprising realization in my analysis. In addition, there is a difference in the sensitivity of aboveground biomass versus belowground biomass to μmax. Belowground biomass is slightly more influenced by changes in μmax than the aboveground values. Finally, the most significant result I found during the sensitivity analysis was that with a 20% decrease in μmax came a 10% shift later in the year for the timing of the transition of a seedling to an adult.

3.4 Model Simulation Results and Discussion

For model climate simulation results, I focused on output from days 365-800. The first year of the simulation was a critical initialization period (Grimm and Railsback 2005), and is not presented or interpreted here. Output from days 365-800 permitted analyses of one year + of growth, biomass, and reproductive phenology

simulated output. Please note that in final revisions of this thesis, an error was noted in parameterization of the New Hampshire temperature data that is being corrected for peer-review manuscripts.

Impacts to Reproductive Phenology

Reproductive phenology was significantly affected by differing latitudes and changing temperatures across all four scenarios, particularly the first flowering day and extent (Figure 23), germination day (Figures 22-23), seed bank dormancy, and the time when the seedlings transitioned to adulthood (Table 10 and 11).

In North Carolina, after year 2025, the probability that an individual would flower increased from 50% to 90%. In year 2090 in New Hampshire, plants were flowering 40 days earlier than in 2050. By year 2025 in Virginia an individual's probability of flowering increased from 50% to 90%. By year 2090, flowering was occurring much earlier in the year at all sites, specifically January in North Carolina, April in Virginia, and March in New Hampshire. On average, flowering started 41 days earlier per a 4.5°C increase in temperature in NH, 2 days earlier per a 4.5°C increase in temperature in VA, and 71 days earlier per a 4.5°C increase in temperature in NC.

For seed germination, the largest change at one site was experienced in Virginia between years 2007 and 2050 by occurring 55 days later in the fall with warmer temperatures. It was interesting that most of the differences experienced across the sites, for seed germination in particular, changed across location/latitude then across simulation years within the same site (Figures 22-23). This was also a case where the year 2050 produced greater changes in model results than the year

2090. For example, germination occurred 76 days later in North Carolina in year 2050 than in New Hampshire in 2050 (Table 11). As the seed germination day is pushed further in the year, it decreases the gap between seed germination and the first flowering day. This is a concern since if the seeds are germinating closer to their flowering day they have less time to grow enough biomass in order to obtain the number of nodes needed to transition to adulthood to be eligible to flower. If the individual then misses out on their window of flowering then they would not have seeded the sediment seed bank.

Despite differences in the time of year and latitude, recall that Silberhorn et al. (1983, Figure 9) reported that key reproductive events all appear to occur at the same temperature thresholds. This same phenomenon emerged from the model results. Furthermore, the results displayed earlier flowering, which is an example of an adaptive mechanism that allows for biomass and population size to increase during optimal conditions, supporting fecundity and greater fitness, and supported by the theoretical predictions described by Tooke and Battery (2010) earlier in Figure 1. Essentially, as flowering shifted earlier in the year, there was an increased number of flowers produced and present during the time of peak flowering.

Another significant finding from the reproductive phenology results was that as temperatures increased across the three sites, the seed bank dormancy time was extended. Seed bank is controlled by the completion of flowering (when the seeds enter the sediment seed bank) and germination (when seeds exit the sediment seed bank). From New Hampshire 2007 and 2090, seed bank dormancy increased by 52 days, for Virginia it increased by 29 days, and in North Carolina it increased by 80

days. This is an important result to consider, especially since *Z. marina* is known to have a short-lived seed viability rate of less than one year. The longer a seed remains in the sediment seed bank, the more at risk it is of losing its viability and being unable to germinate and recruit into the next year's population. The phenomenon of seed mortality is accounted for specifically in Jarvis' (2014) sexual reproduction model at a rate of 0.1 d^{-1}. A second concern with an increased seed dormancy period in the sediment is that the seeds are then exposed to predation for a longer period of time. In Jarvis' model, loss of seeds due to predation is captured by a loss rate of 0.33 d^{-1}. However, there has also been recent work suggesting that the amount of seed predation in eelgrass meadows in the Chesapeake Bay region is not as significant and that seed predation, ultimately, has a limited impact on population recruitment (Manley et al. 2015).

Impacts to Biomass

Total areal biomass (gdw per m^2) was negatively affected by increasing temperatures in New Hampshire and Virginia with results in North Carolina showing increased total biomass at certain points in the year followed by steep declines (i.e. drastic peaks and valleys) due to stressful temperatures producing negative growth rates and increasing mortality due to flowering (Figure 27 and in Table 10).

In the New Hampshire (Figure 27) 2090 simulation, total biomass peaked earlier in the year followed by a steep decline in total biomass and a small recovery due to new germinated seedlings entering the population. By day 800 of the simulation, total biomass in New Hampshire 2090 was 5 times smaller than 2007 levels. After further analyzing the NH 2090 results, it was noted that the individuals

should have prematurely died near the end of the run due to sustained negative aboveground biomass values, despite still maintaining positive root biomass. This is discussed further in the model limitations and future work section where I question the way by which the model accounts for mortality due to increased temperatures. With this in mind, however, I was able to capture and interpret other plant characteristics that would suggest a meadow die-off such as a sustained negative above: below ground ratio, loss of all aboveground biomass, and very small individual ramet biomass (g DW ramet^{-1}) values during adulthood. Therefore, after reviewing additional output from the New Hampshire 2090 results, I would conclude that mortality of the entire population occurred by late fall. As stated previously, please note that in final thesis revisions an error in parameterization of the New Hampshire temperature data was found and is being corrected for peer-review manuscripts.

Results from Virginia (Figure 27) were the most extreme in terms of differences between 2007 and 2090 biomass levels. The 2007 Virginia simulation increased rapidly to nearly 316 gDW m^{-2}. There was a small decline just before the peak in biomass because of mortality due to flowering. Virginia's 2007 total biomass results were the largest and most productive across all simulations. In the Virginia 2090 simulation, all the individuals flowered in the spring and therefore died in early June. No biomass was sustained during the hot summer months in 2090. Eelgrass entered back in the model via seed germination in mid-October where individuals continue to remain seedlings until the end of the simulation.

Results from the simulations of North Carolina conditions (Figure 27), were more difficult to interpret. Total average areal biomass was actually higher in the 2090 simulation than in the 2007 simulation. In the 2090 scenario, individuals were able to rapidly produce biomass and lateral shoots as the warmer temperatures helped increase the individuals' ability to photosynthesize rapidly. This was followed by a slow decline due to stressful temperatures, leading to a large meadow die-off as a result of flowering, which occurred very early in the year on January 4.

Looking deeper into the results, the North Carolina 2090 individuals should not have been able to survive to the end of the run due to the very high temperatures reaching 36°C in the summer and fall months. Additionally, the North Carolina 2090 individuals were not able to flower on their flowering day (January 4) since that day occurred so much earlier in the year and the newly germinated seedlings were not eligible to flower during that earlier time window available. Coupled with mortality due to high temperatures and an inability to supply the sediment seed bank via flowering, the future of eelgrass at the North Carolina site in 2090 seems grim and suggests a narrowing of its biogeographic range. Arnold et al. (2017) suggests equally grim outlooks for temperate seagrasses under warming temperatures.

Other significant biomass characteristics, such as peak densities, number of lateral shoots produced, and average biomass ramet^{-1} were also captured (Table 10). The biomass results are within mean reported ranges for observed eelgrass beds (mean 246.2 +- 22.6 gdw m^{-2}) and densities (Clausen et al. 2014). The results were also quite similar to those observed and modeled by Jarvis et al. (2014), which included peak total areal biomass results around 100 gdw m^{-2}.

Impacts on Life History Strategies

These results imply that with changing and more stressful conditions, we may see *Z. marina* favor sexual reproduction strategies, also known as colonizing characteristics by Kilminister (2015; Figure 25). Colonizing seagrass species are those that invest heavily in sexual reproduction, fluctuate more dramatically and rapidly in year-to-year biomass, and tend to recover from disturbances more quickly (Kilminster et al. 2015). The timing of when seedlings transitioned to adulthood/length of juvenile development in the model signifies its overall growing conditions and environment. In North Carolina, all of the seedlings transitioned to adulthood by mid-February while for New Hampshire and Virginia this did not occur until mid-April and March, respectively. In the warming temperatures at lower latitudes individual seedlings transitioned to adulthood in half the time it took individuals in New Hampshire (Table 10). Rapid growth and development, enhanced by warmer temperatures to a threshold point, in the more southern sites was critical in order for the seedlings to develop into adults before their earlier flowering window past.

Plant stress signals were evident within the biomass and reproductive results during the four warming temperature experiments throughout the simulations. At the same time, *Z.marina's* adaptive responses suggests that shifting life history strategies and phenology may allow the species to persist amid some warming, although the stress signals and results as interpreted imply large-scale mortality events by year 2090 and even 2050. Extremely low levels of ramet biomass are likely indicative of a die-off by year 2090 across all sites. These obvious stress signals and characteristics

in the results signifying threshold points of death caused me to revisit how mortality due to temperature in the model is explicitly represented.

3.5 Model Limitations and Future Work

3.5.1 Mortality due to temperature

In performing the model climate simulations, I discovered a few unexpected weaknesses in the model that will benefit from future improvement. One component that needs adjustment is the mortality due to temperature as I questioned several of the biomass results, especially noting eelgrass survival in year 2090 in North Carolina, despite other biomass characteristics in the model indicating die-off. Currently, mortality in the model occurs via two mechanisms: flowering and negative root biomass. An individual remains in the model only for 60 days post flowering after which it dies. Secondly, the model includes a code that signifies death of an individual once it has negative root biomass. This was an update made in the VEMv.2 that the VEMv.1 did not originally incorporate.

During the final climate simulation runs, it was noted that some of the plants' aboveground biomass values were decreasing negatively below 0 and other sites were experiencing very low adult ramet biomass values. Mortality due to sustained negative aboveground biomass values could also be added in future iterations of the VEMv.2 Additionally, the model does not include a mortality force multiplier for situations when there are consecutive days of stressful temperatures. In a real world setting, we know that many times plant mortality is due to a concentrated bout of stressful temperatures over a short yet sustained period of time. While individuals are

typically resilient and able to cope and bounce back from one or two very stressful days if prolonged for a week or more the individual is likely no longer able to survive. Lastly, as the VEMv.1 was not previously used to simulate conditions in very high temperatures and simulations in high temperatures could not be tested during VEMv.2 calibration and validation due to a lack of available data, there remains a need to test and measure these conditions in future mesocom studies. Overall, I recognize that the model would benefit from improvement in the temperature-mortality relationship, however, I was able to intelligibly interpret the results given many of the other biomass results and meadow indicators of mortality and eelgrass bed health.

3.5.2 Additional Future Work

More broadly, since the model simulates growth in space, hydrodynamic components of seed dispersal and densities should be more explicitly addressed. Future work in the study of the species should focus on documenting additional measurements of growth during the seedling life stage in order to better parameterize, calibrate, and validate the seedling specific growth rate formulation and biomass parameters. Furthermore, more data on seedling establishment is needed as low initial seedling establishment rates still remain the most significant bottleneck to success in seed-based restoration projects (Orth et al. 2007). If we understood the process of seedling establishment better then it could be more formally tested by the VEMv.2, other models, and in controlled mesocosm settings.

Similarly, there is still limited understanding on how other environmental conditions, besides temperature, influence eelgrass' investment in sexual reproduction

and how those could be incorporated into the VEMv.2. For example, recent findings by Johnson et al. (2017) suggest that there is a significant relationship between sediment nutrients and light availability on *Z. marina's* investment in sexual reproduction, showing positive relationships with respect to pore water ammonium concentrations and increased PAR availability. These nutrient relationships, however, are still not well understood in eelgrass ecophysiology nor are they currently represented in the model. Furthermore, according to Zimmerman et al. (2015) one aspect associated with climate change, rising carbon dioxide levels (CO_2), may actually benefit eelgrass. In predictions of their bio-optical model *Grasslight* in the Chesapeake Bay region, Zimmerman et al. (2015) suggests that increasing CO_2 should stimulate photosynthesis sufficiently enough to offset the negative effects of thermal stress on eelgrass growth.

Changing morphology due to genetic variation and varying genotypes is another aspect that should be addressed to improve current models and understanding of seagrass genotypic versus phenotypic diversity (Marsden 2015). As this simulation study is focused only on the effects that warmer waters due to climate change will bring, other aspects such increased atmospheric CO_2, changes in light availability/cloud cover, and fluctuations in nutrient cycles should be assessed going forward and more formally integrated into the model. These changes could have an interactive effect and either contribute to or provide benefits in the face of temperature stress.

Similarly, as evidenced by the increased colonizing characteristics *Z. marina* model output displayed under warming temperature simulations, there is also the

potential for greater species competition with colonizing species such as the abundant *Ruppia maritima* found along the East Coast. A species composition shift fitting this description was recently documented in shallow lagoons along the Florida Gulf Coast by Christiaen (2016), where the less colonizing seagrass species (*Halodule wrightii*) was replaced by the more colonizing species *R. maritima.* In the end, however, the study showed that regardless of which seagrass species was dominant in the small lagoon systems, there was little impact on the ecosystem services that the bed provided on the whole (Christiaen 2016).

Irrespective of the specific model simulation results, the VEMv.2 fills a gap in seagrass literature by presenting a new seedling specific growth rate formulation and parameterization to track the growth rates and allocation of biomass of eelgrass seedlings within their first few months of growth before adulthood and sexual reproduction.

4. CONCLUSION

As the climate simulation results demonstrated, eelgrass response to increased temperatures will likely lead to a decline in biomass and a shift in the species' reproductive phenology and overall life history strategy as an adaptive mechanism for survival in the face of climate change. We documented changes in specific phenological events in the model, such as earlier first flowering days and later germination times due to warmer water temperatures. With future climate change, these shifts could result in higher fecundity since adjusting the flowering day would afford the individual more time during optimal water temperatures to develop their reproductive organs and set seed. If the individuals cannot shift the timing of these

key events then their window of optimal growth and fecundity would reduce. Additionally, as displayed in the model results, earlier flowering days combined with later germination times cause longer seed dormancy periods in the sediment seed bank. The longer the seeds spend in the seed bank, the more vulnerable they are to predation in addition to losing their viability by rotting as eelgrass seeds typically do not survive past 10 months after release (Harwell and Orth 2002, Harrison 1991; Moore et al. 1993; but see Churchill 1983). Changes in *Z. marina*'s biomass and reproductive phenology could also have a lasting effect on ecosystem services. Changes in meadow cover and dynamics could lead to changes in its ability to uptake nutrients, store carbon, and provide habitat and food for other species, including those that are of economic and societal importance (Cole and Moksnes 2016).

Knowing the limitations and "naïve realism" that IBMs (Grimm and Railsback 2005) such as the VEMv.2 are built upon, these results should be treated like any ecological forecast, prediction tool, or well-informed hypothesis, especially when informing species management. What is clear, however, is that additional model updates, studies, data collection, and monitoring of eelgrass sexual reproduction and the seedling life stage are necessary. The seagrass community should also pay close attention to shifts in species composition towards more colonizing species with increasing disturbances and projected climate change.

As our virtual experiments demonstrated, it is also important to fully understand the reproductive phenology of the species in order to anticipate optimal seeding or planting days for restoration. A coastal manager would lose a significant amount of time and resources if seeds were mistakenly broadcast after the season's

first flowering date given that the species has a less than one-year viable sediment seed bank. Coupled with the recent increased attention and interest given to the use of "natural infrastructure" along our coasts, we want to make sure that any engineering, restoration, or management guidelines for these natural systems are both informed by our best understanding of seagrass ecology and the state of future habitat conditions. This type of information is critical to the decisions that coastal managers are making now.

At the highest level, this thesis fills a gap in seagrass modeling literature and underscores the importance of *Z. marina's* phenotypic plasticity and flexible reproductive strategies allowing for eelgrass populations to persist through the end of the century amid projected rates of climate change. According to the model simulations, *Z. marina's* species plasticity in the short term should allow enough time for evolutionary adaptation to keep pace with the projected warming temperatures in the future. This acclimation time could allow for genetic changes to take place rather than an immediate shift in *Z. marina's* biogeographical range or mass extinction. There has also been evidence in the scientific literature of evolutionary adaptation to climate change over relatively short time spans in some species (e.g., five to 30 years); although, this is a very difficult trait to quantify and requires further research (Bradshaw and Holzapfel 2006, Williams et al 2008). These conclusions, albeit informative, should be taken cautiously as the effects of climate change on seagrass ecosystems extend far beyond the scope of any one model and assessing only temperature changes alone.

TABLES

Study	Location
Short 1980	Charlestown Pond, USA
Short 1981	Lzembek Lagoon, USA
Verhagen & Nienhuis 1983	Lake Grevelingen, Netherlands
Wetzel & Neckles 1986	Chesapeake Bay, USA
Zimmerman et al. 1987	N/A
Bach 1993	Kattegat Estuary, Denmark
Bocci 1997	Lagoon of Venice, Italy
Zharova et al. 2001	Lagoon of Venice, Italy
Best et al. 2001 (Buzzelli 1996)	Chesapeake Bay, USA
Cerco & Moore 2001	Chesapeake Bay, USA
Neckles et al. 2005	Maquoit Bay, USA
Harris 2006 (VEMv.1)	Rhode Island, USA
Carr et al. 2012	Chesapeake Bay, USA
Jarvis et al. 2014	Chesapeake Bay, USA

Table 1. List of key eelgrass simulation models 1980 to 2017

	Elements of the original ODD protocol (Grimm et al., 2006)	Elements of the updated ODD protocol
Overview	1. Purpose 2. State variables and scales 3. Process overview and scheduling	1. Purpose 2. Entities, state variables, and scales 3. Process overview and scheduling
Design concepts	4. Design concepts • Emergence • Adaptation • Fitness • Prediction • Sensing • Interaction • Stochasticity • Collectives • Observation	4. Design concepts • Basic principles • Emergence • Adaptation • Objectives • Learning • Prediction • Sensing • Interaction • Stochasticity • Collectives • Observation
Details	5. Initilization 6. Input 7. Submodels	5. Initialization 6. Input data 7. Submodels

Table 2. The seven elements of the original and updated Overview, Design concepts, Details (ODD) protocol. Numbering the seven elements when using the protocol in publication is optional. The elements can also be grouped broadly into three categories but these categories are not meant to be formally included when using the ODD protocol (Grimm et al. 200

Study	Seedling Definition
Ochieng et al. 2010	Three month old plants, age identified by the presence of seed coat still attached to cotyledon and by # of leaves. Allowed for 103 days of growth after 22 day transplant recovery in mesocosm.
Jarvis & Moore 2012	[<1 year of growth] Seedlings reached maximum biomass #'s within 4 months.
Niu et al. 2012	Collected and germinated seeds. The germinated seeds were cultured for two days in 20 degrees C until the cotyledon grew up right out of the seed coat up to 0.5 cm. Study used 800 seedlings with cotyledon length 0.78 +- 0.20 cm. Experiment lasted only 30 days.
Rasmussen at al. 2012	Seedlings identified as small shoots with 2-5 narrow 1-2 mm leaves and no rhizome present. Many of the collected seedlings still had the seed coat attached to the hypocotyle. Collected seedlings 7.9 +- 2.3 cm in height.
Abe et al. 2008	Mature seeds buried in sand laid at bottom of culture bottles. After 2 months culture, many seedlings appeared and grew to a length of 10 cm under 15 degrees C and 50 umol photons/m^-2 per s. Seedlings then held at different light treatments for 6 days. Sample seedlings in study ranged from 10.1 to 16.4 cm in total length.
Jarvis et al. 2014	In model, once germinated, seedlings were then converted back to above and below ground carbon values using fixed conversion factors. When above ground vegetative Z. marina biomass was <0.44g C m^-2 all above and below ground seedling biomass was transferred to the vegetative shoot and root stocks. If more than this, then seedling mortality 100%. Relationship based on inhibitory effect of shading.
Tanner & Parham 2010	Seeds were considered germinated when the seed coat split and the hypocotyl and cotyledon exposed. Seedlings were recognized by the appearance of photosynthetic leaves and adventitious roots (Fig 1E). Seeds vernalized for 1-4 weeks before planting. Time to germination: 3- 29 days (fastest germination in low salinity treatments). Development into seedlings with green leaves took 16 to 56 days. Most seedlings emerged in 50 days. 1,000 seeds planted, 90 days after planting there were 26,000 shoots, of which approximately 15,000 met specifications to harvest and transplant (shoots > 12.25cm tall, rhizome > 2.5 cm long. Fertilization enhanced growth rates of seedlings, proliferation of lateral shoots, and shortened period of time necessary to reach planting size.
J. Bintz dissertation 2002	Seeds planted late December, allowed seeds to germinate and seedlings began to emerge within the month. Within 4 months seedlings reached a mean height of 4 cm and were transplanted to outdoor tanks. PAR Experiment lasted for 12 weeks May –August.

Table 3. A literature review on the description of the term "eelgrass seedling" confirmed multiple inconsistencies in how the seedling life stage was denoted.

Select Model Parameters	Units	Values	Sources
Water Depth	m	0.8	Jarvis et al. 2014
Seedling Maximum Specific Growth Rate	gdw gdw^{-1} day^{-1}	0.03	Rasmussen 2012; Bintz & Nixon 2001
Adult Maximum Specific Growth Rate	gdw gdw^{-1} day^{-1}	0.06	Harris 2006
Nitrogen Loading	g N m^{-2} day^{-1}	0.00854	Harris 2006
Sediment Sulfide Concentration	μMol/L	50	Harris 2006

Table 4. Parameter estimates for VEMv.2

Equation Number	Equations for Seedling Growth	Description
(1)	$\mu = \mu_{max} * f(I) * f(T) * f(S)$	General growth rate formulation
(1a)	$\mu Adult = \left(\left(\frac{\mu_{max}\alpha I}{\mu_{max} + \alpha I}\right) + ro\right) * f(S)$	Adult growth rate formulation
(2)	$Adult\ S\ Limitation = If\ S \leq 55.45\ then\ 1\ elseif\ S \geq 2000\ then\ 0\ else\ 13.6 * S^{(-0.65)}$	Adult sulfide limitation
(3)	$\mu S(I, T) = \mu_{max} * f(I,T) * f(S)$	Seedling growth rate formulation
(4)	$I_z = I_o * e^{-kz}$	Beer-Lambert's Law
(5)	$P(I) = P_{max\,g} * \tanh\left(\frac{I}{I_k}\right) + R$	Jassby & Platt PI curve applied
(6)	$f(I,T) = If\ T \leq 25\ then\ \frac{\left(0.97T - 0.75) * \tanh(\frac{I_z}{4.59T + 3.34}) + (-0.949T - 1.0503\right)}{20.02}\ else\ \frac{((-2.8T + 94.3) * \tanh(\frac{I_z}{4.5978T + 3.347}) + (-0.949T - 1.0503))}{20.02}$	Seedling interactive light-temperature limitation function (adapted from Jassby & Platt)
(7)	$If\ S \leq 1\ then\ 1\ elseif\ S \geq 2000\ then\ 0\ else\ 1.0239 * e^{(-0.002*S)}$	Seedling sulfide limitation

Table 5. List of governing equations for eelgrass growth in VEMv.2 including the two separate specific growth rate formulations for adults and seedlings.

	New Hampshire	Virginia	North Carolina
Seeds Initialized in Seed Bank	100	100	100
Seedlings Initialized	0	0	0
Adults Initialized	0	0	0
Area simulated (m^{-2})	1.68	1.68	1.68
Initial Density (ramets m^{-2})	24	24	24
Day of Year Initialized	1-Jan	1-Jan	1-Jan
Temperature Forcing Function (°C)	Great Bay, NH NERR	South Bay, VA	Research Creek, NC NERR
Light Forcing Function (source, µMol m^{-2} s^{-1})	Great Bay, NH NERR, Surface PAR	Taskina's Creek, VA, Surface PAR	Research Creek, NC NERR, Surface PAR
K(d) Attenuation Coefficient (m-2)	LEM output from Hog Island Bay, VA	LEM output from Hog Island Bay, VA	LEM output from Hog Island Bay, VA
Water Depth (m)	0.8	0.8	0.8
Nutrient Loading (g N m^{-2})	0.00854	0.00854	0.00854
Sediment Sulfide (µMol/L)	50	50	50
Initial Canopy Height (cm)	Seedlings: 4 Adults: 12	Seedlings: 4 Adults: 12	Seedlings: 4 Adults: 12

Table 6. Forcing functions and initial conditions used in VEMv.2 temperature warming simulations.

Development Stage	Temperature (°C)
Spadix primordia (immature flowers)	0.5 - 3
Anthesis (flowering & pollination)	15
Mature fruit	20 - 21
Peak seed production	20 - 23
Seeds in sediment seed bank	0-30
Seed dormancy - no germination	>20
Seed germination	~15
Peak germination and presence of seedlings	5 - 15

Table 7. Relationships between *Z. marina* reproductive phenology and temperature (Moore & Orth 1982, Setchell 1929, Moore et al. 2003).

Seedling Sample	Aboveground biomass	Belowground biomass	Total biomass	Above: belowground ratio
1	0.00652	0.00093	0.00745	7.0
2	0.00121	0.00040	0.00161	3.0
3	0.00168	0.00035	0.00203	4.8
4	0.00084	0.00015	0.00099	5.6
5	0.0014	0.00017	0.00157	8.2
6	0.00319	0.00035	0.00354	9.1
7	0.00139	0.00025	0.00164	5.6
8	0.00355	0.00049	0.00404	7.2
9	0.01222	0.00239	0.01461	5.1
10	0.00123	0.00006	0.00129	20.5
11	0.00233	0.00034	0.00267	6.9
12	0.00061	0.00023	0.00084	2.7
13	0.00111	0.00017	0.00128	6.5
14	0.00118	0.00007	0.00125	16.9
15	0.00256	0.00030	0.00286	8.5
16	0.00119	0.00016	0.00135	7.4
17	0.00094	0.00004	0.00098	23.5
18	0.00201	0.00012	0.00213	16.7
19	0.00142	0.00037	0.00179	3.8
20	0.00151	0.00056	0.00207	2.7
21	0.00127	0.00030	0.00157	4.2
22	0.00126	0.00005	0.00131	25.2
23	0.0011	0.00026	0.00136	4.2
24	0.0011	0.00005	0.00115	22.0
25	0.00166	0.00030	0.00196	5.5
26	0.00131	0.00014	0.00145	9.4
27	0.00112	0.00005	0.00117	22.4
28	0.00106	0.00019	0.00125	5.6
29	0.00086	0.00015	0.00101	5.7
30	0.00099	0.00009	0.00108	11.0
AVERAGE	**0.001994**	**0.000316**	**0.00231**	**9.6**

Table 8. Brush and Orth (2015) unpublished data on seedling biomass (grams DW).

μmax (gDW gDW^{-1} d^{-1})	% change in μmax	Biomass (gDW shoot-1)	% change in total biomass (gDW shoot	AG Biomass (gDW shoot-1)	% change in AG biomass (gDW shoot	BG Biomass (gDW shoot-1)	% change in BG biomass (gDW shoot	Day of year transition to adulthood	% change transition to adulthood
0.03	**0%**	0.00690	0%	0.00539	0%	0.00151	0%	147	0%
0.0315	**5%**	0.00697	1%	0.00543	1%	0.00154	2%	146	-1%
0.033	**10%**	0.00704	2%	0.00548	2%	0.00157	4%	142	-3%
0.036	**20%**	0.00718	4%	0.00555	3%	0.00163	8%	137	-7%
0.0285	**-5%**	0.00684	-1%	0.00536	-1%	0.00148	-2%	149	1%
0.027	**-10%**	0.00678	-2%	0.00531	-1%	0.00146	-3%	158	7%
0.024	**-20%**	0.00665	-4%	0.00525	-3%	0.00141	-7%	161	10%

Table 9. Results from model sensitivity analysis.

Biomass Climate Simulations	NH2007	NH2090	VA2007	VA2090	NC2007	NC2090
Peak biomass during simulation (gdw m^{-2})	73	76	316	37	39	94
Date of peak biomass (day-month)	13-Oct	23-Aug	3-Sep	9-Jun	12-May	16-Jun
Average meadow biomass during simulation (gdw/m^{-2})	24.7	15.0	91.6	3.4	5.3	44.4
Average biomass per ramet (gdw $ramet^{-1}$)	0.096	0.048	0.065	0.029	0.047	0.074
Average number of ramets	258	310	1400	117	113	601
Peak density (ramets/m-2)	465	358	1854	83	86	523
Median density during simulation (ramets/m-2)	71	83	432	83	71	523
Average aboveground to belowground biomass ratio	1.80	1.70	1.80	3.00	2.80	1.10
Total number of flowering ramets during simulation	14	0	17	35	36	108
Total number of lateral shoots produced	724	562	1531	40	80	838
% of ramets that flowered	1.93	0.00	1.11	87.50	45.00	12.89
Date seedlings transitioned to adulthood	18-Apr	11-Apr	13-Feb	17-Mar	19-Feb	15-Feb
Days it took from germination to transition to adulthood	225	207	136	141	111	98

Table 10. Biomass and resource allocation results from seeding experiment.

Reproductive Phenology	NH2007	NH2025	NH2050	NH2090	VA2007	VA2025	VA2050	VA2090	NC2007	NC2025	NC2050	NC2090
Probability of flowering (%)	30	30	50	50	50	90	90	90	90	90	90	1
First flowering day (FFD)	115	115	114	74	103	103	102	101	75	72	5	4
Date of first flowering	24-Apr	24-Apr	23-Apr	14-Mar	12-Apr	12-Apr	11-Apr	10-Apr	15-Mar	12-Mar	5-Jan	4-Jan
Day of seed germination	249	249	262	260	274	295	329	301	305	310	338	314
Date of seed germination	5-Sep	5-Sep	18-Sep	16-Sep	30-Sep	21-Oct	24-Nov	27-Oct	31-Oct	5-Nov	3-Dec	9-Nov
Day of year seedlings transition to adulthood	159	159	158	154	147	141	131	116	109	100	100	89
Date seedlings transition to adulthood (Jan 1 germination)	8-Jun	8-Jun	7-Jun	3-Jun	27-May	21-May	11-May	26-Apr	19-Apr	10-Apr	10-Apr	30-Mar
Seed bank dormancy (days)	134	134	148	186	171	192	227	200	230	238	333	310
Days between seed germination and flowering	231	231	217	179	194	173	138	165	135	127	32	55

Table 11. Results of reproductive phenology from seeding experiment.

	20th Century Temperature Anomaly	Mean Temperatures (°C) above 20th Century Anomaly		
		2025 Anomaly	2050 Anomaly	2090 Anomaly
Annual Average	0	1.04	2.17	4.66
Jan	0	1.18	2.01	4.18
Feb	0	0.71	1.84	3.87
Mar	0	0.92	1.99	4.12
Apr	0	0.95	2.06	4.42
May	0	0.87	2.03	4.45
Jun	0	0.98	2.04	4.66
Jul	0	1.04	2.30	5.16
Aug	0	1.16	2.40	5.46
Sep	0	1.21	2.43	5.24
Oct	0	1.19	2.32	5.25
Nov	0	1.23	2.29	4.73
Dec	0	0.99	2.28	4.38
Dec/Jan/Feb	0	0.96	2.04	4.14
Mar/Apr/May	0	0.91	2.03	4.33
Jun/Jul/Aug	0	1.06	2.24	5.09
Sep/Oct/Nov	0	1.21	2.35	5.07

Table 12. Model temperature anomalies for years 2025, 2050, and 2090 developed using A2 scenarios and averaged over the spatial area of the Chesapeake Bay watershed.

FIGURES

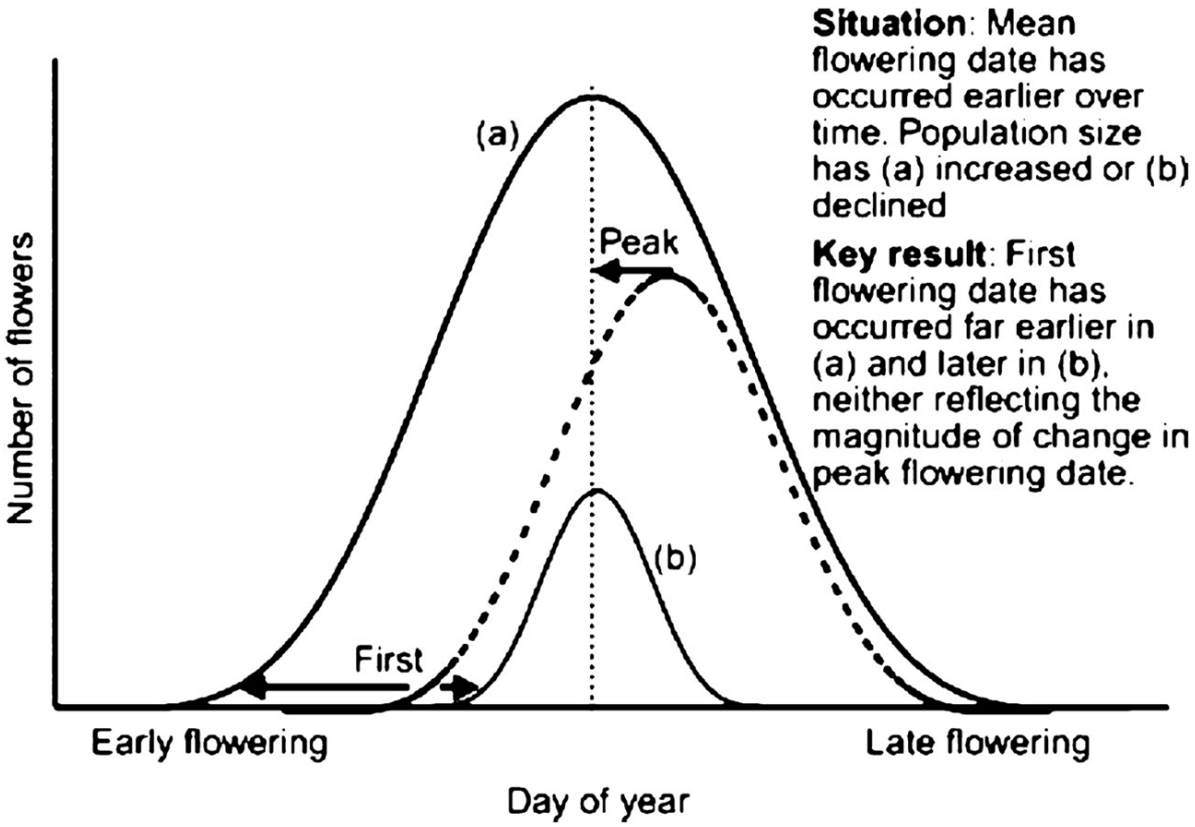

Figure 1.The theoretical effect of changes in population size on changes in first flowering date (reproduced by Tooke and Battey 2010; adapted from Miller-Rushing et al. 2008).

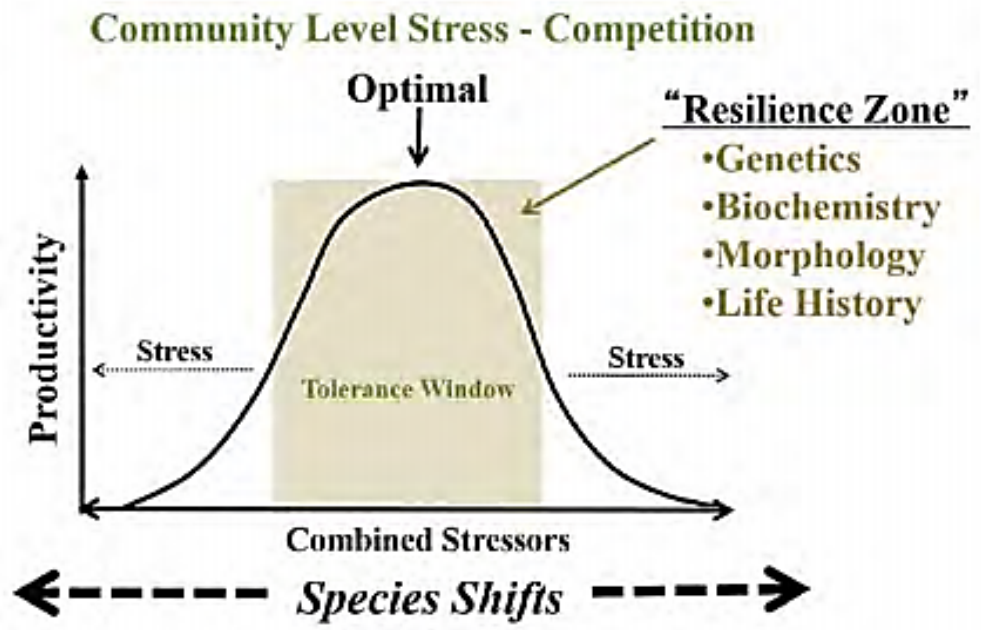

Figure 2. Seagrass species will shift based on their resilience defined by genetics, biochemistry, morphology, life history, and community competition. (Koch 2016).

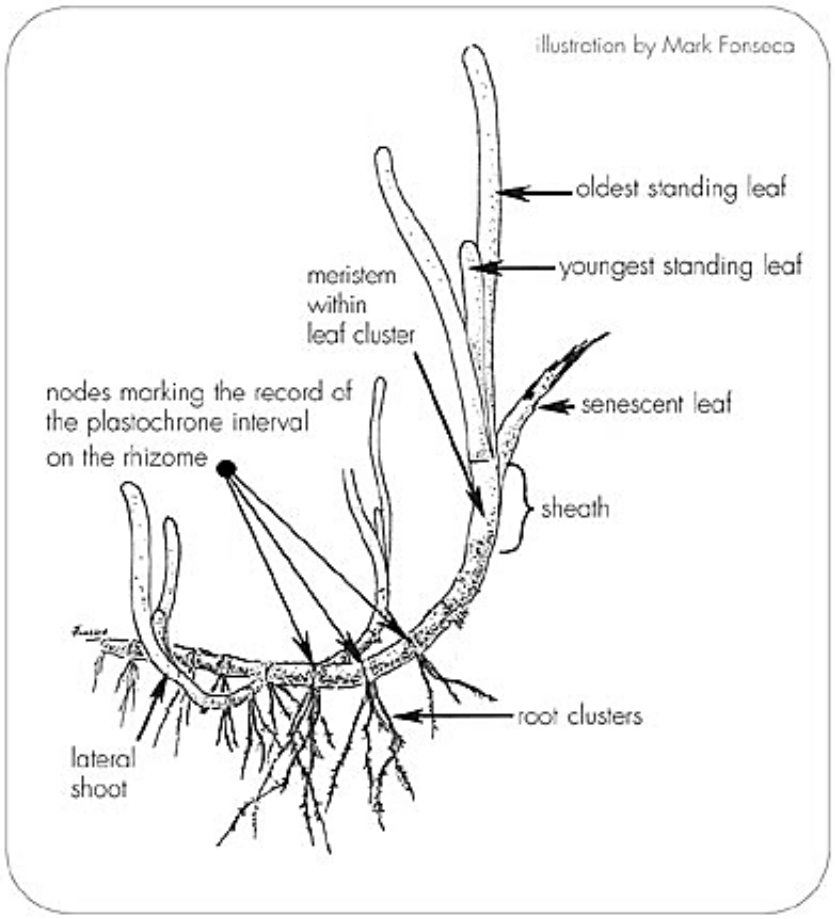

Figure 3. Eelgrass morphology, illustration by Mark Fonseca.

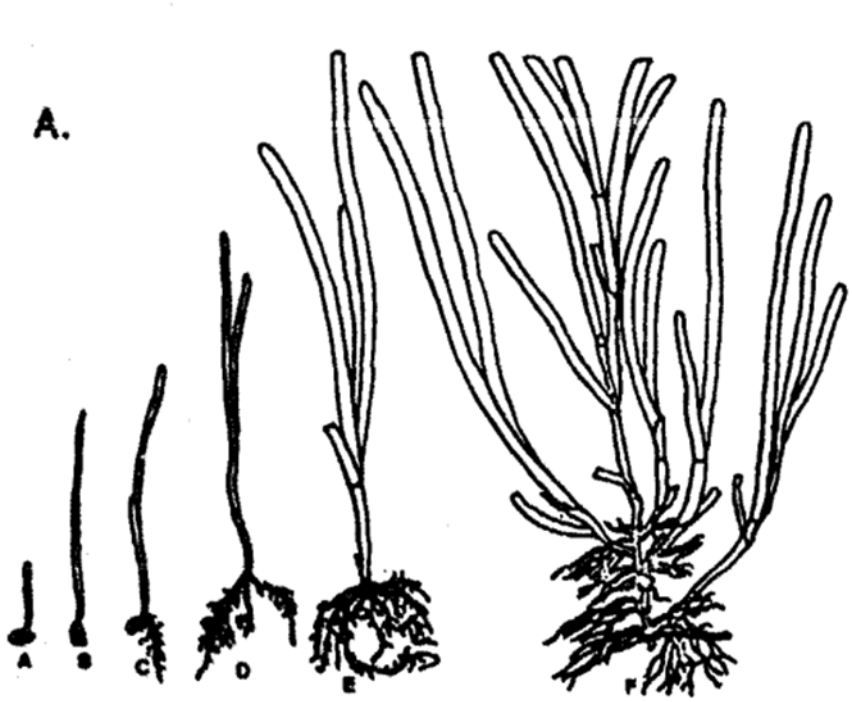

Figure 4. Illustration of eelgrass ontogeny from germinated seed (A, B) to seedling (C, D, E) to adult (F) (Taylor et al. 1954).

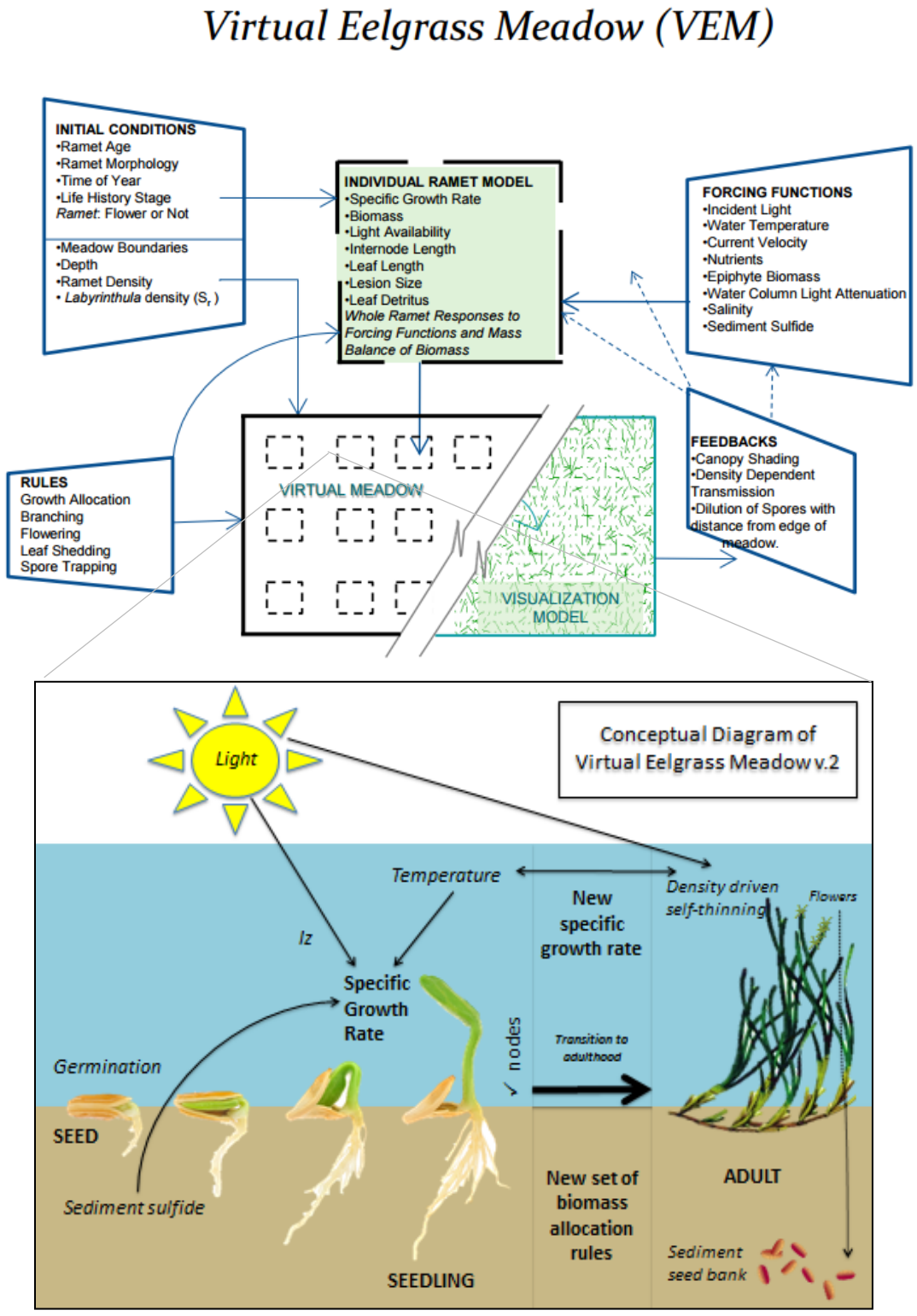

Figure 5. Conceptual diagram of the Virtual Eelgrass Meadow version 2 (VEMv.2) credit: IAN UMCES and Harris (2006)

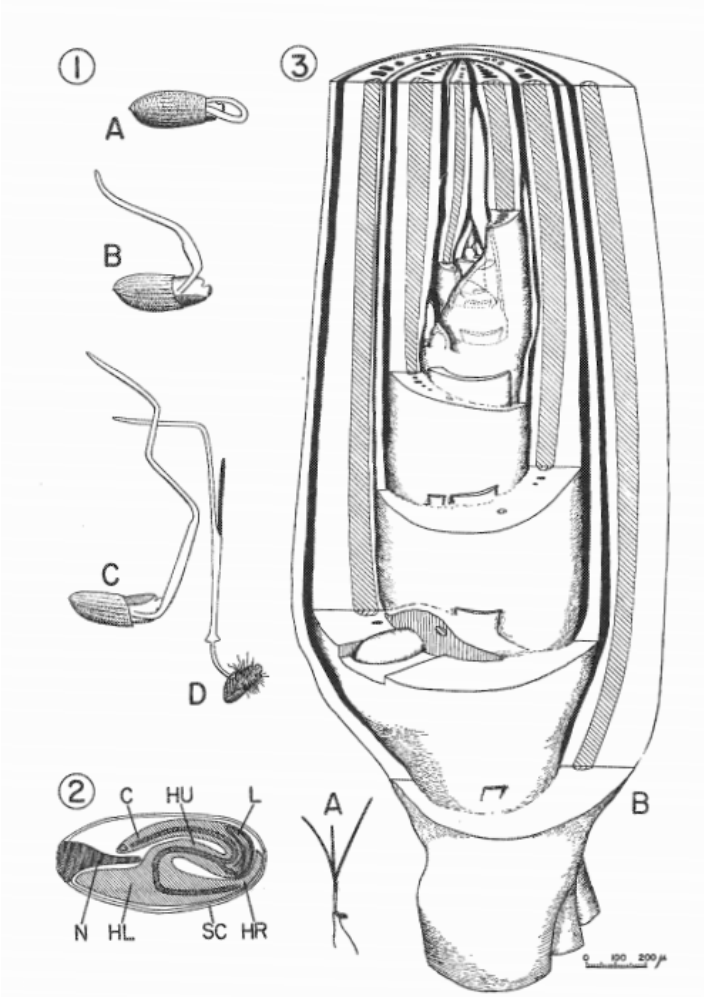

FIG. 1. A, B, and C: sketches of the same seed at 2-day intervals during germination in the laboratory, January, 1953 (× 3.3). D, sketch of a seedling with first leaf extended through the cotyledonary sheath, the first adventitious roots protruding at the cotyledonary node, and hairs developed from the convex surface of the swollen lower hypocotyl (× 2).

FIG. 2. Diagram of a median longitudinal section through a nearly mature seed; procambium is stippled (× 10). *C*, cotyledon; *HL*, lower part of hypocotyl; *HR*, 'radicle' end of hypocotyl; *HU*, upper part of hypocotyl; *L*, leaf; *N*, nucellus; *SC*, seed coat.

FIG. 3. A, sketch of the seedling sectioned to make the reconstruction in B (× 0.35). B, a reconstruction of part of this seedling, from the top of the hypocotyl to beyond the tip of the sixth leaf primordium (×55). Obscuring parts are shown cut away to reveal enclosed structures; broken lines show some structures obscured by overlying parts; diagonal hatching indicates procambium or vascular tissues; solid black shows air spaces, and spaces between the leaves. One root is shown as if exposed at the node of the first leaf (second node); only the bases of the intravaginal squamules are shown. The first branch axis is evident in the axil of the third true leaf.

Figure 6. In-depth botanical illustration and description of the development of a germinated eelgrass seed into a seedling (Taylor 1957).

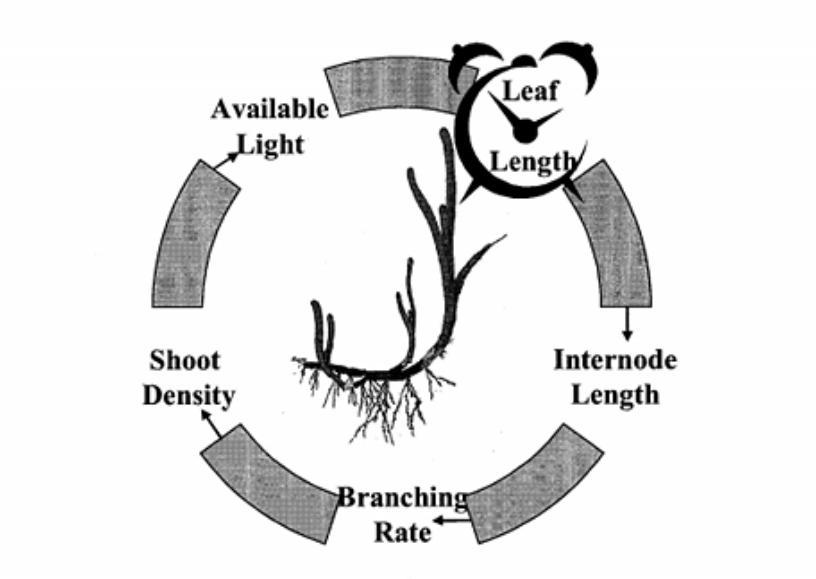

Figure 7. Visualization of the feedback loop between the leaf length “clock” and the dependency of available light on local shoot density in adult plants (Harris 2006).

Rules for VEMv.2 Sexual Reproduction
↓
Count how many days there are in the year above the 20°C average thermal stress threshold.
↓
Use those days to set the individuals' probability of flowering that year (30%, 50%, 90% chance).
↓
When daily average temperature increases to 15°C after winter, flowering commences.
↓
During flowering, multiply the # of flowering shoots by 10 seeds per shoot.
↓
The seeds produced enter the sediment seed bank.
↓
Any reproductive shoots die after 60 days post flowering.
↓
Seeds remain dormant in the seed bank during high summer temperatures.
↓
Only 40% of the seeds in the seed bank remain viable at time of germination.
↓
When temperature decreases below 20°C, viable seeds germinate.
↓
X # of germinated seeds are recruited as new seedlings per m^2 in the model.

Figure 8. VEMv.2 sexual reproduction rules modified by Jarvis et al. 2014.

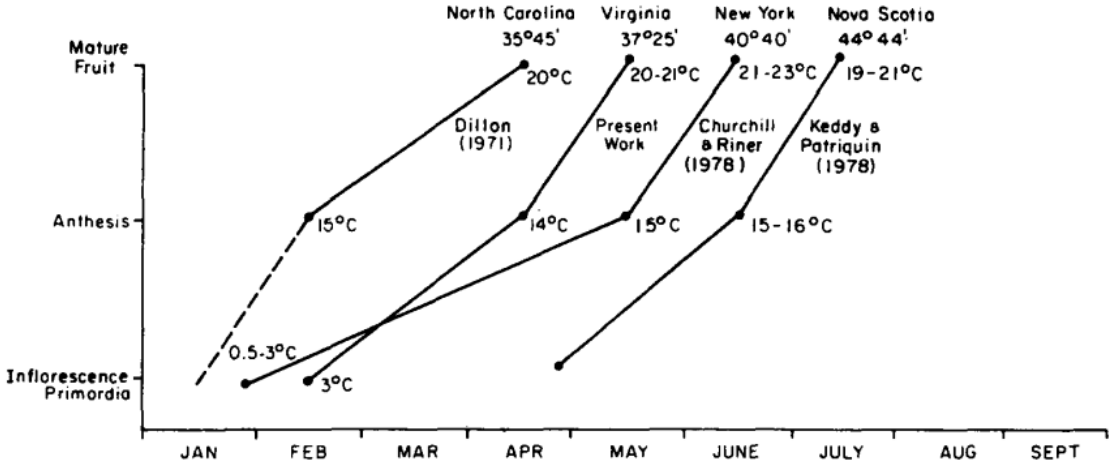

Figure 9. Reproductive phenology of *Z. marina* at different locations (with latitude) along the East coast of the United States (Silberhorn 1983).

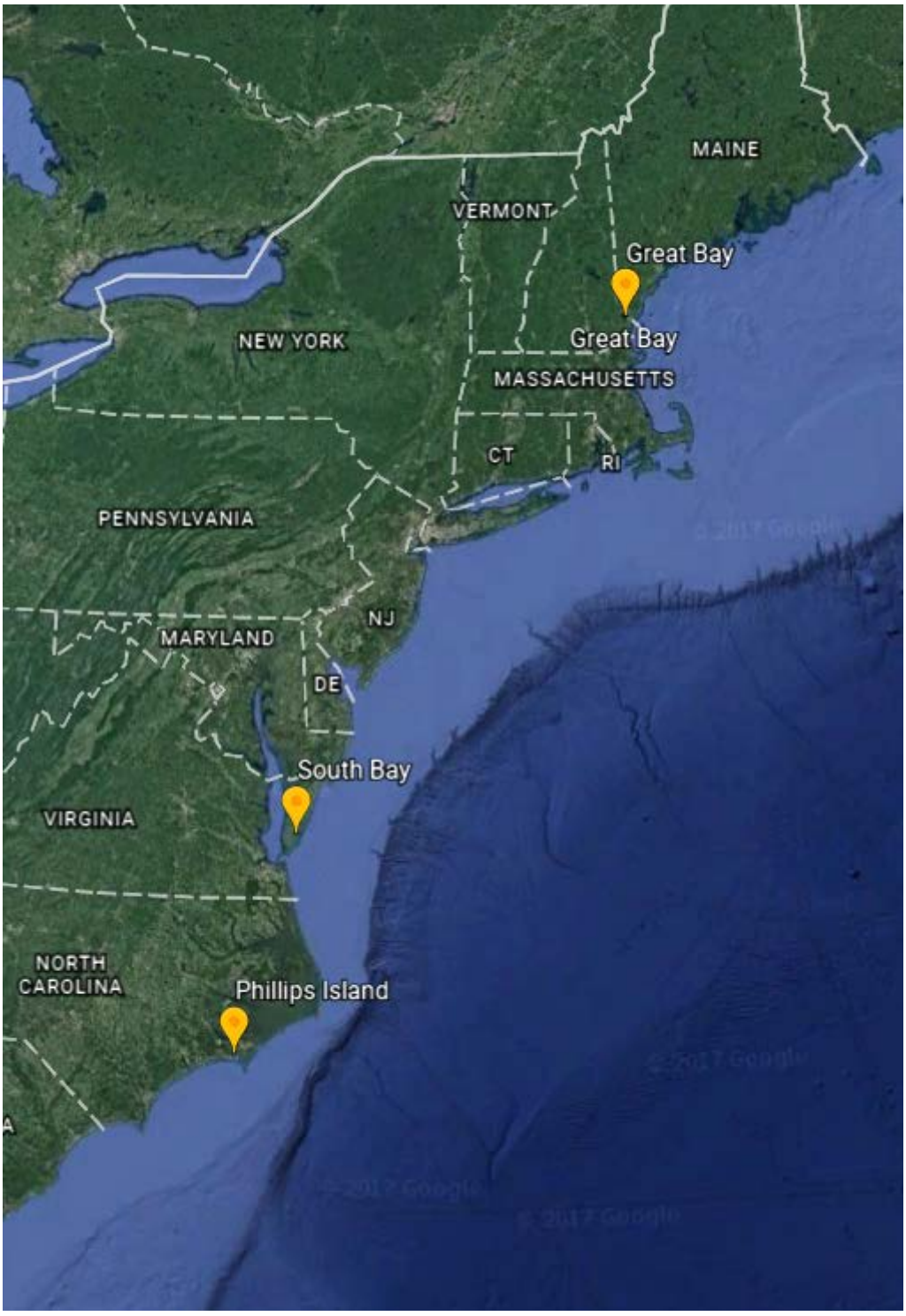

Figure 10. Model simulation locations along a latitudinal gradient of the East coast of the U.S. Featuring: Great Bay, New Hampshire (NH), South Bay, Virginia (VA), and Masonboro, North Carolina (NC).

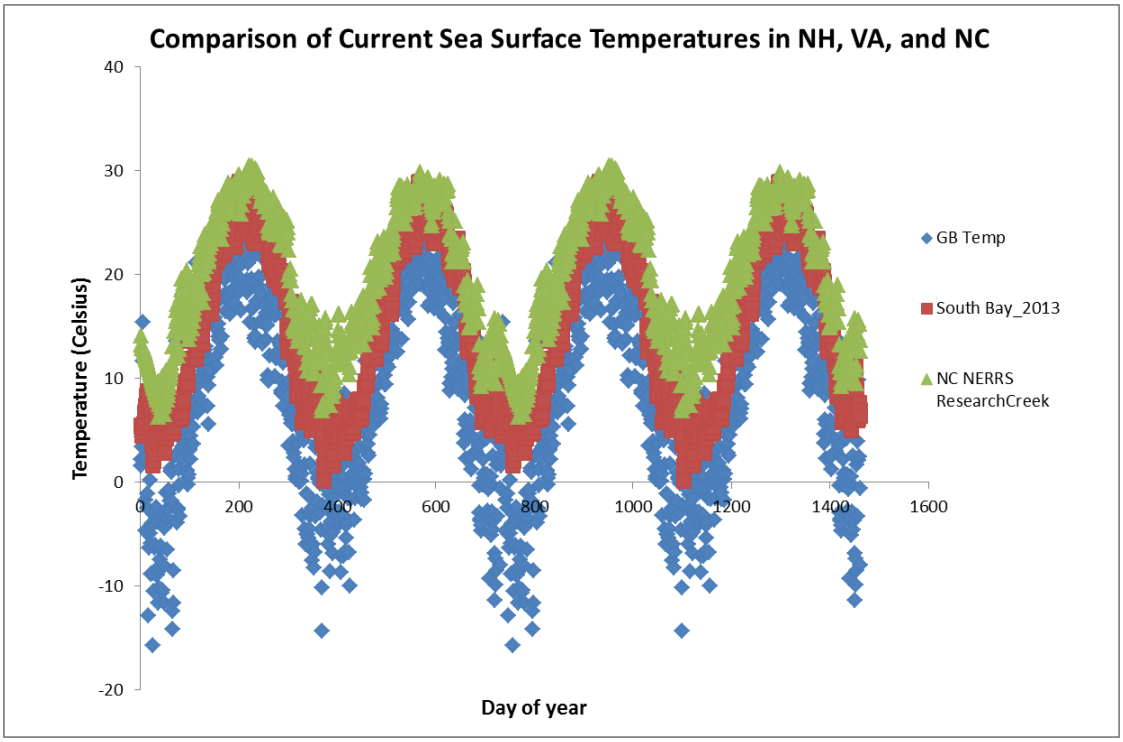

Figure 11. Temperature forcing functions used in the model climate scenarios.

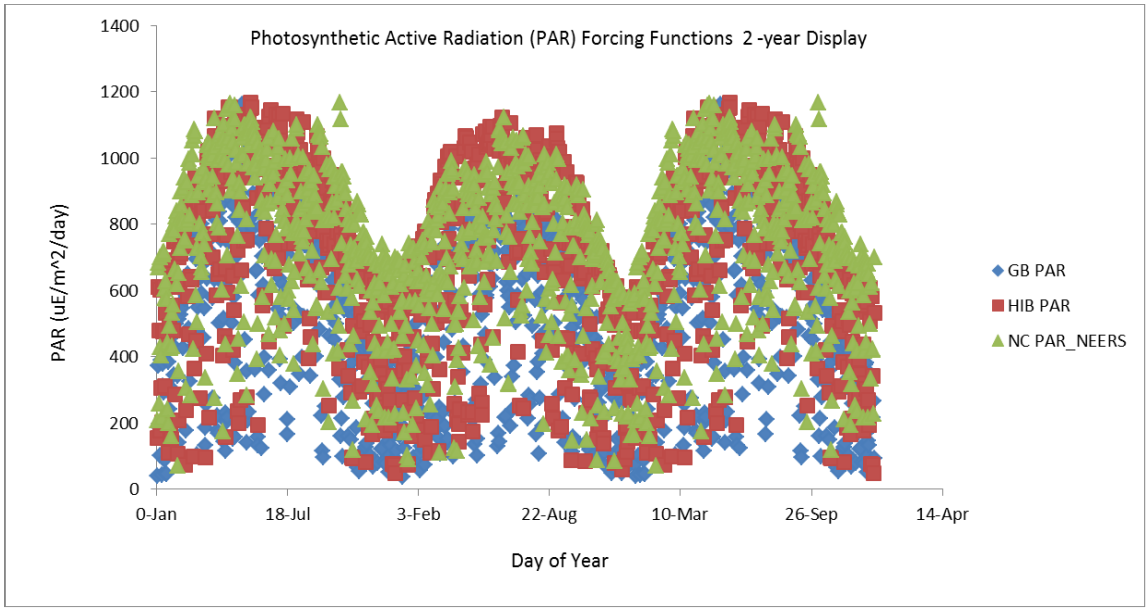

Figure 12. Surface photosynthetically active radiation (PAR) forcing functions used in the model climate scenarios.

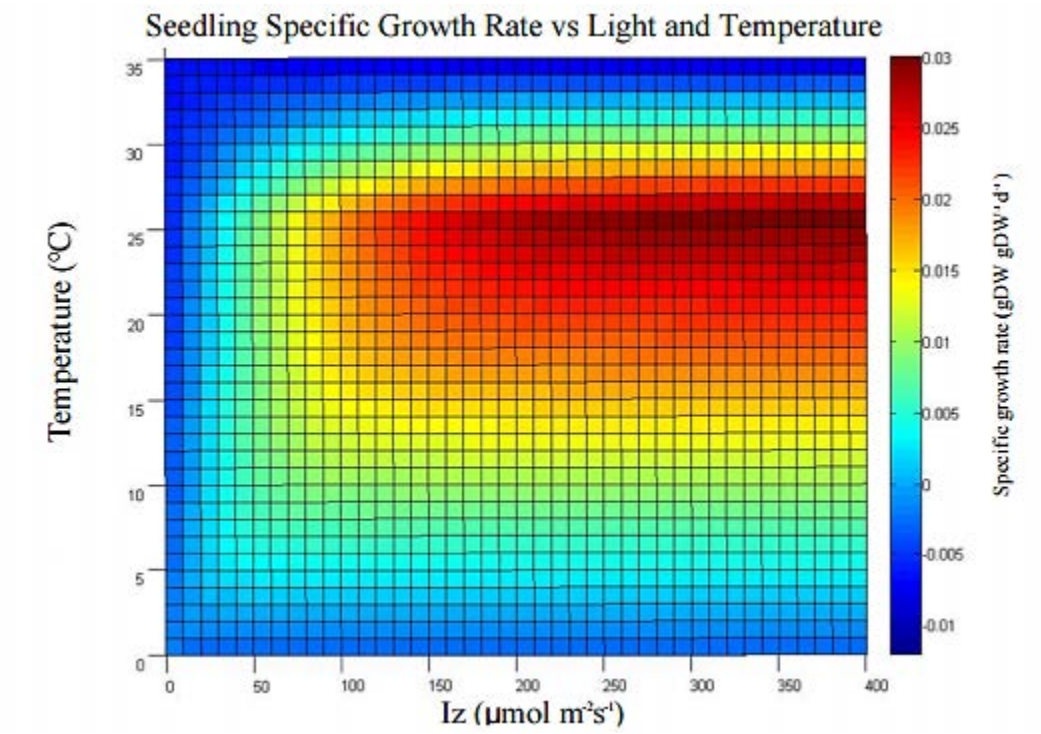

Figure 13. Seedling specific growth rate surface plot showing the optimal temperature zone for growth at 25°C and when light increases above saturating conditions.

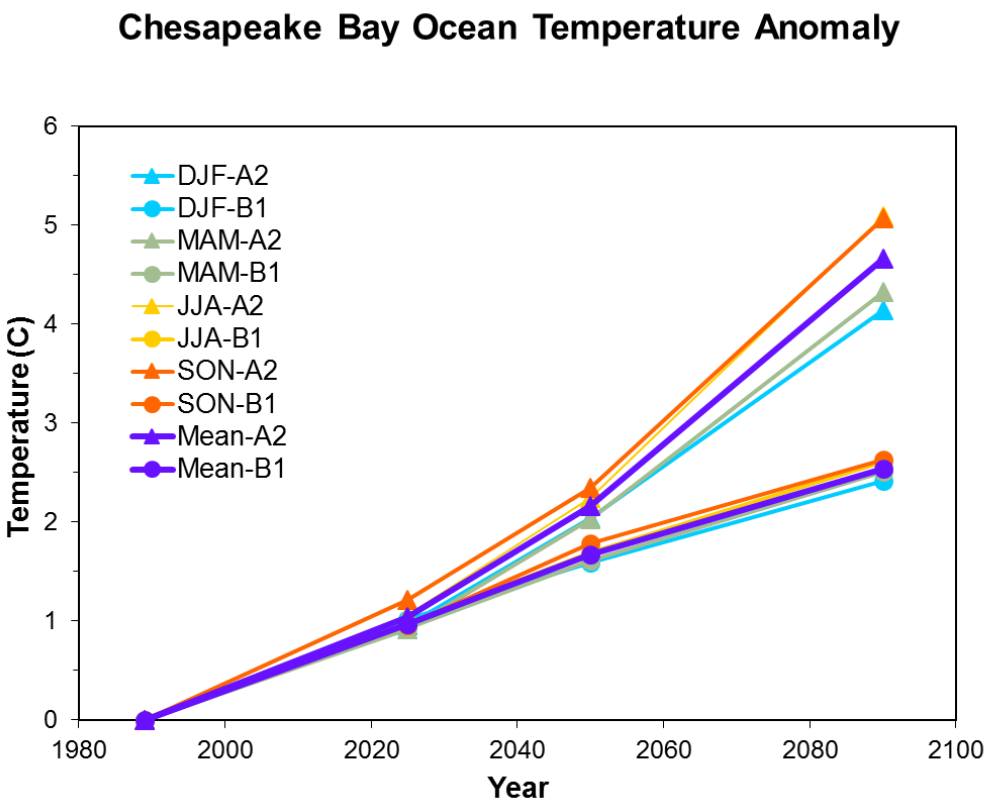

Figure 14. Victoria Cole's (2008) monthly Chesapeake Bay sea surface temperature (SST) anomalies from 1990-2090.

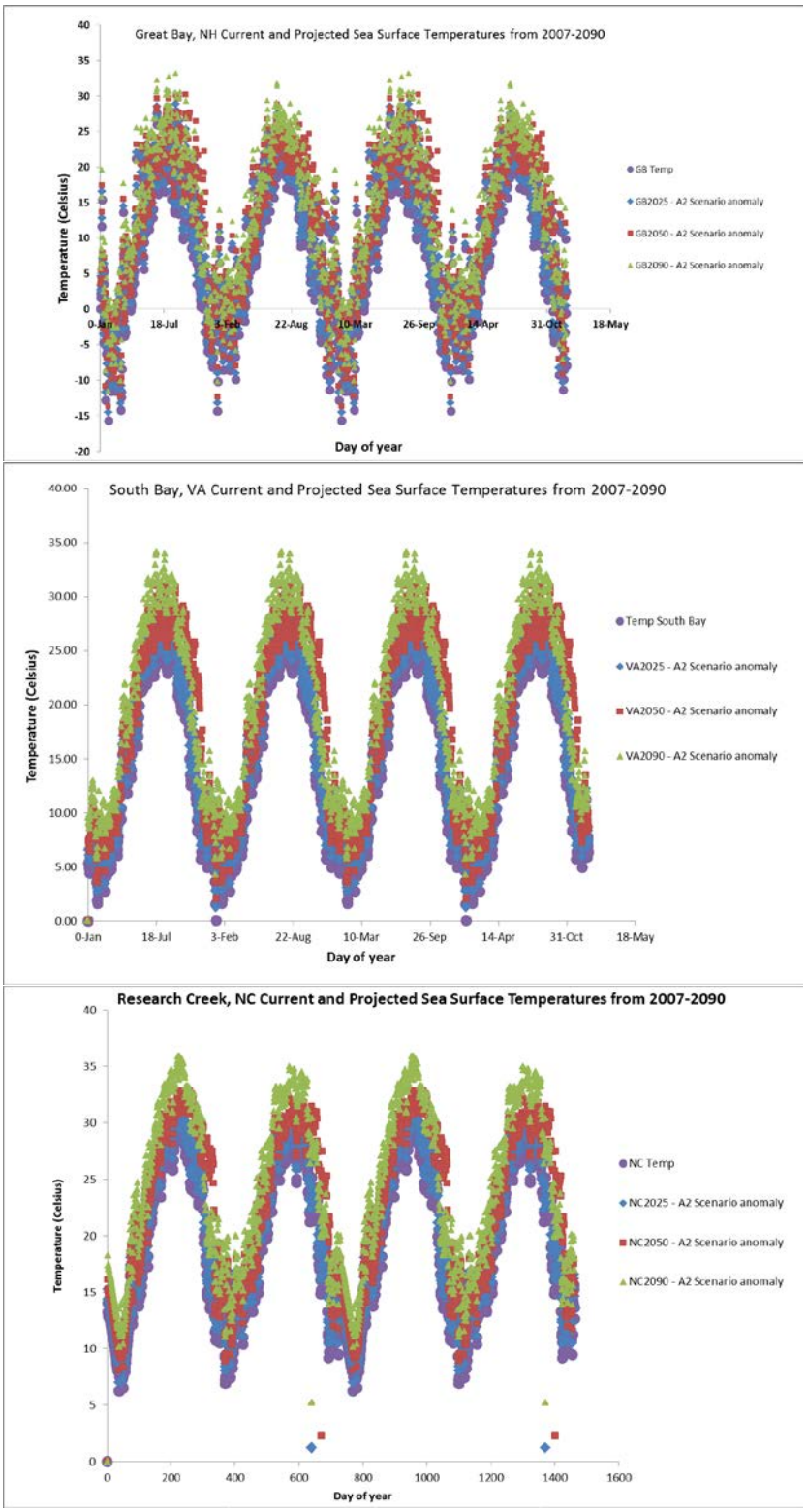

Figure 15. Current and projected temperature forcing functions for use in model climate simulations.

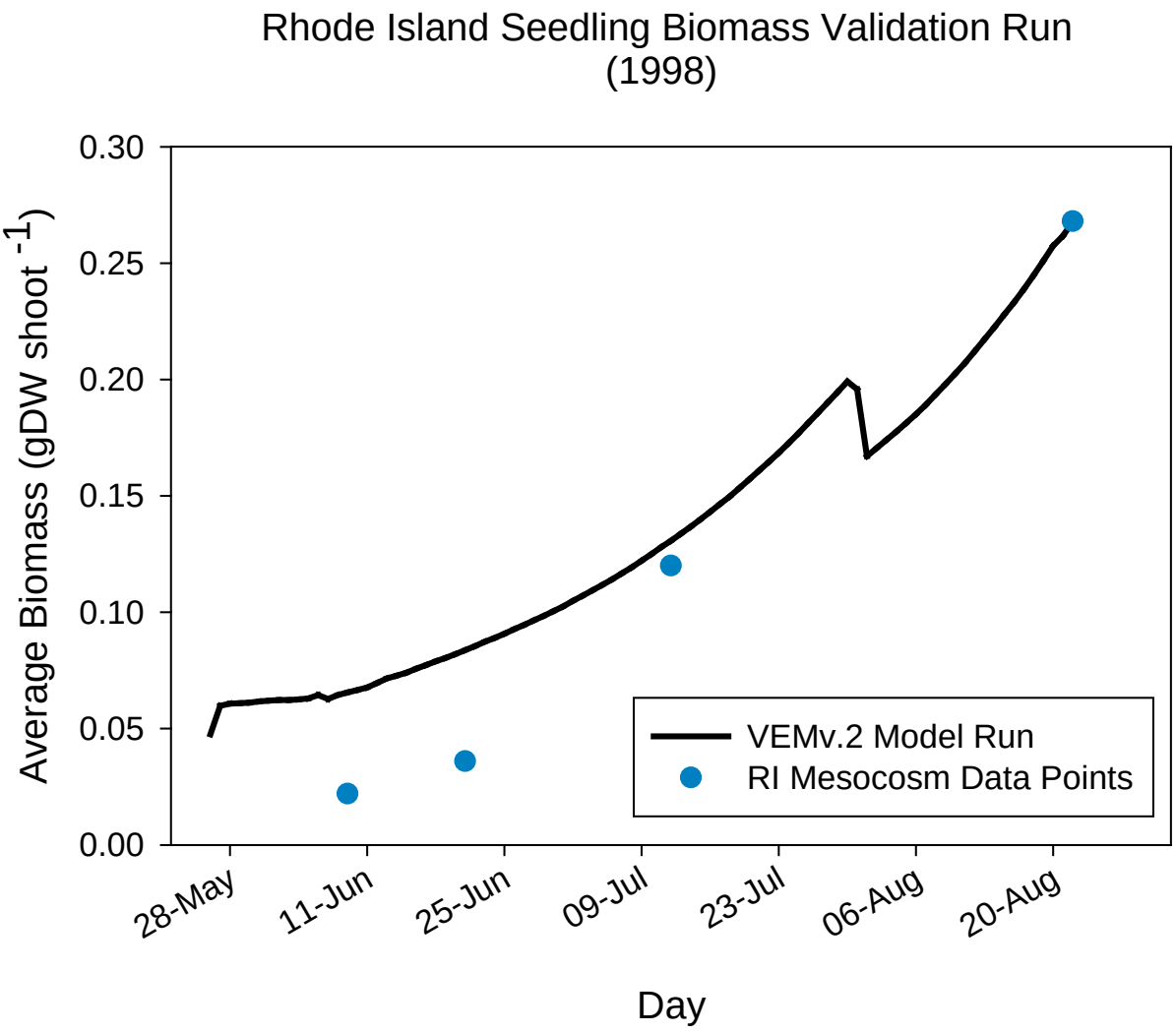

Figure 16. VEMv.2 modeled results versus observed biomass from a Rhode Island mesocosm study (Bintz 2002).

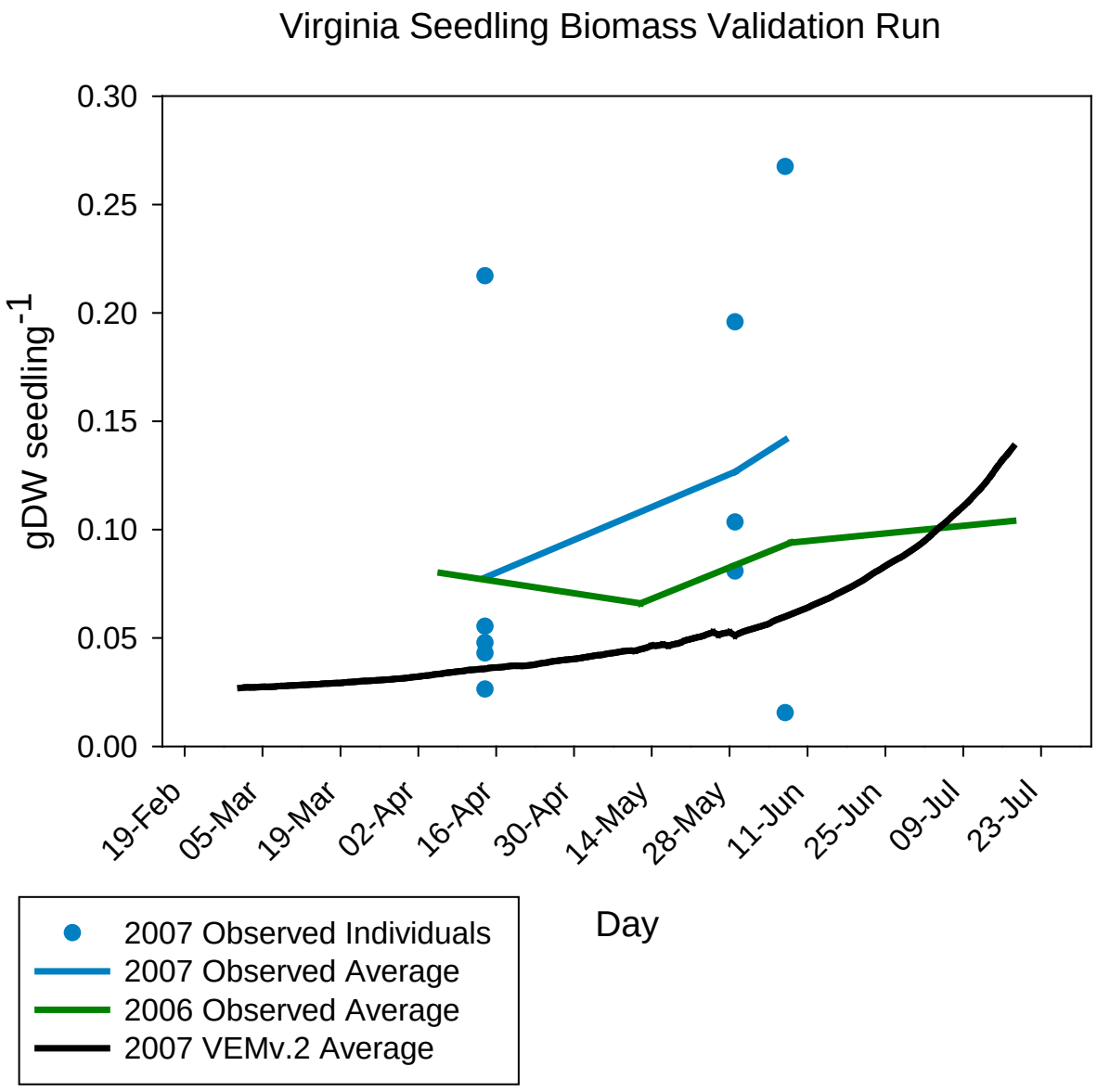

Figure 17. VEMv.2 modeled results versus observed biomass in Virginia's York River.

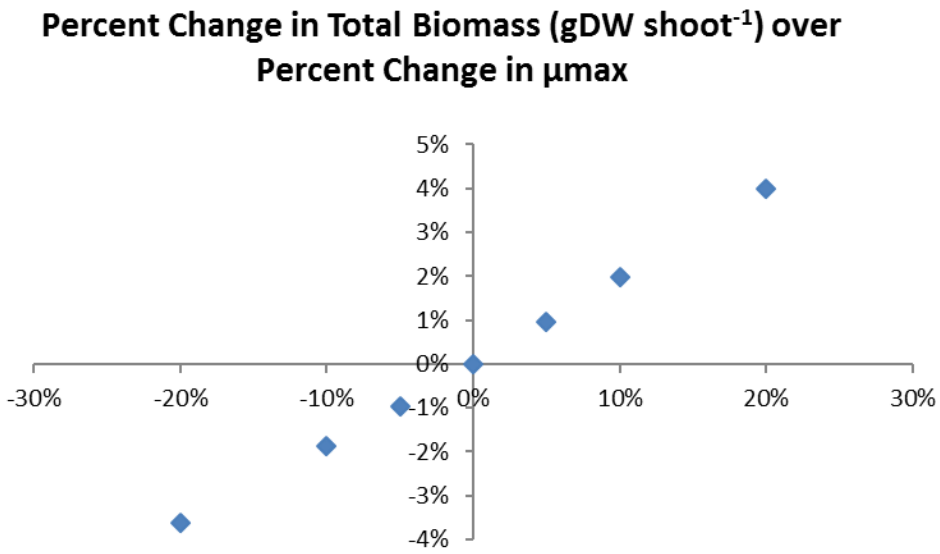

Figure 18. VEMv.2 seedling biomass sensitivity analysis. Horizontal axis shows change in µmax while vertical axis displays the change in total seedling biomass.

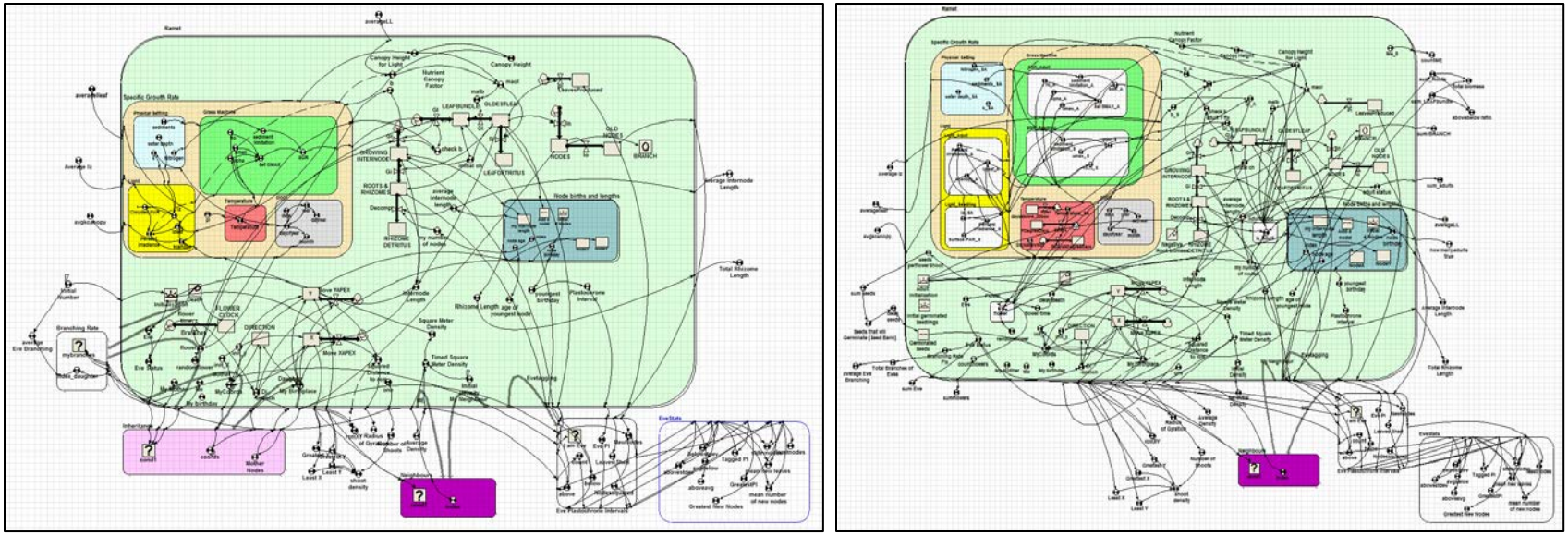

Figure 19. Simile software visual environment for VEMv.1 (left) and VEMv.2 (right).

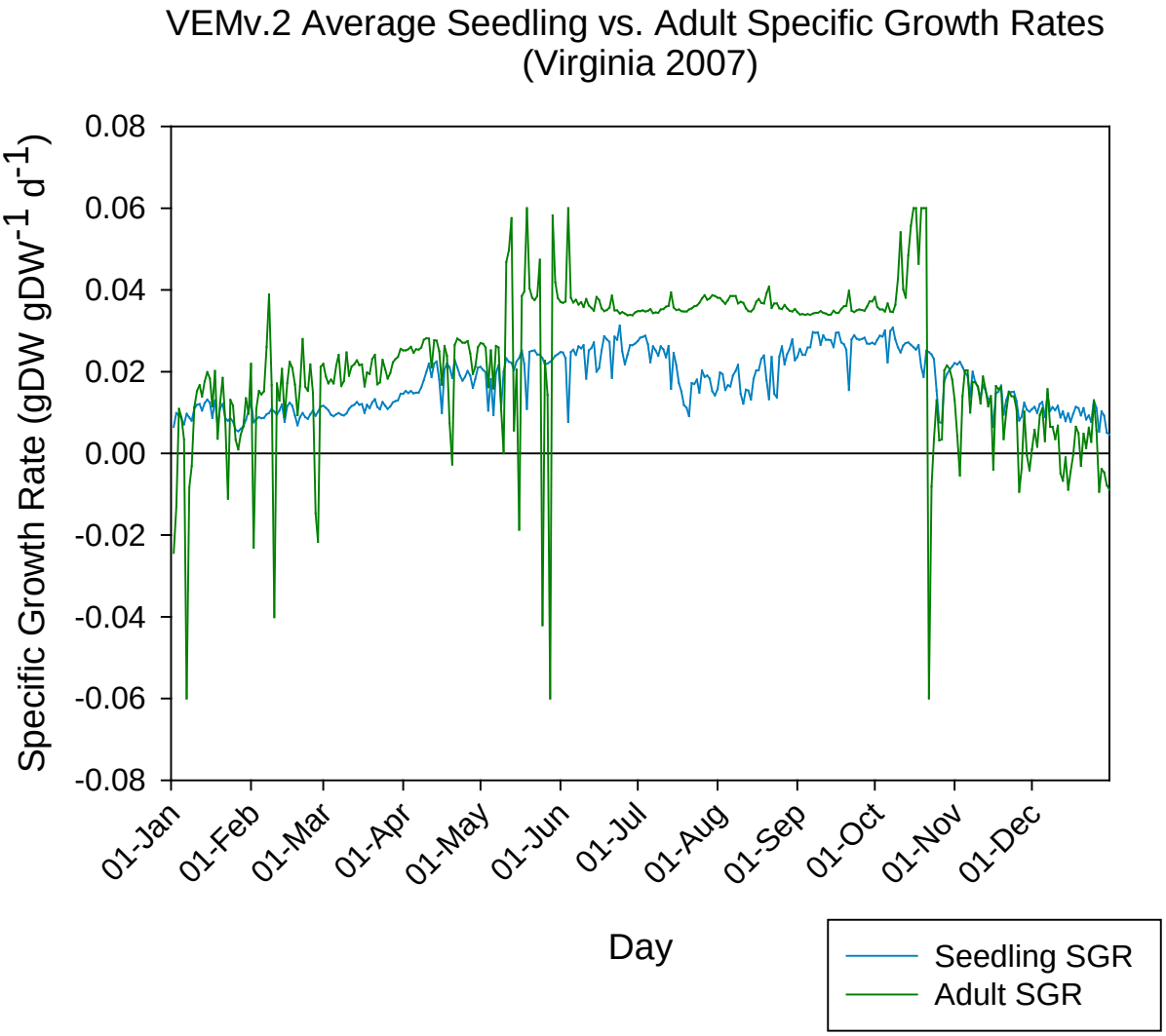

Figure 20. VEMv.2 Average Seedling vs. Adult Specific Growth Rates (Virginia).

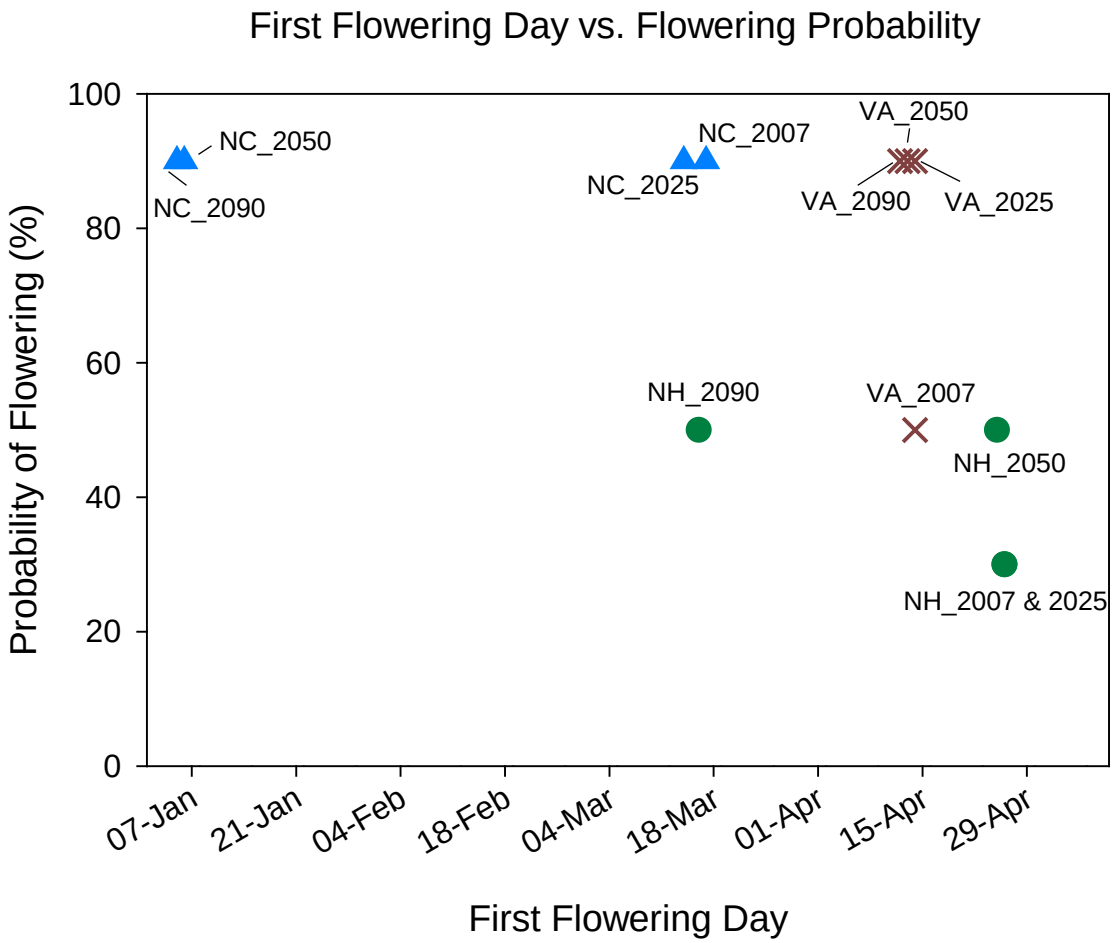

Figure 21. Reproductive phenology results: an individual's probability of flowering versus the first flowering day of the year across the three sites and four temperature scenarios.

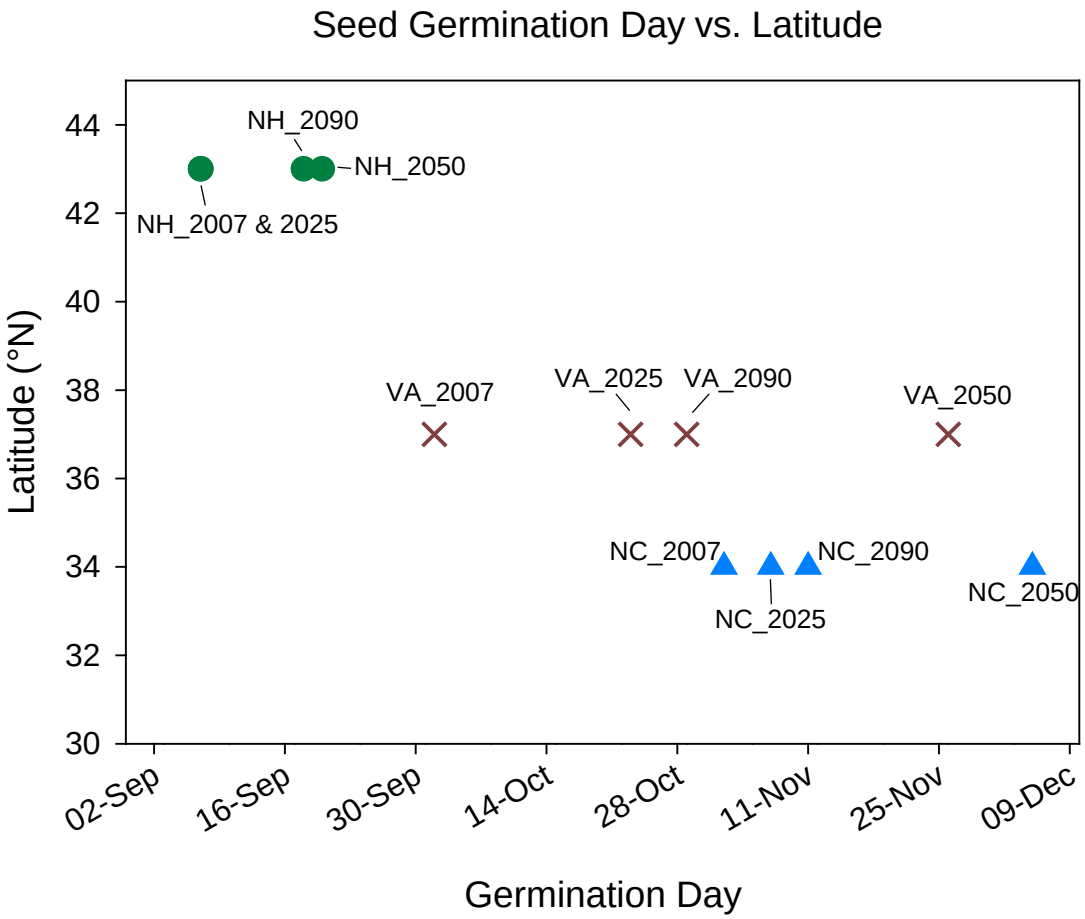

Figure 22. Reproductive phenology results: the effect of latitude on germination day over the model simulations.

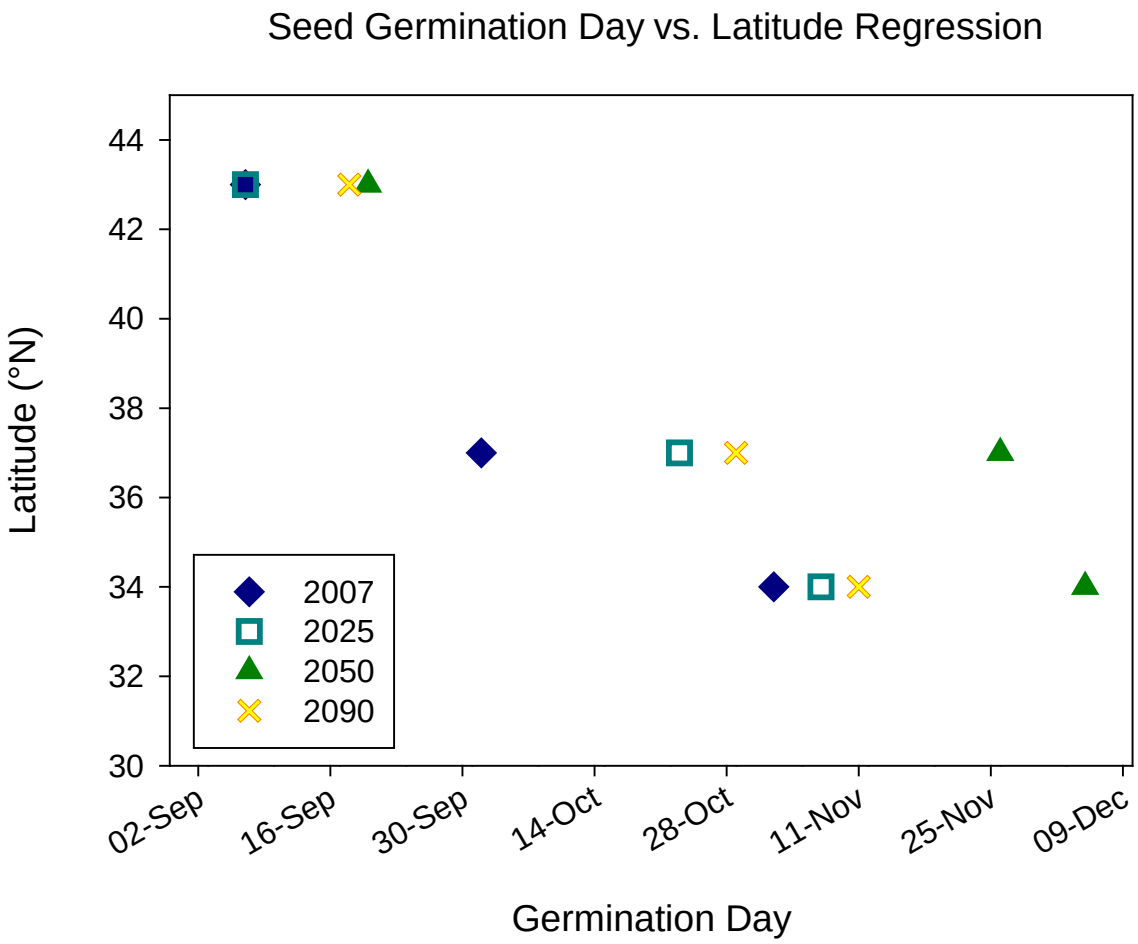

Figure 23. Reproductive phenology results: seed germination day latitude relationships.

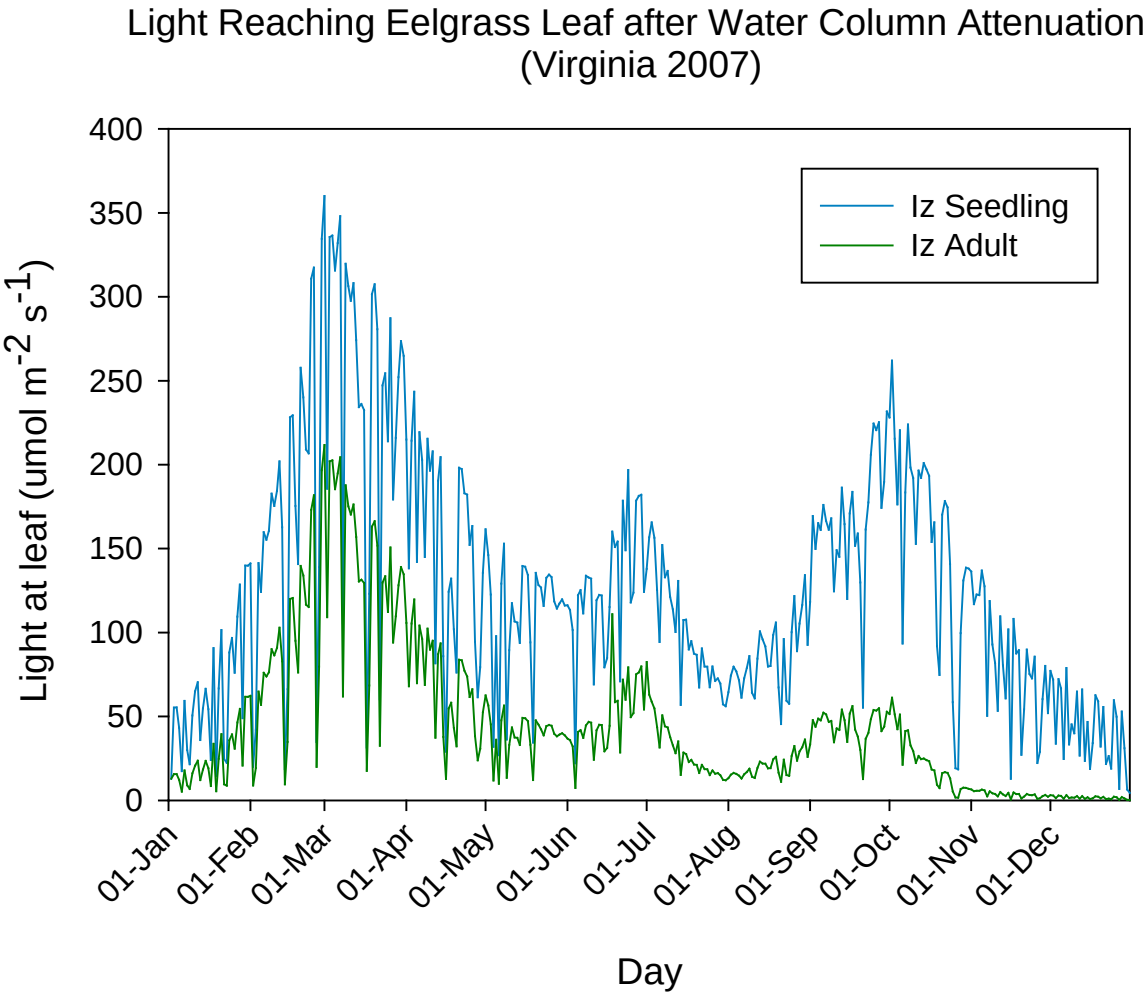

Figure 24. Modeled light reaching eelgrass leaf after attenuating through the water column (Virginia 2007).

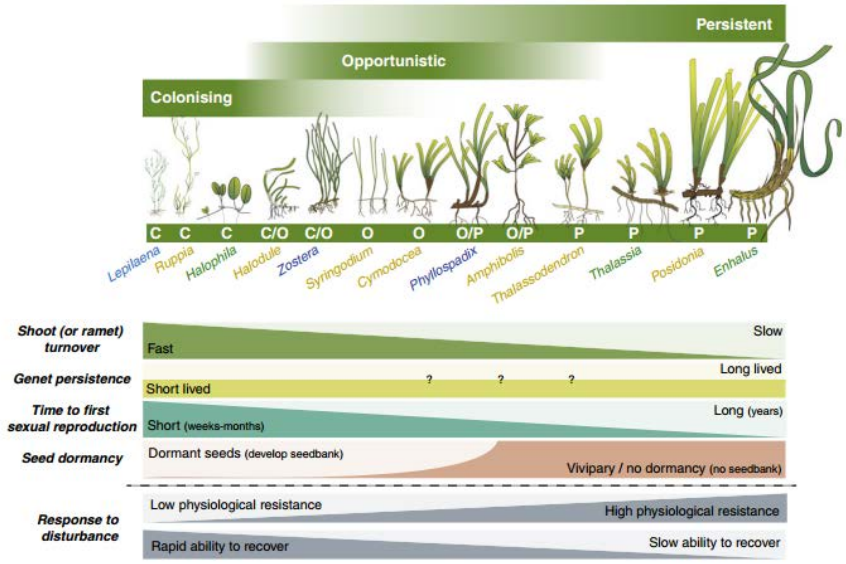

Figure 25. Diagram showing the dominant traits among colonizing (C), opportunistic (O), and persistent seagrass genera with respect to shoot turnover, genet persistence, time to reach sexual maturity and seed dormancy (Kilminster et al. 2015).

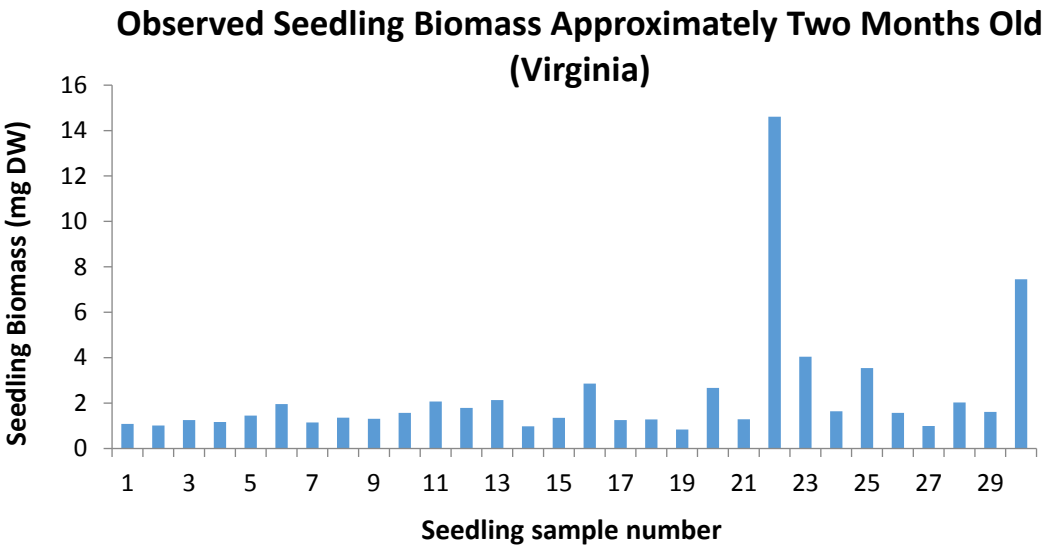

Figure 26. Measured seedling biomass approximately two months post germination. Seeds were collected from South Bay, VA and Mobjack Bay, VA. (Orth and Brush 2015).

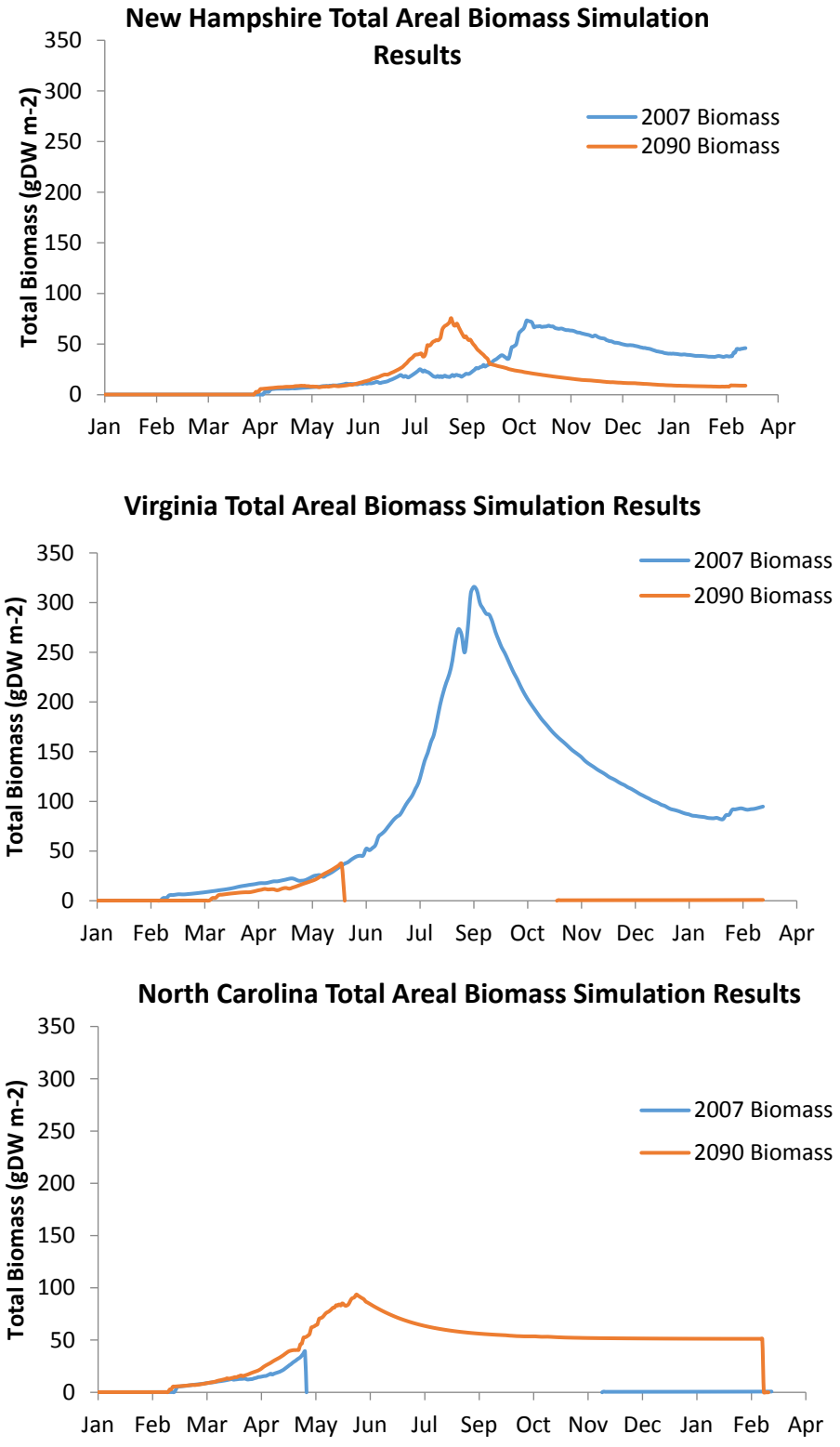

Figure 27. Total areal biomass climate simulation results years 2007 and 2090 (2050 forthcoming).

APPENDIX A – MODEL EQUATIONS

The following equations are reproduced from the VEMv.2 as implemented in SIMILE software.

Model (RI_calibration):

Time step index: 2

Variable Average Density :
Average Density =
sum({Timed_Square_Meter_Density})/count({Timed_Square_Meter_Density }) (real)
Where:
{Timed_Square_Meter_Density} = Value(s) of Ramet/Timed Square Meter Density

Comments:
Takes average of local shoot densities calculated for each individual to compare with total shoot density calculated from entire extent of X-Y coordinates. Units are ramets/m2.

Variable Average Internode Length :
Average Internode Length =
(sum({average_internode_length}))/(count({average_internode_length})) (real)
Where:
{average_internode_length} = Value(s) of Ramet/average internode length

Comments:
This calculates the average internode length for the entire population of ramets. Units are mm.

Variable Average Iz :
Average Iz = (sum({Iz_SA}))/(count({Iz_SA})) (real)
Where:
{Iz_SA} = Value(s) of Ramet/Specific Growth Rate/Light/Light_Seedling/Iz_SA

Variable BINTZ_MESO_PAR :
BINTZ_MESO_PAR = table(int(time())) (real)

Variable BINTZ_MESO_TEMP :

BINTZ_MESO_TEMP = table(int(time())) (real)

Variable Greatest X :
Greatest X = last(greatest({X})) (real)
Where:
{X} = Value(s) of Ramet/X

Variable Greatest Y :
Greatest Y = last(greatest({Y})) (real)
Where:
{Y} = Value(s) of Ramet/Y

Variable Least X :
Least X = last(least({X})) (real)
Where:
{X} = Value(s) of Ramet/X

Variable Least Y :
Least Y = last(least({Y})) (real)
Where:
{Y} = Value(s) of Ramet/Y

Variable NC1_NC2Comb_TEMP :
NC1_NC2Comb_TEMP = table(int(time())) (real)

Variable NC_NERRS_PAR :
NC_NERRS_PAR = table(int(time())) (real)

Variable NC_NERRS_TEMP :
NC_NERRS_TEMP = table(int(time())) (real)

Variable NH_GB_PAR :
NH_GB_PAR = table(int(time())) (real)

Variable NH_GB_TEMP :
NH_GB_TEMP = table(int(time())) (real)

Variable Number of Shoots :
Number of Shoots = count({one}) (int)
Where:
{one} = Value(s) of Ramet/one

Comments:
Total number of shoots in RAMET population model.

Variable Radius of Gyration :
Radius of Gyration =
sqrt((sum({Squared_Distance_to_rcm}))/(count({Squared_Distance_to_rcm}))) (real)
Where:
{Squared_Distance_to_rcm} = Value(s) of Ramet/Squared Distance to rcm

Comments:
Radius of Gyration. Calculated based on Sintes et al. 2005. Describes shape of patch formation

Variable Seeds that will Germinate [Seed Bank] :
Seeds that will Germinate [Seed Bank] = round(sum_seeds*0.4) (real)
Where:
sum_seeds = Value(s) of sum seeds

Comments:
40% of seeds produced are viable J.Jarvis 2014

Should I cut down this percentage further to account for low seedling establishment rates?

Variable Set Initial Density :
Set Initial Density = 50 (int)

Variable South Bay_2013_TEMP :
South Bay_2013_TEMP = table(int(time())) (real)

Variable Total Branches of Eves :
Total Branches of Eves = sum({Branching_Rate_Fix}) (int)
Where:
{Branching_Rate_Fix} = Value(s) of Ramet/Branching Rate Fix

Variable Total Rhizome Length :
Total Rhizome Length = (sum({Rhizome_Length}))/1000 (real)
Where:
{Rhizome_Length} = Value(s) of Ramet/Rhizome Length

Comments:
Takes sum of entire population's internode lengths to calculate the total rhizome length of the meadow. Converts from mm to meters.

Variable VA_HIB_PAR :
VA_HIB_PAR = table(int(time())) (real)

Variable VA_HIB_TEMP :
VA_HIB_TEMP = table(int(time())) (real)

Variable abovebelow ratio :
abovebelow ratio = sum_LEAFbundle/sum_Roots (real)
Where:
sum_Roots = Value(s) of sum_Roots
sum_LEAFbundle = Value(s) of sum_LEAFbundle

Variable average Eve Branching :
average Eve Branching = Total_Branches_of_Eves (real)
Where:
Total_Branches_of_Eves = Value(s) of Total Branches of Eves

Comments:
Total_Branches_of_Eves/Adult__Initialization

need to divide by initialization

Variable averageIleaf :
averageIleaf = (sum({Izleaf_A}))/(count({Izleaf_A})) (real)
Where:
{Izleaf_A} = Value(s) of Ramet/Specific Growth Rate/Light/Light_Adult/Izleaf_A

Variable averageLL :
averageLL =
sum({Canopy_Height__for_Light})/count({Canopy_Height__for_Light}) (real)

Where:

{Canopy_Height__for_Light} = Value(s) of Ramet/Canopy Height for Light

Comments:

Parameter used to calculate average canopy height of entire population. Units are cm.

Variable avgkcanopy :

avgkcanopy = (sum({kcanopy_A}))/(count({kcanopy_A})) (real)

Where:

{kcanopy_A} = Value(s) of Ramet/Specific Growth Rate/Light/Light_Adult/kcanopy_A

Variable countME :

countME = count({Me_0}) (int)

Where:

{Me_0} = Value(s) of Ramet/Me_0

Variable delay seeds :

delay seeds = delay(Seeds_that_will_Germinate__Seed_Bank_, 310) (real)

Where:

Seeds_that_will_Germinate__Seed_Bank_ = Value(s) of Seeds that will Germinate [Seed Bank]

Comments:

delays migration by x time units so that they only enter the model as growing individuals once they germinate. Germination occurs once the temperature decreases from 20 degrees C (accumulates 3 Germination degree days under 20 degrees C).

delay(Seeds_that_will_Germinate__Seed_Bank_, 30)

The seeds remained in the seed-bank until water tem-peratures decreased below 20◦C as this is when germination isinitiated in Chesapeake Bay populations (Moore et al., 1993).

In this case since they flower on day 12 and they would germinate on day 304 then the germinated seeds are delayed by 304-12 = 292 days for North Carolina

VA South Bay: They flower on day 103 and they germinate on day 273 therefore seeds are delayed by 273-103 = 170 days

Variable how many adults true :
how many adults true = howmanytrue({is_adult}) (int)
Where:
{is_adult} = Value(s) of Ramet/Adulthood/is_adult

Variable rcmXY :
rcmXY = [(sum({X}))/(count({X})), (sum({Y}))/(count({Y}))] (2 of real)
Where:
{X} = Value(s) of Ramet/X
{Y} = Value(s) of Ramet/Y

Comments:
Needed to calculate Radius of Gyration parameter. Methods from Sintes et al. (2005).

Variable shoot density :
shoot density = if time()==0 then 0 else Number_of__Shoots/((Greatest_Y-Least_Y)*(Greatest_X-Least_X)) (real)
Where:
Number_of__Shoots = Value(s) of Number of Shoots
Greatest_Y = Value(s) of Greatest Y
Greatest_X = Value(s) of Greatest X
Least_X = Value(s) of Least X
Least_Y = Value(s) of Least Y

Comments:
Shoot density based on furthest extent of X-Y coordinates and number of existing shoots. Units are shoots/m2

Variable sum BRANCH :
sum BRANCH = sum({BRANCH}) (real)
Where:
{BRANCH} = Value(s) of Ramet/BRANCH

Variable sum Eve :
sum Eve = sum({Eve_Status}) (int)
Where:
{Eve_Status} = Value(s) of Ramet/Eve Status

Variable sum seeds :
sum seeds = sum({seeds_perflowerShoot}) (int)

Where:
{seeds_perflowerShoot} = Value(s) of Ramet/seeds perflowerShoot

Comments:
Suming all seeds from all reproduction shoots. Therefore total # of seeds produced from adult flowering population per day

***Need to fix this. I need the seeds to only be counted on the one day of flowering and then remain dormant until the temperature cools to 15C, which could initiate germination.

Variable sum_LEAFbundle :
sum_LEAFbundle = sum({LEAFBUNDLE}) (real)
Where:
{LEAFBUNDLE} = Value(s) of Ramet/LEAFBUNDLE

Variable sum_Roots :
sum_Roots = sum({ROOTS___RHIZOMES}) (real)
Where:
{ROOTS___RHIZOMES} = Value(s) of Ramet/ROOTS & RHIZOMES

Variable sum_adults :
sum_adults = sum({adult_status}) (int)
Where:
{adult_status} = Value(s) of Ramet/adult status

Variable sumflowers :
sumflowers = sum({countflowers}) (int)
Where:
{countflowers} = Value(s) of Ramet/countflowers

Comments:
Does this need to be count flowers=true as opposed to just counting all the individuals?

Variable total biomass :
total biomass = sum_Roots+sum_LEAFbundle (real)
Where:
sum_LEAFbundle = Value(s) of sum_LEAFbundle
sum_Roots = Value(s) of sum_Roots

Submodel Neighbours :

Submodel "Neighbours" is an association submodel between "Ramet" and itself with roles "ME" and "My Neighbour".

Comments:

Relational/Conditional submodel used to identify a ramet's neighbours for the purposes of calculating a local shoot density.

Time step index: 1

Condition cond1 :

cond1 = fmod(time(), 10)==0 and (X_My_Neighbour<=(X_ME+0.1)) and (X_My_Neighbour>(X_ME-0.1)) and (Y_My_Neighbour<=(Y_ME+0.1)) and (Y_My_Neighbour>(Y_ME-0.1)) (cond_spec)

Where:

X_My_Neighbour = Value(s) of ../Ramet/X from submodel "Ramet" in role "My Neighbour"

X_ME = Value(s) of ../Ramet/X from submodel "Ramet" in role "ME"

{every_X} = Value(s) of ../Ramet/X

Y_My_Neighbour = Value(s) of ../Ramet/Y from submodel "Ramet" in role "My Neighbour"

Y_ME = Value(s) of ../Ramet/Y from submodel "Ramet" in role "ME"

{every_Y} = Value(s) of ../Ramet/Y

Comments:

Identify neighbours within 1 meter radius of individual

Variable index :

index = index(1) (int)

Submodel Ramet :

Submodel "Ramet" is a population submodel.

Comments:

The RAMET submodel. Contains all rules and mechanisms for growth and colonization of an eelgrass ramet.

Compartment DIRECTION :

Initial value = if Di__branch== 0.25 then element([My_Birthplace],3)+ 1.326 elseif Di__branch== 0.75 then element([My_Birthplace],3)- 1.326 else element([My_Birthplace],3) (real)

Where:

Di__branch = Value(s) of Di -branch

[My_Birthplace] = Value(s) of My Birthplace

Comments:

This stores the direction that a ramet is heading in so that it may be passed to daughter ramets. Units are radians.

Compartment GROWING INTERNODE :

Initial value = (((initial_ch*0.0012)/1.2)*4.7)/4 (real)

Where:

initial_ch = Value(s) of initial ch

Rate of change = + Gr - Gi

Comments:

A growing node-internode not yet "born". In reality, this reserve might be more closely associated with the meristem. Units are grams

Compartment LEAFBUNDLE :

Initial value = if is_adult then ((initial_ch*0.0012)/1.2)*3.7 else .013 (real)

Where:

initial_ch = Value(s) of initial ch

is_adult = Value(s) of Adulthood/is_adult

Rate of change = + Gl - Ol

Comments:

Leaves, excluding the eldest. Units are grams.

Seedling data taken from Brush/Orth data (estimate). .001 is representative of a seedling with one leaf when it is initialized (Jan 1) in the model or once it germinates. The maol for a seedling is .0012

Compartment LEAFDETRITUS :

Initial value = 0 (real)

Rate of change = + Sl

Comments:

State variable keeping track of leaf detritus. Units are grams.

Compartment LeavesProduced :

Initial value = 0 (real)

Rate of change = + Pl

Compartment NODES :

Initial value = if is_adult then 4 else 0 (real)
Where:
is_adult = Value(s) of Adulthood/is_adult

Rate of change = + In - Io
Comments:
This keeps track of the number of nodes up to a value of 4 and is only included here because this number represents the magic time when a shoot should create new branches! This stock and flow equation keeps track of new sets of 4* and less nodes to accomplish the node driven branching rate. Units are number of nodes.

Compartment OLD NODES :
Initial value = 0 (real)

Rate of change = + Io
Comments:
We might like the mortality of a shoot ot be related to how many nodes it has so we store the toal number here. This also provides a check on the population size value taken from the node population submodel. Units are # of nodes.

Compartment OLDESTLEAF :
Initial value = if is_adult then (initial_ch*0.0012)/1.2 else 0.015 (real)
Where:
initial_ch = Value(s) of initial ch
is_adult = Value(s) of Adulthood/is_adult

Rate of change = + Ol - Sl
Comments:
The "oldest leaf", or material that exceeds the maximum biomass threshold for the "leaves" compartment. Units are grams.

***If is_adult then_____ else ___

0.001 is taken from calibration data from M. Brush young seedlings

Compartment RHIZOME DETRITUS :
Initial value = 0 (real)

Rate of change = + Decomp
Comments:
Biomass of decayed and decaying internode material. Units are grams.

Compartment ROOTS & RHIZOMES :

Initial value = if is_adult then ((((initial_ch*0.0012)/1.2)*4.7)/4)*2.5 else .013 (real)

Where:

is_adult = Value(s) of Adulthood/is_adult

initial_ch = Value(s) of initial ch

Rate of change = + Gi - Decomp

Comments:

Established roots and rhizomes. This would be what you would actually measure as rhizome growth if you were taking field measurements. Units are g.

original: ((((initial_ch*0.0012)/1.2)*4.7)/4)*3

Compartment X :

Initial value = if time()<=1 then init_X else element([My_Birthplace],1) (real)

Where:

init_X = Value(s) of init_X

[My_Birthplace] = Value(s) of My Birthplace

Rate of change = + Move XAPEX

Comments:

X coordinate for current position in 2-D space. Units are meters. Initial Eve population starting location determined by planted grid location.

Compartment Y :

Initial value = if time()<=1 then init_y else element([My_Birthplace],2) (real)

Where:

init_y = Value(s) of init_y

[My_Birthplace] = Value(s) of My Birthplace

Rate of change = + Move YAPEX

Comments:

Y coordinate for current location in 2-D space. Units are meters. nitial Eve population starting location determined by planted grid location.

Creation Adult Initialization :

Adult Initialization = 0 (real)

Comments:

Initial number of ADULT ramets.

Creation Initial germinated Seedlings :
Initial germinated Seedlings = 3 (real)

Comments:
Number of seeds that will germinate and establish into seedlings. This number is taken from the adult model flowering components and feed into the seedling model (Not entirely correct). This is the initial number of seedlings at the start of a model run.

Flow Sl :
Sl = if OLDESTLEAF>maol then OLDESTLEAF-maol else 0 (real)
Where:
OLDESTLEAF = Value(s) of OLDESTLEAF
maol = Value(s) of maol

Comments:
This is telling the model when to "empty" the oldest leaf tank. This represents the flow of leaf material to a detrital pool and is a function of the oldest leaf mass value. Units are g/d

Flow Decomp : Decomposing Root and Rhizome material
Decomp = if time()<2 or my_number__of_nodes<1 then 0 elseif fmod(my_number__of_nodes,12)==0 then (average_internode_length*12)/317.28 else 0 (real)
Where:
my_number__of_nodes = Value(s) of my number of nodes
average_internode_length = Value(s) of average internode length

Comments:
This rate flows into a decomposing root and rhizome state variable to indicate that older internodes decay. This is another event-driven flow where every 12 nodes, the biomass associated with approximately 12 internodes is subtracted from the Roots and Rhizomes compartment. This is accomplished by taking the remainder of the current number of nodes after dividing by 12. If this value is equal to 0, the flow occurs. Also includes a conversion factor from mm to grams. Units are g/day

Flow Gi : Internode growth rate
Gi = if delay(OLDESTLEAF,1)>OLDESTLEAF then GROWING_INTERNODE else 0 (real)
Where:
OLDESTLEAF = Value(s) of OLDESTLEAF
GROWING_INTERNODE = Value(s) of GROWING INTERNODE

Comments:

A new node is "born" each time that an old leaf is sloughed off. Therefore, the "growing R&R" tank should empty each time this event occurs. To accomplish this, the model checks to see if the tank for "Oldest Leaf Mass" was larger on the last time step then on the current time step. If it's lower, then that tank has been emptied and a new node is born! The material in the "growingR&R" should therefore move on to the established "Roots&Rhizomes" tank. Units are g/d.

Flow Gl : leaf growth
Gl = Leaves*(if is_adult then GI_A else GI_S) (real)
Where:
Leaves = Value(s) of LEAFBUNDLE
is_adult = Value(s) of Adulthood/is_adult
GI_A = Value(s) of GI_A
GI_S = Value(s) of GI_S

Comments:
gdw/day

Flow Gr : Root and Rhizome growth
Gr = if flower=="true" then 0 else b*Gl/(1-b) (real)
Where:
b = Value(s) of b
flower = Value(s) of Flower/flower
Gl = Value(s) of Gl

Comments:
gdw/day

Flow In :
In = if delay(OLDESTLEAF,1)>OLDESTLEAF then 1 else 0 (real)
Where:
OLDESTLEAF = Value(s) of OLDESTLEAF

Comments:
This separate stock and flow formulation uses the tanks full of plant material (gdw) to simulate the "birth" of nodes. These births are then used to drive the reproduction rate of the shoot, so it's important to keep track of them here as well as within the separate node population submodel.

As with the formulation forr the Gi term, this flux checks to see if the oldest leaf has been sloughed off before deciding to add one node to the nodes tank. Units are nodes/day.

Flow Io :
Io = if NODES==4 then 4 else 0 (real)
Where:

NODES = Value(s) of NODES

Comments:

This empties the nodes out of "node" tank once new branches have been created. Units are nodes/day

Flow Move XAPEX :

Move XAPEX = if channel_is(Adult__Initialization) then Internode_Length*cos((element([My_Birthplace],3))) elseif Di__branch==0.25 then Internode_Length*cos((element([My_Birthplace],3)+1.326)) else Internode_Length*cos((element([My_Birthplace],3)-1.326)) (real)

Where:

Adult__Initialization = Value(s) of Adult Initialization
[My_Birthplace] = Value(s) of My Birthplace
Internode_Length = Value(s) of Internode Length
Di__branch = Value(s) of Di -branch

Comments:

This drives the movement of the ramet through 2-D space when a new internode is "born". To enable dichotomous branching, a Constant is set based on whether a ramet's mother had 5 nodes when it was "born". If this value is equal to 0.25, the ramet branches to the "right", if it is equal to 0.75 the new ramet branches to the "left" of the mother axis, and if this ramet is part of the initial "eve" population it continues in its original direction. This flow and the "Move YAPEX" flow use the same constant to determine direction.

The equation uses basic geometry/trigonometry to calculate the change in the X coordinate and a constant value for the branching angle specified in radians. The units for this flow are meters/day.

Flow Move YAPEX :

Move YAPEX = if channel_is(Adult__Initialization) then Internode_Length*sin((element([My_Birthplace],3))) elseif Di__branch==0.25 then Internode_Length*sin((element([My_Birthplace],3)+1.326)) else Internode_Length*sin((element([My_Birthplace],3)-1.326)) (real)

Where:

Adult__Initialization = Value(s) of Adult Initialization
[My_Birthplace] = Value(s) of My Birthplace
Internode_Length = Value(s) of Internode Length
Di__branch = Value(s) of Di -branch

Comments:

This drives the movement of the ramet through 2-D space when a new internode is "born". To enable dichotomous branching, a Constant is set based on whether a ramet's mother had 5 nodes when it was "born". If this value is equal to 0.25, the ramet branches to the "right", if it is equal to 0.75 the new ramet branches to

the "left" of the mother axis, and if this ramet is part of the initial "eve" population it continues in its original direction. This flow and the "Move XAPEX" flow use the same constant to determine direction.

The equation uses basic geometry/trigonometry to calculate the change in the Y coordinate and a constant value for the branching angle specified in radians. The units for this flow are meters/day.

Flow Ol : transition of leaves to older leaf compartment.
Ol = if LEAFBUNDLE>malb then LEAFBUNDLE-malb else 0 (real)
Where:
LEAFBUNDLE = Value(s) of LEAFBUNDLE
malb = Value(s) of malb

Comments:
This rate governs when an old leaf transitions into it's final position before being sloughed off. This is accomplished by comparing the "leaves" tank with a maximum biomass that is set by the canopy height multiplied by a leaf distribution factor and weight conversion factor which is then multiplied by a Nitrogen factor which adjusts this biomass amount according to the Nitrogen conditions set by the user. Higher nitrogen values or lower light levels result in a higher maximum biomass for the leaf compartment, as evidence points to elongation of leaves. Units are g/d

You'll notice that the way this equation is set results in a continuous flow of material to the "oldest" compartment once the maxbiomass has been achieved. This isn't quite how reality works, but it is a way to make this part of the model function with stock and flows rather than creating a separate leaf population submodel (and its inherent trickiness for assemblying values).

Flow Pl :
Pl = if delay(OLDESTLEAF,1)>OLDESTLEAF then 1 else 0 (real)
Where:
OLDESTLEAF = Value(s) of OLDESTLEAF

Immigration Germinated Seeds :
Germinated Seeds = if time(0)==314 then 40 else round(delay__seeds) (real)
Where:
delay__seeds = Value(s) of ../delay seeds

Comments:
The number of seeds that will germinate from the seed bank enter the model as inidividuals 1 time step after the reproductive shoots flower. For this reason,

their SGR is 0 for the first year since technically the seeds have not yet germinated. Germination takes place once temperatures cool after the hot summer. Therefore the seeds are in a period of dormany until year 2.

Waiting until January for the SGR to kick in is also representative of germination in late fall as there is often a delay between germination and when the first photosynthetic leaves emerge from the ground's surface.

You don't need to count or sum this value you can just use the bos itself to do so.

Loss Death : Mortality
Death = delay_death (real)
Where:
delay_death = Value(s) of delay death

Comments:
Mortality due to flowering. Units are ramets/day

if FLOWER__CLOCK==41 then 1 else 0 - original

Do I meed tp add a mortality of biomass is 0? Is there any biomass below 0?

Individuals are removed at the start of the time step following the one in which their number came up therefore I wanted to delay the time from flower to death.

Loss Negative Root biomass :
Negative Root biomass = if ROOTS___RHIZOMES<-0.1 then 1 else 0 (int)
Where:
ROOTS___RHIZOMES = Value(s) of ROOTS & RHIZOMES

Comments:
This code signifies death of an individual that has lost all of its root biomass (i.e. below 0). Previously, root biomass was just going very negative but the individual was not dying.

Reproduction BRANCH : Asexual Reproduction/Branching
BRANCH = if NODES==4 and delay(NODES,1)==3 then 1 elseif NODES==3 and delay(NODES,1)==2 then 1 else 0 (real)
Where:
NODES = Value(s) of NODES

Comments:

This population process relates the number of nodes produced since the last lateral shoot to help time the creation of a new lateral shoot. Data used to choose a value of 4/3(?) nodes between shoots was taken from ponds/mesocosm data. Units are number of shoots per day.

original *if NODES==6 and delay(NODES,1)==5 then 1 elseif NODES==5 and delay(NODES,1)==4 then 1 else 0*

Once you produce a BRANCH you are an adult.

Variable Branching Rate Fix :
Branching Rate Fix = in_progenitor(Eve_Status) (int)
Where:
Eve_Status = Value(s) of Eve Status

Comments:
Currently you have two associations in the model which are set up as parent/child associations, i.e., the Branching Rate and Inheritance submodels. I can't see why you need an association for branching rate at all, since all it is doing is counting the number of ramets that are branches of Eves. To get this count, you could just add a variable in the ramet submodel with the equation 'in_progenitor(Eve_Status)' which would have the value 1 if its parent was Eve and 0 otherwise. You would then sum this outside the submodel to get the total branches of Eves, to divide by the initial number to get the average Eve branching. You could then delete the Branching Rate submodel.

Variable Canopy Height :
Canopy Height =
Nutrient__Canopy_Factor*Canopy_Height__for_Light (real)
Where:
Nutrient__Canopy_Factor = Value(s) of Nutrient Canopy Factor
Canopy_Height__for_Light = Value(s) of Canopy Height for Light

Variable Canopy Height for Light : User defined maximum canopy height for depth
Canopy Height for Light = if time()<=1 or time()==init_time() then 30 else 28.55*(((if is_adult then Percent_Irradiance_A else Percent_Irradiance_S))^-.60) (real)
Where:
Percent_Irradiance_S = Value(s) of Specific Growth Rate/Light/Light_Seedling/Percent Irradiance_S

Percent_Irradiance_A = Value(s) of Specific Growth Rate/Light/Light_Adult/Percent Irradiance_A

is_adult = Value(s) of Adulthood/is_adult

Minimum = 14, Maximum = 150

Comments:

At the moment this is a user defined varible. It would be nice to have a canopy height versus depth relationship! Units are centimeters.

Variable Di -branch :

Di -branch = if Eve_Status==1 then 5 elseif in_progenitor(NODES)==5 then 0.25 else 0.75 (real)

Where:

NODES = Value(s) of NODES

Eve_Status = Value(s) of Eve Status

Variable Eve :

Eve = not channel_is(BRANCH) (boolean)

Where:

BRANCH = Value(s) of BRANCH

Comments:

Used to determine if an individual is part of the "eve" population.

-I do not think this works. It just tells you the same what index(1) or Me tells you since even newly added individuals during a run were returning a "true" value.

if channel_is(BRANCH) or channel_is(germinated seeds) then 0 else 1

Variable Eve Status :

Eve Status = if Eve=="true" then 1 else 0 (int)

Where:

Eve = Value(s) of Eve

Comments:

Boolean function to signal Eve Status

Variable GI_A :

GI_A = if flower=="true" then 0 else SGR_A (real)

Where:

SGR_A = Value(s) of Specific Growth Rate/Grass Machine/SGR_Adult/SGR_A

flower = Value(s) of Flower/flower

Variable GI_S :
GI_S = if flower=="true" then 0 else SGR_S (real)
Where:
SGR_S = Value(s) of Specific Growth Rate/Grass Machine/SGR_Seedling/SGR_S
flower = Value(s) of Flower/flower

Variable Initial Density :
Initial Density = Set_Initial_Density (int)
Where:
Set_Initial_Density = Value(s) of ../Set Initial Density

Comments:
Density of shoots specified for start of simulation. Units should be shoots/m2

Variable Internode Length :
Internode Length = if time(1)==1 then (ROOTS___RHIZOMES*317.28)/1000 elseif delay(OLDESTLEAF,1)>OLDESTLEAF then 317.28*GROWING_INTERNODE/1000 else 0 (real)
Where:
OLDESTLEAF = Value(s) of OLDESTLEAF
GROWING_INTERNODE = Value(s) of GROWING INTERNODE
ROOTS___RHIZOMES = Value(s) of ROOTS & RHIZOMES

Comments:
This translates the biomass associated with the "birth" of a node into an internode length that can be used to drive the spatial coordinates of the simulations. Units are m.

Variable Me :
Me = index(1) (int)

Comments:
My identity

Variable Me_0 :
Me_0 = index(1) (int)

Variable My Birthplace :
My Birthplace = if Eve_Status==1 then [MyCoords] else in_progenitor([MyCoords]) (3 of real)
Where:

[MyCoords] = Value(s) of MyCoords
Eve_Status = Value(s) of Eve Status

Comments:
Coordinates and direction of mother ramet at time of branching passed from Inheritence submodel.

.....Jasper The Inheritance submodel is used to pass 'My Coords' of the parent to
'My Birthplace' of the offspring. This could be done more simply by adding a direct influence, and setting the equation for 'My Birthplace' to 'if Eve_Status==1 then [My_Coords] else in_progenitor([My_Coords])'.
The condition is needed because the condition in the association includes a clause that makes it exist between Eve individuals and themselves, so their birthplaces are their own coords.

The same submodel also passes values from NODES in the parent to Di-branch in the offspring. This connection could also be made directly
with the in_progenitor() function, after which the Inheritance submodel
could be deleted.

Variable My Mother :
My Mother = parent(1) (int)

Comments:
Identity of mother within population

Variable My birthday :
My birthday = init_time() (real)

Comments:
Self-explanatory. Units are day of simulation.

Variable MyCoords :
MyCoords = if time()<=1 then[init_X,init_y,rand_var(0,6.36)] elseif channel_is(Germinated_Seeds) then[init_X,init_y,rand_var(0,6.36)] else last([X,Y,DIRECTION]) (3 of real)
Where:
init_X = Value(s) of init_X
init_y = Value(s) of init_y
Germinated_Seeds = Value(s) of Germinated Seeds
X = Value(s) of X

Y = Value(s) of Y
DIRECTION = Value(s) of DIRECTION

Comments:
Array used to pass an individual's coordinates and direction to herself or her daughter ramet.

if time()<=1 then[rand_var(0,1.5),rand_var(0,.95),rand_var(0,6.36)]else last([X,Y,DIRECTION])

previous: if time()<=1 then[init_X,init_y,rand_var(0,6.36)] else last([X,Y,DIRECTION]) when init_x and y were an input table number

Variable Nutrient Canopy Factor :
Nutrient Canopy Factor = (.206*Log(Nitrogen_SA))+1.7679 (real)
Where:
Nitrogen_SA = Value(s) of Specific Growth Rate/Physical Setting/Nitrogen_SA

Comments:
An observation in many enrichment experiments has been that leaves elongate in response to higher nitrogen concentrations. This multiplicative factor takes the predicted weight of the longest leaf (which limits for the leaf compartment are based upon) and increases or decreases these limits under high or low nitrogen conditions, respectively. The regression equation was taken from Roberts et al.

Variable Plastochrone Interval :
Plastochrone Interval = if delay(youngest__birthday, 1)==youngest__birthday then 0 else youngest__birthday-delay(youngest__birthday,1) (real)
Where:
youngest__birthday = Value(s) of youngest birthday

Comments:
Calculates node plastochrone interval (which is, by default, also the leaf plastochrone interval). Units are days.

Variable Rhizome Length :
Rhizome Length = sum({my_internode_length}) (real)
Where:
{my_internode_length} = Value(s) of Node births and lengths/my internode length

Comments:

Sums the entire node submodel for a ramet to calculate its total rhizome length. This will include ALL rhizome ever produced, even if some of the older material eventually dies off. Units are mm.

Variable Square Meter Density :

Square Meter Density = if time()<=1 then 75 elseif fmod(time(),10)==0 then Timed_Square_Meter_Density else delay(Timed_Square_Meter_Density,fmod(time(),10)) (real)

Where:

Timed_Square_Meter_Density = Value(s) of Timed Square Meter Density

Comments:

To increase simulation time, a local shoot density is only calculated every 10th time step. This variable take that value from the "Timed Square Density" variable and associates it with the timesteps in between so that a constant shoot density is maintained for the intervening simulation steps. It will look something like this:

Time	Shoot Density
10	300
11	300
12	300
....19	300
20	update to new density - 320
21	320

and so on.

Units are ramets/m2

if time()<=1 then 75 elseif fmod(time(),10)==0 then Timed_Square_Meter_Density else delay(Timed_Square_Meter_Density,fmod(time(),10)) --former code, playing around to see if this is the reason the the adult SGR changes among individuals.

if time()<=1 then 75 elseif fmod(time(),10)==0 then Timed_Square_Meter_Density else delay(Timed_Square_Meter_Density,fmod(time(),10))

Variable Squared Distance to rcm :

Squared Distance to rcm = (X-(element([rcmXY],1)))^2 + (Y-(element([rcmXY],2)))^2 (1*1*1)

Where:

X = Value(s) of X

Y = Value(s) of Y

[rcmXY] = Value(s) of ../rcmXY

Comments:

Needed to calculate Radius of Gyration parameter. From Sintes et al. 2005

Variable Timed Square Meter Density :

Timed Square Meter Density = if time()<=1 then Initial_Density elseif fmod(time(),10)==0 then (count({index_My_Neighbour})*25) else 0 (real)

Where:

Initial_Density = Value(s) of Initial Density

{index_My_Neighbour} = Value(s) of ../Neighbours/index for submodel "Ramet" in role "My Neighbour"

index_ME = Value(s) of ../Neighbours/index for submodel "Ramet" in role "ME"

{every_index} = Value(s) of ../Neighbours/index

Comments:

To increase simulation speed, the conditional neighbour model only checks for neighbours every 10th time step. This variable stores a value from the submodel on each of these 10th timesteps.

Variable adult status :

adult status = if is_adult=="true" then 1 else 0 (int)

Where:

is_adult = Value(s) of Adulthood/is_adult

Variable age of youngest node :

age of youngest node = least({node_age}) (real)

Where:

{node_age} = Value(s) of Node births and lengths/node age

Comments:

Units are days.

Variable average internode length :

average internode length = if my_number__of_nodes>0 then sum({my_internode_length})/my_number__of_nodes else 0 (real)

Where:

my_number__of_nodes = Value(s) of my number of nodes

{my_internode_length} = Value(s) of Node births and lengths/my internode length

Comments:

Self explanatory. Units are mm.

Variable b :

b = if is_adult =="true" then b_A else b_S (real)
Where:
b_A = Value(s) of b_A
b_S = Value(s) of b_S
is_adult = Value(s) of Adulthood/is_adult

Variable b_A : % Belowground Allocation
b_A = if Temperature_SA< 8.064 then 0.65 else (-.0346*Temperature_SA)+1.029 (real)
Where:
Temperature_SA = Value(s) of Specific Growth Rate/Temperature/Temperature_SA

Comments:
Function of temperature. Based on data from 1999 Mesocosm. Excel file assocated with parameterization named "k1_temperatureallocation". Because data does not include situations with VERY cold temperatures or very high belowground partitioning, the if-then statement caps the max percentage at 0.75 at a temperature of 7.385 degrees (the temperature associated with this value using the temperature-partitioning regression equation). units are a percentage of growth (gdw/gdw)

Variable b_S : % below ground allocation
b_S = if Iz_SA<=90 then 0.30 elseif Iz_SA >=499 then 0.51 else 0.43 (real)
Where:
Iz_SA = Value(s) of Specific Growth Rate/Light/Light_Seedling/Iz_SA

Comments:
Iz below ground biomass seedling relationship taken from Joanne Bintz's dissertation work. Units are a percentage of growth (gdw/gdw)

Less below ground allocation when light is limited.

Variable check b adult ? fix :
check b adult ? fix = if Temperature_SA< 8.064 then 0.75 else Gr/(Gr+Gl) (real)
Where:
Temperature_SA = Value(s) of Specific Growth Rate/Temperature/Temperature_SA
Gl = Value(s) of Gl
Gr = Value(s) of Gr

Variable countflowers :
countflowers = if flower=="true" then 1 else 0 (int)
Where:
flower = Value(s) of Flower/flower

Variable delay death :
delay death = delay(flower_time,60) (real)
Where:
flower_time = Value(s) of flower time

Comments:
However, recent observations
of an annual form of Zostera marina L. indicate that flowering
plants in some populations are produced during the first year of
growth. All flowering shoots, regardless of their age, will die by
the end of the growing season, and usually by the end of the flowering period. Therefore, I chose 60 days.

Delay death until 21 degrees around when fruit maturation is complete.

Variable flower time :
flower time = if flower=="true" then 1 else 0 (int)
Where:
flower = Value(s) of Flower/flower

Variable init_X :
init_X = if time()<=1 then rand_const(0,1.6) else last(X) (real)
Where:
X = Value(s) of X

Comments:
previous if time()<=1 then rand_const(0,1.6) elseif channel_is(Germinated_Seeds) then rand_const(0,1.6) else last(X)

Variable init_y :
init_y = if time()<=1 then rand_const(0,1.05) else last(Y) (real)
Where:
Y = Value(s) of Y

Comments:
a table?

Variable initial ch :

initial ch = if is_adult then 12 else 4 (real)
Where:
is_adult = Value(s) of Adulthood/is_adult

Comments:
cm -JF
Adults is rand_var(12.17,25.92)
Seedlings is randvar(2,9)
This variable is initial length of seedlings in cm

Variable malb :
malb = if is_adult then Oldest_Leaf_Mass* 3.7 else Oldest_Leaf_Mass (real)
Where:
Oldest_Leaf_Mass = Value(s) of maol
is_adult = Value(s) of Adulthood/is_adult

Comments:
This value is an estimate of how much four leaves should weigh based on the canopy height of the shoot. It is also assumed that the third leaf is the longest leaf and determines the canopy height. Distribution of weight amongst the leaves was determined from empirical data taken from mesocosms and field sites in southern Rhode Island. Kept in file "Leaf sizes". Units are gdw.

Variable maol : Oldest Leaf Mass determined by Nutrient conditions
maol = if is_adult then Nutrient__Canopy_Factor*(Canopy_Height__for_Light* 0.0012 / 1.2) else 0.015*1.2 (real)
Where:
Nutrient__Canopy_Factor = Value(s) of Nutrient Canopy Factor
Canopy_Height__for_Light = Value(s) of Canopy Height for Light
is_adult = Value(s) of Adulthood/is_adult

Comments:
Uses the canopy height as representative of the third leaf. Empirically, the third leaf represents 1.2xoldest leaf length, so the oldest leaf weight is calculated based on this trick as well as a limitation factor based on nutrient conditions. The .0012 factor is the conversion factor to get from length (cm) to biomass (g). Taken from Nixon Lab data. File = Leaf SIzes

***If is_adult then ____ else 0.0012
Units of biomass, seedling set at .0012 g DW (biomass) do not confuse with the conversion factor (coincidence that they are the same value). .0012 taken from Brush seedling data. Add a bit more perhaps .002 so that seedlings are not creating new nodes in one week or less from model calibration.

Variable my number of nodes :
my number of nodes = count({index}) (int)
Where:
{index} = Value(s) of Node births and lengths/index

Comments:
Counts number of instances in node submodel to determine how many nodes exist for an individual ramet.

Variable one :
one = 1 (int)

Comments:
Parameter used to count number of individuals outside RAMET submodel

Variable randomflower :
randomflower = rand_const(0,1) (real)

Comments:
Random constant to determine probablistically if a plant should flower or not.

Variable seeds perflowerShoot :
seeds perflowerShoot = if flower=="true" then 10 else 0 (int)
Where:
flower = Value(s) of Flower/flower

Comments:
After april, the flowering shoots will have set out 10 seeds per reproductive shoot. J.Jarvis 2014

Silberhorn 1983 (~23 seeds per flowering shoot). Another parameter to consider.

Variable youngest birthday :
youngest birthday = greatest({node_birthday}) (real)
Where:
{node_birthday} = Value(s) of Node births and lengths/node birthday

Comments:
Units are day of simulation

Submodel Ramet/Flower :

Variable flower :

flower = if Temperature_SA>=15 and (time(0)==4 or time(0)==369 or time(0)==734 or time(0)==1099) and my_number__of_nodes>=3 and randomflower<=(if Daysabove20<=100 then 0.3 elseif Daysabove20>=150 then 0.9 else 0.5) then "true" else "false" (boolean)

Where:

randomflower = Value(s) of ../randomflower

Temperature_SA = Value(s) of ../Specific Growth Rate/Temperature/Temperature_SA

my_number__of_nodes = Value(s) of ../my number of nodes

Daysabove20 = Value(s) of ../Specific Growth Rate/Temperature/Daysabove20

Comments:

JF current code: This code causes an individual to flower once the temperature degree day compartment reaches 3 days above an average of 15 degrees C and the individual has more than 3 nodes (adulthood). The Daysabove20 is a parameter that is linked to the temperature and dictates the set probability of flowering. The flowering is not a percentage of the total shoots but of the adult shoots that are capable of flowering. At the moment, you need to manually input the index aka "day" of year the 3rd 15 C degree day is met.

L.Harris' previos code: Hybrid mechanistic-stochastic parameter to decide if an individual should flower. Linked to temperature, time of year, and overall probability. Boolean.

Flowering percentages were chosen after performing a review of eelgrass reproductive shoot percentages in the literature.

GB: ~50 days above 20 per year
VA South Bay: ~126 days above 20 per year
NC ~180 days above 20 per year

How do I return just the first "true" value? Cannot unless it is an array.

previous code before the degree days compartment: if Temperature_SA>=16 and delay(Temperature_SA,1)<15 and my_number__of_nodes>3 and randomflower<=(if Daysabove20<=100 then 0.3 elseif Daysabove20>=150 then 0.9 else 0.5) then "true" else "false"

Submodel Ramet/Adulthood :

Variable is_adult :

is_adult = if channel_is(Adult__Initialization) or channel_is(BRANCH) or my_number__of_nodes>=4 then "true" else "false" (boolean)

Where:

Adult__Initialization = Value(s) of ../Adult Initialization
BRANCH = Value(s) of ../BRANCH
my_number__of_nodes = Value(s) of ../my number of nodes

Comments:

original: my_number__of_nodes>=4, however, this was allowing for new branches off of adult eves to be considered seedlings, which is incorrect.

You are an adult if you have >=4 nodes, are a branch/have branched, or were initialized as an adult. One except is seedlings with 3 nodes that sexually reproduce, while it is possible they are not technically considered adults in the model yet (just shy of adulthood).

Submodel Ramet/Node births and lengths :

Submodel "Ramet/Node births and lengths" is a population submodel.

Comments:

This population submodel was created so that I could calculate a length to be associated with each node. It accomplishes this, but processing the values of the population list becomes tricky, so there are many variables outside of the model used to transfer information back to the other variables in the shoot model.

Compartment NodeX :

Initial value = if index(1)==2 then 17 elseif index(1)==3 then 34 elseif index(1)==4 then 51 else X (real)

Where:

X = Value(s) of ../X

Comments:

Takes location of X when node is born to initialize state variable of a node instance in this submodel. These coordinates then mark the location of the node in 2-D space. Units are meters.

Compartment NodeY :

Initial value = if index(1)<5 then 0 else Y (real)

Where:

Y = Value(s) of ../Y

Comments:

Takes location of X when node is born to initialize state variable of a node instance in this submodel. These coordinates then mark the location of the node in 2-D space. Units are meters.

Compartment my internode length :
Initial value = if index(1)==1 then 0 else 317.28*Growing_R___R (real)
Where:
Growing_R___R = Value(s) of ../GROWING INTERNODE

Comments:
This sets the size of the internode associated with this node. The first node (i.e. oldest) of the ramet is given a value of 0, so that an internode length between nodes 0 and 1 is associated with node 1, between nodes 1 and 2 is associated with node 2, and so on. Units are mm.

Creation Initial # Nodes : Initialization
Initial # Nodes = if channel_is(Adult__Initialization) then 4 else 0 (real)
Where:
Adult__Initialization = Value(s) of ../Adult Initialization

Comments:
How many nodes to start the model off with.

Immigration Add a node! : Immigration
Add a node! = if delay(OLDESTLEAF,1)>OLDESTLEAF then 1 else 0 (real)
Where:
OLDESTLEAF = Value(s) of ../OLDESTLEAF

Comments:
A new node is "born" each time that an old leaf is sloughed off. In actuality, this is an immigration process in population terms, rather than a reproductive term. To accomplish this, the "birth" immigration process checks to see if the tank for "Oldest Leaf Mass" was larger on the last time step than on the current time step. If it's lower, then that tank has been emptied and a new node is born! Units are nodes/day.

Couldn't the mass of the leaf decrease during hot temperatures and thus signify a node birth incorrectly?*

Variable index :
index = index(1) (int)

Comments:

instance of a particular node in the node population submodel

Variable node birthday :
node birthday = if channel_is(Initial___Nodes) then 0 else init_time(1) (real)
Where:
Initial___Nodes = Value(s) of Initial # Nodes

Comments:
Time step when a node was created.

Variable node age :
node age = time(1)-init_time(1) (real)

Submodel Ramet/Specific Growth Rate :

Submodel Ramet/Specific Growth Rate/Light :

Submodel Ramet/Specific Growth Rate/Light/Light_Seedling :

Variable Iz_SA :
Iz_SA = table(int(time())) (real)

Comments:
Light at depth after attenuation through water column

Surface_PAR_S*exp((-k_SA*water_depth_SA))

Variable Percent Irradiance_S :
Percent Irradiance_S = Iz_SA/(Surface_PAR_S) (real)
Where:
Iz_SA = Value(s) of Iz_SA
Surface_PAR_S = Value(s) of Surface PAR_S

Comments:
Percent of surface irradiance reaching leaf of eelgrass. No leafIZ like adult model.

Variable Surface PAR_S : Irradiance values
Surface PAR_S = table(int(time())) (real)
Minimum = 1, Maximum = 64
Comments:

Taken from Taskinas Creek Dataset from M.Brush. (Not for all)

Submodel Ramet/Specific Growth Rate/Light/Light_Adult :

Variable Izleaf_A :
Izleaf_A = if time()<=1 or time()==init_time() then Iz_SA else Iz_SA*exp(-1*kcanopy* 0.005 *Canopy_Height__for_Light) (real)
Where:
Iz_SA = Value(s) of ../Light_Seedling/Iz_SA
kcanopy = Value(s) of kcanopy_A
Canopy_Height__for_Light = Value(s) of ../../../Canopy Height for Light

Comments:
Light reaching eelgrass leaf after attenuation of downwelling irradiance by shading of leaves. .005 conversion from cm to m and using only half the length of the leaf. Units are uMol/m2/day.

Variable Percent Irradiance_A :
Percent Irradiance_A = if time()<=1 or Square_Meter__Density<=220 then Iz_SA/Surface_PAR_S else Izleaf_A/Surface_PAR_S+.05 (real)
Where:
Surface_PAR_S = Value(s) of ../Light_Seedling/Surface PAR_S
Iz_SA = Value(s) of ../Light_Seedling/Iz_SA
Square_Meter__Density = Value(s) of ../../../Square Meter Density
Izleaf_A = Value(s) of Izleaf_A

Comments:
Percent of surface irradiance reaching leaf of eelgrass.

Variable kcanopy_A : Light attenuation through leaf canopy
kcanopy_A = if time()<=1 then 0 else 2.09+0.00018*Square_Meter__Density*0.01*Canopy_Height__for_Light (real)
Where:
Square_Meter__Density = Value(s) of ../../../Square Meter Density
Canopy_Height__for_Light = Value(s) of ../../../Canopy Height for Light

Comments:
From Short 1980 taking effect of local shoot density and leaf length into account. Units are m-1.

Submodel Ramet/Specific Growth Rate/Grass Machine :

Submodel Ramet/Specific Growth Rate/Grass Machine/SGR_Seedling :

Variable F(I,T)_S :
F(I,T)_S = if Temperature_SA<= 25 then ((((0.97*Temperature_SA-0.75)*tanh((Iz_SA/(((4.5978*Temperature_SA)+3.3473))))+((-0.0949*Temperature_SA)-1.0503))/20.02) else ((((-2.8*Temperature_SA)+94.3)*tanh((Iz_SA/(((4.5978*Temperature_SA)+3.3473))))+((-0.0949*Temperature_SA)-1.0503))/20.02)) (real)
Where:
Temperature_SA = Value(s) of ../../Temperature/Temperature_SA
Iz_SA = Value(s) of ../../Light/Light_Seedling/Iz_SA

Comments:
Abe et al. 2008 formulation, uses a Jassby & Platt parameterization

Variable SGR_S : Simulated Specific Growth Rate
SGR_S = umax_S*F_I_T__S*sediment_limitation_S+.006 (real)
Where:
umax_S = Value(s) of umax_S
sediment_limitation_S = Value(s) of sediment limitation_S
F_I_T__S = Value(s) of F(I,T)_S

Comments:
Units are gdw/gdw/day

if channel_is(Germinated_Seeds) and year<= 1 then 0 -- the seeds which enter the model as x, y coordinates at the moment of flowering remain dormant in the model as they do not technically germinate until year 2 due to hot summer temperatures(dormany) and ease of model programming this phenomena.

previous code for when I did not have the degree day compartments: if channel_is(Germinated_Seeds) and year<= 1 then 0 else umax_S*F_I_T__S*sediment_limitation_S

umax_S*F_I_T__S*sediment_limitation_S

Variable sediment limitation_S :
sediment limitation_S = if sediments_SA<=1 then 1 elseif sediments_SA>=2000 then 0 else 1.0239*exp(-0.002*sediments_SA) (real)
Where:
sediments_SA = Value(s) of ../../Physical Setting/sediments_SA

Comments:

Taken from Dooley et al 2012. Scaled from 0 to 1 as limitation factor.

Variable umax_S : maximum specific growth rate
umax_S = .03 (real)

Comments:
units: gdw/gdw/day
Taken as approximate average from values reported in literatare and explained on AERS poster. .03

Submodel Ramet/Specific Growth Rate/Grass Machine/SGR_Adult :

Variable SGR_A : specific growth rate
SGR_A = if Set_GMAX_A>=.06 then .06 elseif Set_GMAX_A<=-.06 then -.06 else Set_GMAX_A*sediment__limitation_A (real)
Where:
sediment__limitation_A = Value(s) of sediment limitation_A
Set_GMAX_A = Value(s) of Set GMAX_A

Comments:
Takes GMAX set by temperature and light and reduced by sediment sulfide limitation factor. Units are gdw/gdw/day. I put boundary limits on the adult GMAX since this formulation was causing at times abnormally large spikes and dips in the SGR due to an extreme GMAX.

See Table 1.2 of Harris dissertation for why 0.06 was chosen as a boundary limit.

Variable Set GMAX_A : Setting t and Light determined Gmax
Set GMAX_A = ((umax*alpha*Izleaf/(umax+alpha*Izleaf))+ro) (real)
Where:
alpha = Value(s) of alpha_A
umax = Value(s) of umax_A
ro = Value(s) of ro_A
Izleaf = Value(s) of ../../Light/Light_Adult/Izleaf_A

Comments:
PS equation from Baly and other sources. Uses Relationship reported by Olesen and Sand-Jensen for Growth versus Irradiance. All units for calculating parameters and Gmax converted from ash-free dry weight to dry weight.
units are g/gdw/d

Variable alpha_A : slope of Growth versus Irradiance curves

alpha_A = if Temperature_SA>30 then .005 elseif Temperature_SA<5 then 0.005 else (-0.0003*Temperature_SA^2)+0.007*Temperature_SA-0.0296 (real)

Where:

Temperature_SA = Value(s) of ../../Temperature/Temperature_SA

Comments:

Based on data from Olesen and sand-jensen
equations found in picurves_3.xls
Units are g / gdw/d over mol/m2/d

Variable ro_A : repsiratory cost from Growth versus Irradiance curves.

ro_A = -0.000727*Temperature_SA-.006825 (real)

Where:

Temperature_SA = Value(s) of ../../Temperature/Temperature_SA

Comments:

Taken from Olesen_sandjensen
predicted using equations found in PICURVE_3.xls
Units are grams dw growth/gdw/d

Variable sediment limitation_A : Sediment Sulfide limitation

sediment limitation_A = if sediments_SA<=55.45 then 1 elseif sediments_SA>=2000 then 0 else 13.6*(sediments_SA^-.65) (real)

Where:

sediments_SA = Value(s) of ../../Physical Setting/sediments_SA

Comments:

Taken From Goodman thesis. Converted from units of oxygen/min/dm2. Scaled from 0 to 1 as limitation factor.

Variable umax_A : maxgrowth from Growth versus irradiance curves.

umax_A = if Temperature_SA<=0 then -0.001 else .0177*log(Temperature_SA)+.0011 (real)

Where:

Temperature_SA = Value(s) of ../../Temperature/Temperature_SA

Comments:

predicted from Olesen and sand jensen
Based on equations found in PICURVES_3.xls
units are grams dw growth/gramsdryweight/d

Submodel Ramet/Specific Growth Rate/Temperature :

Compartment FDegreeDays :

Initial value = 0 (real)

Rate of change = + FDays
Comments:
Daily temperature readings can be used to calculate growing degree-days, which is a measure of accumulated heat. Since plant development is temperature-dependent, phenological events of plants can also be used to track degree-days…
Full bloom: date 95% of flowers have opened (e.g. 1 out of 20 buds remains closed).
First bloom: date first flower on the plant opens to reveal pistils and / or stamens.

I simply just need more data on eelgrass flowering first blooms and full bloom to properly parameterize this variable.

I wish I could make the coding here more elegant, however, whatever date it is on the 3rd day above 15 degrees C that is the flowering date.

VA South Bay: Flowers on day 103

Compartment GDegreeDays :
Initial value = 0 (real)

Rate of change = + GDays

Compartment daysabove_20box :
Initial value = 0 (real)

Rate of change = + flow1

Flow FDays :
FDays = if Temperature_SA>15 then 1 else 0 (real)
Where:
Temperature_SA = Value(s) of Temperature_SA

Flow GDays :
GDays = if Temperature_SA<20 and time(0)>244 then 1 else 0 (real)
Where:
Temperature_SA = Value(s) of Temperature_SA

Comments:

244 is Sept 1st. Germination occurs when temps fall below 20 degrees C in the fall.

Flow flow1 :
flow1 = if Temperature_SA>20 then 1 else 0 (real)
Where:
Temperature_SA = Value(s) of Temperature_SA

Variable Daysabove20 :
Daysabove20 = 90 (real)

Comments:

Counts how many days are above 20degrees C in the year (or an average year). This correlates to temperature stress, when respiration increases more than photosynthesis.

Need to figure out how to have this parameter connect directly to the Temperature table to actually count how many days are above 20 to make the model more seamless with less necessary inputs. Right now you have to count.

Around the world, average optimal growth temperatures are ~15-20°C, above which productivity begins to decrease due to the dramatic effect of temperature on respiration (Marsh et al. 1986). The optimum temperature for photosynthesis and growth is commonly based on measurements taken in saturating light conditions.

"So, on the whole I would suggest floral induction of Z. marina is more likely in individuals at a higher metabolic state/most likely size (growing well)."

The model output matches literature valueswhich indicate that as water temperatures increase above 20◦C Z.marina respiration increases at a greater rate than photosynthesiscausing stress and eventually mortality when water temperaturesare greater than 25◦C (Marsh et al., 1986; Nejrup et al., 2008). Jarvis(2014)

GB: ~50 days above 20 per year
VA South Bay: ~126 days above 20 per year
NC ~180 days above 20 per year

Variable Temperature_SA : Temperature degrees celsius
Temperature_SA = table(int(time())) (real)

Comments:

Taken from Short 1980 model. for Ninigret Pond. Units are degrees Celsius.

I should change this to use table data? JF

Submodel Ramet/Specific Growth Rate/clock :

Comments:
Submodel to track time

Variable dayofyear :
dayofyear = days-(year-1)*365 (int)
Where:
days = Value(s) of days
year = Value(s) of year

Variable dayreal :
dayreal = time(1)-(year-1)*365 (real)
Where:
year = Value(s) of year

Variable days :
days = int(time(1)) (int)

Variable month : month #
month = if dayofyear<32 then 1 elseif dayofyear<60 then 2 elseif dayofyear<91 then 3 elseif dayofyear<121 then 4 elseif dayofyear<152 then 5 elseif dayofyear<182 then 6 elseif dayofyear<213 then 7 elseif dayofyear<244 then 8 elseif dayofyear<274 then 9 elseif dayofyear<305 then 10 elseif dayofyear<335 then 11 else 12 (int)
Where:
dayofyear = Value(s) of dayofyear

Comments:
For plotting purposes

Variable year :
year = int((days-1)/365)+1 (int)
Where:
days = Value(s) of days

Submodel Ramet/Specific Growth Rate/Physical Setting :

Variable Nitrogen_SA : g N/m2 as in Roberts et al. paper
Nitrogen_SA = 0.00854 (real)
Minimum = 0.10000000000000001, Maximum = 100
Comments:
Units are g N/m2/day

Variable k_SA :
k_SA = table(int(time())) (real)

Comments:
light attenuation coefficient units m-1. Taken from M.Brush averaged k-value for HIB.

Variable sediments_SA : Sediment sulfide concentration
sediments_SA = 50 (real)
Minimum = 25, Maximum = 2000
Comments:
Units are uMol sulfide. Will eventually take this and use regression to relate to sediment grain size characteristics.

Variable water depth_SA :
water depth_SA = 0.8 (real)
Minimum = 0.10000000000000001, Maximum = 5.0
Comments:
units meters.

Submodel EveStats :

Variable Greatest New Nodes :
Greatest New Nodes = greatest({NewNodes}) (real)
Where:
{NewNodes} = Value(s) of ../Eve Plastochrone Intervals/NewNodes

Variable GreatestPI :
GreatestPI = greatest({Eve_PI_0}) (real)
Where:
{Eve_PI_0} = Value(s) of ../Eve Plastochrone Intervals/Eve PI

Variable Tagged PI :
Tagged PI = if sum({Eve_PI_0})==0 then 0 else sum({Eve_PI_0})/sum({count}) (real)
Where:
{Eve_PI_0} = Value(s) of ../Eve Plastochrone Intervals/Eve PI

{count} = Value(s) of ../Eve Plastochrone Intervals/count

Variable aboveavg :
aboveavg = (sum({above}))/(count({above})) (real)
Where:
{above} = Value(s) of ../Eve Plastochrone Intervals/above

Variable abovestdev :
abovestdev = sqrt(((sum({count})*sum({above}))-
(sum({above})*sum({above})))/(sum({count})*(sum({count})-1))) (real)
Where:
{count} = Value(s) of ../Eve Plastochrone Intervals/count
{above} = Value(s) of ../Eve Plastochrone Intervals/above

Variable avgbelow :
avgbelow = (sum({below}))/(count({below})) (real)
Where:
{below} = Value(s) of ../Eve Plastochrone Intervals/below

Variable belowstdev :
belowstdev = sqrt(((sum({count})*sum({below}))-
(sum({below})*sum({below})))/(sum({count})*(sum({count})-1))) (real)
Where:
{count} = Value(s) of ../Eve Plastochrone Intervals/count
{below} = Value(s) of ../Eve Plastochrone Intervals/below

Variable leastnodes :
leastnodes = least({NewNodes}) (real)
Where:
{NewNodes} = Value(s) of ../Eve Plastochrone Intervals/NewNodes

Variable mean new leaves :
mean new leaves = (sum({Leaves_Shed}))/(sum({count})) (real)
Where:
{count} = Value(s) of ../Eve Plastochrone Intervals/count
{Leaves_Shed} = Value(s) of ../Eve Plastochrone Intervals/Leaves
Shed

Variable mean number of new nodes :

mean number of new nodes =
(sum({NewNodes}))/(sum({count})) (real)
Where:
{count} = Value(s) of ../Eve Plastochrone Intervals/count
{NewNodes} = Value(s) of ../Eve Plastochrone Intervals/NewNodes

Variable stdevnodes :
stdevnodes = sqrt(((sum({count})*sum({Nodessquared}))-(sum({NewNodes})*sum({NewNodes})))/(sum({count})*(sum({count})-1))) (real)
Where:
{count} = Value(s) of ../Eve Plastochrone Intervals/count
{NewNodes} = Value(s) of ../Eve Plastochrone Intervals/NewNodes
{Nodessquared} = Value(s) of ../Eve Plastochrone Intervals/Nodessquared

Submodel Eve Plastochrone Intervals :

Submodel "Eve Plastochrone Intervals" is a "Evetagging" satellite of submodel "Ramet".

Condition I am Eve :
I am Eve = Eve_Status_Evetagging==1 (cond_spec)
Where:
Eve_Status_Evetagging = Value(s) of ../Ramet/Eve Status from submodel "Ramet" in role "Evetagging"
{every_Eve_Status} = Value(s) of ../Ramet/Eve Status

Variable Eve PI :
Eve PI = Plastochrone_Interval_Evetagging (real)
Where:
Plastochrone_Interval_Evetagging = Value(s) of ../Ramet/Plastochrone Interval from submodel "Ramet" in role "Evetagging"
{every_Plastochrone_Interval} = Value(s) of ../Ramet/Plastochrone Interval

Variable Leaves Shed :
Leaves Shed = LeavesProduced_Evetagging (real)
Where:
LeavesProduced_Evetagging = Value(s) of ../Ramet/LeavesProduced from submodel "Ramet" in role "Evetagging"
{every_LeavesProduced} = Value(s) of ../Ramet/LeavesProduced

Variable NewNodes :
NewNodes = (OLD__NODES_Evetagging+NODES_Evetagging) (real)
Where:
OLD__NODES_Evetagging = Value(s) of ../Ramet/OLD NODES from submodel "Ramet" in role "Evetagging"
{every_OLD__NODES} = Value(s) of ../Ramet/OLD NODES
NODES_Evetagging = Value(s) of ../Ramet/NODES from submodel "Ramet" in role "Evetagging"
{every_NODES} = Value(s) of ../Ramet/NODES

Variable Nodessquared :
Nodessquared = NewNodes*NewNodes (real)
Where:
NewNodes = Value(s) of NewNodes

Variable above :
above = OLDESTLEAF_Evetagging+LEAFBUNDLE_Evetagging (real)
Where:
OLDESTLEAF_Evetagging = Value(s) of ../Ramet/OLDESTLEAF from submodel "Ramet" in role "Evetagging"
{every_OLDESTLEAF} = Value(s) of ../Ramet/OLDESTLEAF
LEAFBUNDLE_Evetagging = Value(s) of ../Ramet/LEAFBUNDLE from submodel "Ramet" in role "Evetagging"
{every_LEAFBUNDLE} = Value(s) of ../Ramet/LEAFBUNDLE

Variable below :
below = GROWING_INTERNODE___Evetagging+ROOTS___RHIZOMES_Evetagging (real)
Where:
GROWING_INTERNODE___Evetagging = Value(s) of ../Ramet/GROWING INTERNODE from submodel "Ramet" in role "Evetagging"
{every_GROWING_INTERNODE} = Value(s) of ../Ramet/GROWING INTERNODE
ROOTS___RHIZOMES_Evetagging = Value(s) of ../Ramet/ROOTS & RHIZOMES from submodel "Ramet" in role "Evetagging"{every_ROOTS___RHIZOMES} = Value(s) of ../Ramet/ROOTS & RHIZOMES

Variable count :

count = one_Evetagging (int)
Where:
one_Evetagging = Value(s) of ../Ramet/one from submodel "Ramet" in role "Evetagging"
{every_one} = Value(s) of ../Ramet/one

APPENDIX B – COMPLETE MODEL PROGRAMMING

The complete set of VEMv.2 declarations and programming from SIMILE in C++ is provided. Individual based models greatly rely on their programming software so it is important to share the code in its entirety.

```
#include <support1.cpp>
/* GLOBAL DECLARATIONS */
/* global this
 */
/* global array_9
 */
/* global array_10
 */
/* global array_11
 */
/* global array
 */
/* global array_0
 */
/* global array_1
 */
/* global array_2
 */
/* global array_3
 */
/* global array_4
 */
/* global array_5
 */
/* global array_6
 */
/* global array_7
 */
/* global array_8
 */
char simile_identifier[] = "program='AME',version=5.97,edition=enterprise,date=unused,size=176,";
int phasecount = 2;
double ts[3];
double dts[3];
#include "../Functions/procs.cpp"
/* CONSTANT DECLARATIONS */
double array_9[1461] = { 3.1, 3, 3, 3,  2.9,  2.9,  2.9,  2.8,  2.8,  2.8,  2.7,  2.7,  2.7,  2.6,  2.6,  2.6,  2.5,
2.5,  2.5,  2.4,  2.4,  2.4,  2.3,  2.3,  2.3,  2.2,  2.2,  2.2,  2.1,  2.1,  2.1,  2.1, 2, 2, 2,  1.9,  1.9,  1.9,  1.8,
1.8,  1.8,  1.7,  1.7,  1.7,  1.6,  1.6,  1.6,  1.5,  1.5,  1.5,  1.4,  1.4,  1.4,  1.3,
  1.3,  1.3,  1.2,  1.2,  1.2,  1.1,  1.1,  1.2,  1.2,  1.2,  1.2,  1.2,  1.3,  1.3,  1.3,  1.3,  1.4,  1.4,  1.4,  1.4,
1.4,  1.5,  1.5,  1.5,  1.5,  1.6,  1.6,  1.6,  1.6,  1.6,  1.7,  1.7,  1.7,  1.7,  1.8,  1.8,  1.8,  1.8,  1.8,  1.9,
1.9,  1.9,  1.9, 2, 2, 2, 2, 2,  2.1,  2.1,  2.1,  2.1,  2.2,
  2.2,  2.2,  2.2,  2.2,  2.3,  2.3,  2.3,  2.3,  2.4,  2.4,  2.4,  2.4,  2.4,  2.5,  2.5,  2.5,  2.5,  2.5,  2.5,  2.5,
2.5,  2.5,  2.6,  2.6,  2.6,  2.6,  2.6,  2.6,  2.6,  2.6,  2.6,  2.6,  2.7,  2.7,  2.7,  2.7,  2.7,  2.7,  2.7,  2.7,
2.7,  2.7,  2.8,  2.8,  2.8,  2.7,  2.7,  2.7,  2.7,  2.7,  2.6,
```

2.6, 2.6, 2.6, 2.5, 2.5, 2.5, 2.5, 2.4, 2.4, 2.4, 2.4, 2.3, 2.3, 2.3, 2.3, 2.2, 2.2, 2.2, 2.2, 2.1, 2.1, 2.1, 2.1, 2.4, 2.4, 2.4, 2.5, 2.5, 2.5, 2.6, 2.6, 2.6, 2.6, 2.7, 2.7, 2.7, 2.8, 2.8, 2.8, 2.9, 2.9, 2.9, 3, 3, 3, 3.1, 3.1, 3.1, 3.1, 3.2, 3.2, 3.2,
3.3, 3.3, 3.3, 3.3, 3.3, 3.2, 3.2, 3.2, 3.1, 3.1, 3.1, 3, 3, 3, 2.9, 2.9, 2.9, 2.8, 2.8, 2.8, 2.8, 2.7, 2.7, 2.7, 2.7, 2.6, 2.6, 2.6, 2.5, 2.5, 2.5, 2.4, 2.4, 2.4, 2.3, 2.3, 2.3, 2.3, 2.2, 2.2, 2.2, 2.2, 2.1, 2.1, 2.1, 2.1, 2.1, 2, 2, 2, 2, 1.9, 1.9, 1.9, 1.9,
1.8, 1.8, 1.8, 1.8, 1.7, 1.7, 1.7, 1.7, 1.6, 1.6, 1.6, 1.7, 1.7, 1.7, 1.7, 1.7, 1.7, 1.7, 1.8, 1.8, 1.8, 1.8, 1.8, 1.8, 1.9, 1.9, 1.9, 1.9, 1.9, 1.9, 1.9, 2, 2, 2, 2, 2, 2, 2.1, 2.1, 2.1, 2.1, 2.1, 2.1, 2.1, 2.2, 2.2, 2.2, 2.2, 2.2, 2.2, 2.3, 2.3, 2.3, 2.3,
2.3, 2.3, 2.4, 2.4, 2.4, 2.4, 2.4, 2.4, 2.4, 2.5, 2.5, 2.5, 2.5, 2.5, 2.5, 2.6, 2.6, 2.6, 2.6, 2.6, 2.6, 2.6, 2.7, 2.7, 2.7, 2.7, 2.7, 2.7, 2.8, 2.8, 2.8, 2.8, 2.8, 2.8, 2.8, 2.9, 2.9, 2.9, 2.9, 2.9, 2.9, 3, 3, 3, 3, 3, 3, 3, 3.1, 3, 3, 3, 2.9, 2.9, 2.9, 2.8,
2.8, 2.8, 2.7, 2.7, 2.7, 2.6, 2.6, 2.6, 2.5, 2.5, 2.5, 2.4, 2.4, 2.4, 2.3, 2.3, 2.3, 2.2, 2.2, 2.2, 2.1, 2.1, 2.1, 2.1, 2, 2, 2, 1.9, 1.9, 1.9, 1.8, 1.8, 1.8, 1.7, 1.7, 1.7, 1.6, 1.6, 1.6, 1.5, 1.5, 1.5, 1.4, 1.4, 1.4, 1.3, 1.3, 1.3, 1.2, 1.2, 1.2, 1.1,
1.1, 1.2, 1.2, 1.2, 1.2, 1.2, 1.3, 1.3, 1.3, 1.3, 1.4, 1.4, 1.4, 1.4, 1.4, 1.5, 1.5, 1.5, 1.5, 1.6, 1.6, 1.6, 1.6, 1.6, 1.7, 1.7, 1.7, 1.7, 1.8, 1.8, 1.8, 1.8, 1.8, 1.9, 1.9, 1.9, 1.9, 2, 2, 2, 2, 2, 2.1, 2.1, 2.1, 2.1, 2.2, 2.2, 2.2, 2.2, 2.2, 2.3, 2.3,
2.3, 2.3, 2.4, 2.4, 2.4, 2.4, 2.4, 2.5, 2.5, 2.5, 2.5, 2.5, 2.5, 2.5, 2.5, 2.5, 2.6, 2.6, 2.6, 2.6, 2.6, 2.6, 2.6, 2.6, 2.7, 2.7, 2.7, 2.7, 2.7, 2.7, 2.7, 2.7, 2.7, 2.7, 2.8, 2.8, 2.8, 2.7, 2.7, 2.7, 2.7, 2.7, 2.6, 2.6, 2.6, 2.6, 2.5, 2.5, 2.5,
2.5, 2.4, 2.4, 2.4, 2.4, 2.3, 2.3, 2.3, 2.3, 2.2, 2.2, 2.2, 2.2, 2.1, 2.1, 2.1, 2.1, 2, 2.4, 2.4, 2.5, 2.5, 2.5, 2.6, 2.6, 2.6, 2.6, 2.7, 2.7, 2.7, 2.8, 2.8, 2.8, 2.9, 2.9, 2.9, 3, 3, 3, 3.1, 3.1, 3.1, 3.1, 3.2, 3.2, 3.2, 3.3, 3.3, 3.3, 3.3, 3.3, 3.2, 3.2,
3.2, 3.1, 3.1, 3.1, 3, 3, 3, 2.9, 2.9, 2.9, 2.8, 2.8, 2.8, 2.8, 2.7, 2.7, 2.7, 2.6, 2.6, 2.6, 2.5, 2.5, 2.5, 2.4, 2.4, 2.4, 2.3, 2.3, 2.3, 2.3, 2.2, 2.2, 2.2, 2.2, 2.1, 2.1, 2.1, 2.1, 2.1, 2, 2, 2, 2, 1.9, 1.9, 1.9, 1.9, 1.8, 1.8, 1.8, 1.8, 1.7, 1.7, 1.7,
1.7, 1.6, 1.6, 1.6, 1.7, 1.7, 1.7, 1.7, 1.7, 1.7, 1.7, 1.8, 1.8, 1.8, 1.8, 1.8, 1.8, 1.9, 1.9, 1.9, 1.9, 1.9, 1.9, 1.9, 2, 2, 2, 2, 2, 2, 2.1, 2.1, 2.1, 2.1, 2.1, 2.1, 2.1, 2.2, 2.2, 2.2, 2.2, 2.2, 2.2, 2.3, 2.3, 2.3, 2.3, 2.3, 2.3, 2.4, 2.4, 2.4, 2.4, 2.4,
2.4, 2.4, 2.5, 2.5, 2.5, 2.5, 2.5, 2.5, 2.6, 2.6, 2.6, 2.6, 2.6, 2.6, 2.6, 2.7, 2.7, 2.7, 2.7, 2.7, 2.7, 2.8, 2.8, 2.8, 2.8, 2.8, 2.8, 2.8, 2.9, 2.9, 2.9, 2.9, 2.9, 2.9, 3, 3, 3, 3, 3, 3, 3, 3.1, 3, 3, 3, 2.9, 2.9, 2.9, 2.8, 2.8, 2.8, 2.7, 2.7, 2.7, 2.6, 2.6,
2.6, 2.5, 2.5, 2.5, 2.4, 2.4, 2.4, 2.3, 2.3, 2.3, 2.2, 2.2, 2.2, 2.1, 2.1, 2.1, 2.1, 2, 2, 2, 1.9, 1.9, 1.9, 1.8, 1.8, 1.8, 1.7, 1.7, 1.7, 1.6, 1.6, 1.6, 1.5, 1.5, 1.5, 1.4, 1.4, 1.4, 1.3, 1.3, 1.3, 1.2, 1.2, 1.2, 1.1, 1.1, 1.2, 1.2, 1.2, 1.2, 1.2, 1.3,
1.3, 1.3, 1.3, 1.4, 1.4, 1.4, 1.4, 1.4, 1.5, 1.5, 1.5, 1.5, 1.6, 1.6, 1.6, 1.6, 1.6, 1.7, 1.7, 1.7, 1.7, 1.8, 1.8, 1.8, 1.8, 1.8, 1.9, 1.9, 1.9, 1.9, 2, 2, 2, 2, 2, 2.1, 2.1, 2.1, 2.1, 2.2, 2.2, 2.2, 2.2, 2.2, 2.3, 2.3, 2.3, 2.3, 2.4, 2.4, 2.4, 2.4, 2.4,
2.5, 2.5, 2.5, 2.5, 2.5, 2.5, 2.5, 2.5, 2.5, 2.6, 2.6, 2.6, 2.6, 2.6, 2.6, 2.6, 2.6, 2.6, 2.6, 2.7, 2.7, 2.7, 2.7, 2.7, 2.7, 2.7, 2.7, 2.7, 2.7, 2.8, 2.8, 2.8, 2.7, 2.7, 2.7, 2.7, 2.7, 2.6, 2.6, 2.6, 2.6, 2.5, 2.5, 2.5, 2.5, 2.4, 2.4, 2.4, 2.4, 2.3, 2.3,
2.3, 2.3, 2.2, 2.2, 2.2, 2.2, 2.1, 2.1, 2.1, 2.1, 2.4, 2.4, 2.4, 2.5, 2.5, 2.5, 2.6, 2.6, 2.6, 2.6, 2.7, 2.7, 2.7, 2.8, 2.8, 2.8, 2.9, 2.9, 2.9, 3, 3, 3, 3.1, 3.1, 3.1, 3.1, 3.2, 3.2, 3.2, 3.3, 3.3, 3.3, 3.3, 3.3, 3.2, 3.2, 3.2, 3.1, 3.1, 3.1, 3, 3, 3, 2.9,
2.9, 2.9, 2.8, 2.8, 2.8, 2.8, 2.7, 2.7, 2.7, 2.6, 2.6, 2.6, 2.5, 2.5, 2.5, 2.4, 2.4, 2.4, 2.3, 2.3, 2.3, 2.3, 2.2, 2.2, 2.2, 2.2, 2.1, 2.1, 2.1, 2.1, 2.1, 2, 2, 2, 2, 1.9, 1.9, 1.9, 1.9, 1.8, 1.8, 1.8, 1.8, 1.7, 1.7, 1.7, 1.7, 1.6, 1.6, 1.6, 1.7, 1.7, 1.7,
1.7, 1.7, 1.7, 1.7, 1.8, 1.8, 1.8, 1.8, 1.8, 1.8, 1.9, 1.9, 1.9, 1.9, 1.9, 1.9, 1.9, 2, 2, 2, 2, 2, 2, 2.1, 2.1, 2.1, 2.1, 2.1, 2.1, 2.1, 2.2, 2.2, 2.2, 2.2, 2.2, 2.2, 2.3, 2.3, 2.3, 2.3, 2.3, 2.3, 2.4, 2.4, 2.4, 2.4, 2.4, 2.4, 2.4, 2.5, 2.5, 2.5, 2.5, 2.5,
2.5, 2.6, 2.6, 2.6, 2.6, 2.6, 2.6, 2.6, 2.7, 2.7, 2.7, 2.7, 2.7, 2.7, 2.8, 2.8, 2.8, 2.8, 2.8, 2.8, 2.8, 2.9, 2.9, 2.9, 2.9, 2.9, 2.9, 3, 3, 3, 3, 3, 3, 3.1, 3, 3, 3, 2.9, 2.9, 2.9, 2.8, 2.8, 2.8, 2.7, 2.7, 2.7, 2.6, 2.6, 2.6, 2.5, 2.5, 2.5, 2.4, 2.4, 2.4,

2.3, 2.3, 2.3, 2.2, 2.2, 2.2, 2.1, 2.1, 2.1, 2.1, 2, 2, 2, 1.9, 1.9, 1.9, 1.8, 1.8, 1.8, 1.7, 1.7, 1.7, 1.6, 1.6, 1.6, 1.5, 1.5, 1.5, 1.4, 1.4, 1.4, 1.3, 1.3, 1.3, 1.2, 1.2, 1.2, 1.1, 1.1, 1.2, 1.2, 1.2, 1.2, 1.2, 1.3, 1.3, 1.3, 1.3, 1.4, 1.4, 1.4, 1.4,
1.4, 1.5, 1.5, 1.5, 1.5, 1.6, 1.6, 1.6, 1.6, 1.6, 1.7, 1.7, 1.7, 1.7, 1.8, 1.8, 1.8, 1.8, 1.8, 1.9, 1.9, 1.9, 1.9, 2, 2, 2, 2, 2, 2.1, 2.1, 2.1, 2.1, 2.2, 2.2, 2.2, 2.2, 2.2, 2.3, 2.3, 2.3, 2.3, 2.4, 2.4, 2.4, 2.4, 2.4, 2.5, 2.5, 2.5, 2.5, 2.5, 2.5, 2.5,
2.5, 2.5, 2.6, 2.6, 2.6, 2.6, 2.6, 2.6, 2.6, 2.6, 2.6, 2.6, 2.7, 2.7, 2.7, 2.7, 2.7, 2.7, 2.7, 2.7, 2.7, 2.7, 2.8, 2.8, 2.8, 2.7, 2.7, 2.7, 2.7, 2.7, 2.6, 2.6, 2.6, 2.6, 2.5, 2.5, 2.5, 2.5, 2.4, 2.4, 2.4, 2.4, 2.3, 2.3, 2.3, 2.3, 2.2, 2.2, 2.2, 2.2, 2.1,
2.1, 2.1, 2.1, 2, 2.4, 2.4, 2.5, 2.5, 2.5, 2.6, 2.6, 2.6, 2.6, 2.7, 2.7, 2.7, 2.8, 2.8, 2.8, 2.9, 2.9, 2.9, 3, 3, 3, 3.1, 3.1, 3.1, 3.1, 3.2, 3.2, 3.2, 3.3, 3.3, 3.3, 3.3, 3.3, 3.2, 3.2, 3.2, 3.1, 3.1, 3.1, 3, 3, 3, 2.9, 2.9, 2.9, 2.8, 2.8, 2.8, 2.8, 2.7,
2.7, 2.7, 2.6, 2.6, 2.6, 2.5, 2.5, 2.5, 2.4, 2.4, 2.4, 2.3, 2.3, 2.3, 2.3, 2.2, 2.2, 2.2, 2.2, 2.1, 2.1, 2.1, 2.1, 2.1, 2, 2, 2, 2, 1.9, 1.9, 1.9, 1.9, 1.8, 1.8, 1.8, 1.8, 1.7, 1.7, 1.7, 1.7, 1.6, 1.6, 1.6, 1.7, 1.7, 1.7, 1.7, 1.7, 1.7, 1.7, 1.8, 1.8, 1.8,
1.8, 1.8, 1.8, 1.9, 1.9, 1.9, 1.9, 1.9, 1.9, 1.9, 2, 2, 2, 2, 2, 2, 2.1, 2.1, 2.1, 2.1, 2.1, 2.1, 2.1, 2.2, 2.2, 2.2, 2.2, 2.2, 2.2, 2.2, 2.3, 2.3, 2.3, 2.3, 2.3, 2.3, 2.4, 2.4, 2.4, 2.4, 2.4, 2.4, 2.4, 2.5, 2.5, 2.5, 2.5, 2.5, 2.5, 2.6, 2.6, 2.6, 2.6, 2.6, 2.6,
2.6, 2.7, 2.7, 2.7, 2.7, 2.7, 2.7, 2.7, 2.8, 2.8, 2.8, 2.8, 2.8, 2.8, 2.8, 2.9, 2.9, 2.9, 2.9, 2.9, 2.9, 3, 3, 3, 3, 3, 3};
double array_10[1462] = { 5.35, 5.51, 4.91, 4.42, 4.73, 5.22, 5.63, 5.99, 6.83, 7.29, 7.42, 8.04, 8.34, 8.77, 8.53, 8.26, 8.02, 7.13, 6.49, 7.16, 6.98, 5.54, 4.16, 3.58, 1.9, 1.59, 2.02, 2.59, 3.85, 5.81, 7.8, 5.28, 4.18, 4.45, 4.23, 4.31, 5, 5.27, 6.12, 5.5, 4.92, 5.8, 7, 7.19, 6.97,
7.34, 6.77, 4.79, 2.78, 4.45, 5.26, 4.47, 4.15, 5.01, 5.75, 5.5, 5.52, 6.57, 6.79, 6.32, 5.85, 4.93, 4.6, 5.01, 5.23, 4.97, 4.81, 5.22, 6.12, 6.68, 6.94, 7.5, 6.95, 7.06, 7.97, 6.91, 6.22, 7.44, 8.04, 6.99, 6.03, 7.49, 6.75, 6.07, 6.62, 7.4, 7.63, 7.82, 9.17,
9.16, 9.79, 9.34, 9.71, 9.29, 9.35, 9.4, 10.42, 11.94, 13.54, 15.38, 16.25, 15.44, 15.81, 14.34, 14.35, 15.01, 16.35, 17.42, 17.57, 16.16, 14.61, 12.93, 11.83, 12.66, 14.01, 14.93, 15.66, 15.77, 15.15, 14.89, 14.36, 14.33, 14.25, 13.45, 13.19, 13.99, 15.44, 16.18,
17.52, 19.22, 19.22, 18.9, 17.37, 16.72, 17.78, 20.53, 21.09, 19.38, 19.91, 20.86, 21.38, 21.09, 19.15, 17.88, 16.17, 17.08, 18.13, 18.83, 19.95, 21.38, 22.27, 22.68, 22.72, 21.66, 21.16, 21.89, 21.8, 22.19, 21.42, 22.87, 23.41, 22.63, 23.24, 25.06, 22.2, 22.6, 23.66,
23.46, 22.95, 21.87, 22.08, 22.23, 23.04, 24.66, 25.53, 26.6, 26.96, 26.38, 26.4, 25.19, 23.78, 23.11, 22.78, 23.85, 24.57, 25.39, 26.04, 25.9, 26.35, 24.54, 24.71, 24.06, 22.66, 23.32, 24.41, 26.15, 26.84, 27.72, 28.67, 28.9, 28.94, 26.86, 26.43, 26.02, 26.44, 24.11,
23.91, 24.92, 25.27, 25.53, 25.85, 25.89, 25.08, 25.55, 26.81, 26.01, 25.34, 24.36, 23.91, 25.14, 26.35, 27.13, 27.15, 27.87, 28.47, 26.57, 24.29, 23.5, 23.28, 23.06, 22.94, 23.83, 25.08, 25.97, 26.21, 24.91, 24.26, 24.07, 24.25, 25.3, 25.2, 25.17, 25.92, 26.47,
26.56, 26.8, 26.27, 26.46, 25.36, 24.44, 24.09, 24.56, 24.81, 25.49, 26.1, 25.84, 24.17, 23.72, 23.21, 21.88, 20.84, 21.28, 21.81, 22.27, 22.56, 20.99, 20.44, 20.62, 20.88, 20.42, 19.95, 19.71, 19.42, 20.2, 21.47, 22.26, 22.65, 23.19, 23.52, 23.28, 20.91, 19.09,
18.71, 19.48, 20.13, 20.1, 19.63, 19.11, 19.53, 20.34, 20.04, 19.36, 18.65, 18.18, 17.66, 17.34, 15.03, 13.54, 13.04, 12.55, 13.62, 14.48, 15.37, 16.3, 16.95, 17.44, 15.89, 13.9, 13.24, 13.74, 14.58, 12.67, 11.86, 11.38, 11.26, 10.83, 9.32, 8.27, 8.42, 9.37, 10.21,
11.82, 10.97, 9.76, 9.91, 10.58, 11.02, 7.86, 5.76, 6.37, 8.09, 5.97, 5.9, 6.08, 6.67, 7.11, 7.37, 8.05, 8.78, 10.06, 9.36, 7.94, 7.41, 7.15, 6.29, 5.72, 4.96, 5.66, 6.86, 6.63, 6.84, 6.65, 6.22, 6.71, 7.97, 10.44, 11.27, 8.98, 7.08, 6.14, 5.92, 6.11, 7.37, 7.7,
6.99, 6.37, 6.62, 5.4, 3.66, 4.55, 5.93, 1.56, 0.08, 1.69, 3.04, 5.65, 7.03, 6.86, 8.77, 8.53, 8.26, 8.02, 7.13, 6.49, 7.16, 6.98, 5.54, 4.16, 3.58, 1.9, 1.59, 2.02, 2.59, 3.85, 5.81, 7.8, 5.28, 4.18, 4.45, 4.23, 4.31, 5, 5.27, 6.12, 5.5, 4.92, 5.8, 7, 7.19,
6.97, 7.34, 6.77, 4.79, 2.78, 4.45, 5.26, 4.47, 4.15, 5.01, 5.75, 5.5, 5.52, 6.57, 6.79, 6.32, 5.85, 4.93, 4.6, 5.01, 5.23, 4.97, 4.81, 5.22, 6.12, 6.68, 6.94, 7.5, 6.95, 7.06, 7.97, 6.91, 6.22, 7.44, 8.04, 6.99, 6.03, 7.49, 6.75, 6.07, 6.62, 7.4, 7.63, 7.82,

9.17, 9.16, 9.79, 9.34, 9.71, 9.29, 9.35, 9.4, 10.42, 11.94, 13.54, 15.38, 16.25, 15.44, 15.81, 14.34, 14.35, 15.01, 16.35, 17.42, 17.57, 16.16, 14.61, 12.93, 11.83, 12.66, 14.01, 14.93, 15.66, 15.77, 15.15, 14.89, 14.36, 14.33, 14.25, 13.45, 13.19, 13.99, 15.44, 16.18, 17.52, 19.22, 19.22, 18.9, 17.37, 16.72, 17.78, 20.53, 21.09, 19.38, 19.91, 20.86, 21.38, 21.09, 19.15, 17.88, 16.17, 17.08, 18.13, 18.83, 19.95, 21.38, 22.27, 22.68, 22.72, 21.66, 21.16, 21.89, 21.8, 22.19, 21.42, 22.87, 23.41, 22.63, 23.24, 25.06, 22.2, 22.6, 23.66, 23.46, 22.95, 21.87, 22.08, 22.23, 23.04, 24.66, 25.53, 26.6, 26.96, 26.38, 26.4, 25.19, 23.78, 23.11, 22.78, 23.85, 24.57, 25.39, 26.04, 25.9, 26.35, 24.54, 24.71, 24.06, 22.66, 23.32, 24.41, 26.15, 26.84, 27.72, 28.67, 28.9, 28.94, 26.86, 26.43, 26.02, 26.44, 24.11, 23.91, 24.92, 25.27, 25.53, 25.85, 25.89, 25.08, 25.55, 26.81, 26.01, 25.34, 24.36, 23.91, 25.14, 26.35, 27.13, 27.15, 27.87, 28.47, 26.57, 24.29, 23.5, 23.28, 23.06, 22.94, 23.83, 25.08, 25.97, 26.21, 24.91, 24.26, 24.07, 24.25, 25.3, 25.2, 25.17, 25.92, 26.47, 26.56, 26.8, 26.27, 26.46, 25.36, 24.44, 24.09, 24.56, 24.81, 25.49, 26.1, 25.84, 24.17, 23.72, 23.21, 21.88, 20.84, 21.28, 21.81, 22.27, 22.56, 20.99, 20.44, 20.62, 20.88, 20.42, 19.95, 19.71, 19.42, 20.2, 21.47, 22.26, 22.65, 23.19, 23.52, 23.28, 20.91, 19.09, 18.71, 19.48, 20.13, 20.1, 19.63, 19.11, 19.53, 20.34, 20.04, 19.36, 18.65, 18.18, 17.66, 17.34, 15.03, 13.54, 13.04, 12.55, 13.62, 14.48, 15.37, 16.3, 16.95, 17.44, 15.89, 13.9, 13.24, 13.74, 14.58, 12.67, 11.86, 11.38, 11.26, 10.83, 9.32, 8.27, 8.42, 9.37, 10.21, 11.82, 10.97, 9.76, 9.91, 10.58, 11.02, 7.86, 5.76, 6.37, 8.09, 5.97, 5.9, 6.08, 6.67, 7.11, 7.37, 8.05, 8.78, 10.06, 9.36, 7.94, 7.41, 7.15, 6.29, 5.72, 4.96, 5.66, 6.86, 6.63, 6.84, 6.65, 6.22, 6.71, 7.97, 10.44, 11.27, 8.98, 7.08, 6.14, 5.92, 6.11, 7.37, 7.7, 6.99, 6.37, 5.35, 5.51, 4.91, 4.42, 4.73, 5.22, 5.63, 5.99, 6.83, 7.29, 7.42, 8.04, 8.34, 8.77, 8.53, 8.26, 8.02, 7.13, 6.49, 7.16, 6.98, 5.54, 4.16, 3.58, 1.9, 1.59, 2.02, 2.59, 3.85, 5.81, 7.8, 5.28, 4.18, 4.45, 4.23, 4.31, 5, 5.27, 6.12, 5.5, 4.92, 5.8, 7, 7.19, 6.97, 7.34, 6.77, 4.79, 2.78, 4.45, 5.26, 4.47, 4.15, 5.01, 5.75, 5.5, 5.52, 6.57, 6.79, 6.32, 5.85, 4.93, 4.6, 5.01, 5.23, 4.97, 4.81, 5.22, 6.12, 6.68, 6.94, 7.5, 6.95, 7.06, 7.97, 6.91, 6.22, 7.44, 8.04, 6.99, 6.03, 7.49, 6.75, 6.07, 6.62, 7.4, 7.63, 7.82, 9.17, 9.16, 9.79, 9.34, 9.71, 9.29, 9.35, 9.4, 10.42, 11.94, 13.54, 15.38, 16.25, 15.44, 15.81, 14.34, 14.35, 15.01, 16.35, 17.42, 17.57, 16.16, 14.61, 12.93, 11.83, 12.66, 14.01, 14.93, 15.66, 15.77, 15.15, 14.89, 14.36, 14.33, 14.25, 13.45, 13.19, 13.99, 15.44, 16.18, 17.52, 19.22, 19.22, 18.9, 17.37, 16.72, 17.78, 20.53, 21.09, 19.38, 19.91, 20.86, 21.38, 21.09, 19.15, 17.88, 16.17, 17.08, 18.13, 18.83, 19.95, 21.38, 22.27, 22.68, 22.72, 21.66, 21.16, 21.89, 21.8, 22.19, 21.42, 22.87, 23.41, 22.63, 23.24, 25.06, 22.2, 22.6, 23.66, 23.46, 22.95, 21.87, 22.08, 22.23, 23.04, 24.66, 25.53, 26.6, 26.96, 26.38, 26.4, 25.19, 23.78, 23.11, 22.78, 23.85, 24.57, 25.39, 26.04, 25.9, 26.35, 24.54, 24.71, 24.06, 22.66, 23.32, 24.41, 26.15, 26.84, 27.72, 28.67, 28.9, 28.94, 26.86, 26.43, 26.02, 26.44, 24.11, 23.91, 24.92, 25.27, 25.53, 25.85, 25.89, 25.08, 25.55, 26.81, 26.01, 25.34, 24.36, 23.91, 25.14, 26.35, 27.13, 27.15, 27.87, 28.47, 26.57, 24.29, 23.5, 23.28, 23.06, 22.94, 23.83, 25.08, 25.97, 26.21, 24.91, 24.26, 24.07, 24.25, 25.3, 25.2, 25.17, 25.92, 26.47, 26.56, 26.8, 26.27, 26.46, 25.36, 24.44, 24.09, 24.56, 24.81, 25.49, 26.1, 25.84, 24.17, 23.72, 23.21, 21.88, 20.84, 21.28, 21.81, 22.27, 22.56, 20.99, 20.44, 20.62, 20.88, 20.42, 19.95, 19.71, 19.42, 20.2, 21.47, 22.26, 22.65, 23.19, 23.52, 23.28, 20.91, 19.09, 18.71, 19.48, 20.13, 20.1, 19.63, 19.11, 19.53, 20.34, 20.04, 19.36, 18.65, 18.18, 17.66, 17.34, 15.03, 13.54, 13.04, 12.55, 13.62, 14.48, 15.37, 16.3, 16.95, 17.44, 15.89, 13.9, 13.24, 13.74, 14.58, 12.67, 11.86, 11.38, 11.26, 10.83, 9.32, 8.27, 8.42, 9.37, 10.21, 11.82, 10.97, 9.76, 9.91, 10.58, 11.02, 7.86, 5.76, 6.37, 8.09, 5.97, 5.9, 6.08, 6.67, 7.11, 7.37, 8.05, 8.78, 10.06, 9.36, 7.94, 7.41, 7.15, 6.29, 5.72, 4.96, 5.66, 6.86, 6.63, 6.84, 6.65, 6.22, 6.71, 7.97, 10.44, 11.27, 8.98, 7.08, 6.14, 5.92, 6.11, 7.37, 7.7, 6.99, 6.37, 6.62, 5.4, 3.66, 4.55, 5.93, 1.56, 0.08, 1.69, 3.04, 5.65, 7.03, 6.86, 8.77, 8.53, 8.26, 8.02, 7.13, 6.49, 7.16, 6.98, 5.54, 4.16, 3.58, 1.9, 1.59, 2.02, 2.59, 3.85, 5.81, 7.8, 5.28, 4.18, 4.45, 4.23, 4.31, 5, 5.27, 6.12, 5.5, 4.92, 5.8, 7, 7.19, 6.97, 7.34, 6.77, 4.79, 2.78, 4.45, 5.26, 4.47, 4.15, 5.01, 5.75, 5.5, 5.52, 6.57, 6.79, 6.32, 5.85, 4.93, 4.6, 5.01, 5.23, 4.97, 4.81, 5.22, 6.12, 6.68, 6.94, 7.5, 6.95, 7.06, 7.97, 6.91, 6.22, 7.44, 8.04, 6.99, 6.03, 7.49, 6.75,

6.07, 6.62, 7.4, 7.63, 7.82, 9.17, 9.16, 9.79, 9.34, 9.71, 9.29, 9.35, 9.4, 10.42, 11.94, 13.54,
15.38, 16.25, 15.44, 15.81, 14.34, 14.35, 15.01, 16.35, 17.42, 17.57, 16.16, 14.61, 12.93,
11.83, 12.66, 14.01, 14.93, 15.66, 15.77, 15.15, 14.89, 14.36, 14.33, 14.25,
13.45, 13.19, 13.99, 15.44, 16.18, 17.52, 19.22, 19.22, 18.9, 17.37, 16.72, 17.78, 20.53,
21.09, 19.38, 19.91, 20.86, 21.38, 21.09, 19.15, 17.88, 16.17, 17.08, 18.13, 18.83, 19.95,
21.38, 22.27, 22.68, 22.72, 21.66, 21.16, 21.89, 21.8, 22.19, 21.42, 22.87, 23.41,
22.63, 23.24, 25.06, 22.2, 22.6, 23.66, 23.46, 22.95, 21.87, 22.08, 22.23, 23.04, 24.66, 25.53,
26.6, 26.96, 26.38, 26.4, 25.19, 23.78, 23.11, 22.78, 23.85, 24.57, 25.39, 26.04, 25.9, 26.35,
24.54, 24.71, 24.06, 22.66, 23.32, 24.41, 26.15, 26.84, 27.72, 28.67, 28.9,
28.94, 26.86, 26.43, 26.02, 26.44, 24.11, 23.91, 24.92, 25.27, 25.53, 25.85, 25.89, 25.08,
25.55, 26.81, 26.01, 25.34, 24.36, 23.91, 25.14, 26.35, 27.13, 27.15, 27.87, 28.47, 26.57,
24.29, 23.5, 23.28, 23.06, 22.94, 23.83, 25.08, 25.97, 26.21, 24.91, 24.26, 24.07,
24.25, 25.3, 25.2, 25.17, 25.92, 26.47, 26.56, 26.8, 26.27, 26.46, 25.36, 24.44, 24.09, 24.56,
24.81, 25.49, 26.1, 25.84, 24.17, 23.72, 23.21, 21.88, 20.84, 21.28, 21.81, 22.27, 22.56, 20.99,
20.44, 20.62, 20.88, 20.42, 19.95, 19.71, 19.42, 20.2, 21.47, 22.26, 22.65,
23.19, 23.52, 23.28, 20.91, 19.09, 18.71, 19.48, 20.13, 20.1, 19.63, 19.11, 19.53, 20.34,
20.04, 19.36, 18.65, 18.18, 17.66, 17.34, 15.03, 13.54, 13.04, 12.55, 13.62, 14.48, 15.37, 16.3,
16.95, 17.44, 15.89, 13.9, 13.24, 13.74, 14.58, 12.67, 11.86, 11.38, 11.26,
10.83, 9.32, 8.27, 8.42, 9.37, 10.21, 11.82, 10.97, 9.76, 9.91, 10.58, 11.02, 7.86, 5.76, 6.37,
8.09, 5.97, 5.9, 6.08, 6.67, 7.11, 7.37, 8.05, 8.78, 10.06, 9.36, 7.94, 7.41, 7.15, 6.29, 5.72,
4.96, 5.66, 6.86, 6.63, 6.84, 6.65, 6.22, 6.71, 7.97, 10.44, 11.27,
8.98, 7.08, 6.14, 5.92, 6.11, 7.37, 7.7, 6.99, 6.37};
double array_11[1461] = { 153.654, 609.322, 609.874, 477.263, 177.386, 602.607, 304.802,
202.382, 475.385, 610.067, 609.962, 311.855, 475.012, 532.216, 435.247, 192.961, 669.957,
108.318, 492.542, 692.03, 167.653, 152.71, 554.195, 609.117, 478.603, 635.811, 746.949,
284.922, 750.093, 750.133, 757.072,
107.417, 208.553, 699.726, 614.803, 730.377, 710.037, 733.854, 771.639, 740.91, 778.346,
787.155, 633.822, 72.724, 238.139, 821.219, 825.214, 582.492, 467.407, 855.863, 736.231,
640.663, 633.687, 879.17, 897.949, 98.424, 362.95, 873.888, 940.395, 447.848, 809.048,
878.951,
824.655, 867.183, 909.149, 275.492, 904.637, 867.153, 842.009, 871.807, 840.34, 718.108,
723.675, 713.143, 96.551, 408.303, 1001.66, 1021.07, 931.982, 216.309, 889.37, 915.327,
769.761, 1032.997, 698.719, 841.614, 982.561, 1066.106, 1117.919, 907.707, 583.654, 904.529,
1027.573,
649.434, 1003.085, 927.638, 662.735, 1067.436, 973.153, 1030.066, 404.393, 943.172,
1097.228, 460.799, 155.767, 666.671, 767.818, 592.461, 442.707, 1151.961, 1147.735,
1152.133, 1147.37, 958.264, 1028.802, 644.204, 419.284, 540.734, 925.019, 1101.386,
1079.686, 905.61, 236.302,
722.521, 198.71, 953.659, 1129.34, 268.127, 661.105, 939.621, 852.932, 847.539, 750.376,
1117.695, 1113.333, 1074.191, 753.225, 275.824, 1084.405, 1112.626, 1099.03, 1003.971,
1149.208, 1166.13, 1153.055, 1028.043, 990.701, 1016.658, 1038.027, 1090.597, 1090.934,
1065.547,
879.919, 192.223, 1060.037, 1085.346, 965.739, 1070.321, 1062.909, 1056.636, 553.412,
880.468, 904.391, 901.77, 585.747, 577.168, 785.65, 1091.391, 1029.541, 970.618, 446.342,
1124.679, 937.715, 1144.004, 685.074, 719.377, 1038.245, 973.431, 976.946, 666.34, 740.128,
1068.468,
1129.533, 1066.506, 926.21, 697.46, 1123.925, 1063.021, 1093.929, 970.357, 910.489,
869.556, 1133.638, 493.428, 1008.239, 1010.553, 844.414, 964.888, 889.008, 882.116, 741.524,
997.3, 879.069, 950.332, 804.352, 952.524, 850.055, 939.533, 900.162, 740.219, 786.235,
902.884,
1041.147, 1116.053, 1076.321, 926.036, 791.382, 939.856, 942.725, 1025.298, 764.594,
669.56, 930.788, 1109.155, 974.629, 933.558, 811.317, 751.493, 924.73, 994.727, 632.073,
394.296, 832.585, 514.416, 460.191, 801.922, 973.379, 656.362, 778.078, 858.955, 915.045,
631.847,

805.775, 1066.135, 943.301, 1039.166, 1014.133, 1023.545, 967.551, 937.285, 976.411, 668.325, 799.35, 778.366, 999.614, 882.574, 593.93, 846.761, 910.432, 750.904, 726.759, 593.513, 252.668, 737.356, 749.562, 869.285, 947.578, 931.425, 877.907, 679.179, 739.097, 902.904,
820.698, 942.49, 775.219, 687.405, 858.958, 364.08, 714.836, 872.582, 772.719, 748.694, 644.891, 829.629, 811.129, 848.475, 832.781, 816.5, 703.43, 757.042, 419.98, 341.276, 777.709, 815.45, 798.104, 697.072, 289.789, 94.995, 91.271, 493.144, 648.515, 743.776, 740.321,
731.545, 627.038, 657.804, 655.541, 734.38, 741.313, 292.638, 689.822, 539.714, 476.268, 309.926, 689.761, 506.097, 382.369, 641.036, 81.132, 680.435, 596.556, 609.796, 185.144, 368.768, 612.402, 513.6, 495.062, 631.76, 164.386, 213.321, 445.394, 591.579, 384.164, 616.981,
576.301, 270.282, 577.937, 534.121, 197.353, 630.321, 288.547, 391.889, 346.292, 560.965, 231.395, 572.355, 220.434, 438.044, 175.936, 320.406, 587.445, 554.601, 301.539, 566.103, 220.731, 268.034, 192.52, 606.547, 506.969, 75.332, 582.947, 342.649, 71.795, 48.52306667,
532.5961838, 585.3088324, 401.7610704, 583.1623414, 598.7400591, 473.2642754, 454.2020681, 599.4973093, 506.9334399, 509.2308116, 342.3852554, 269.749022, 356.1182434, 174.9532573, 399.2074862, 454.612008, 586.4447077, 73.68955131, 545.268518, 75.40617481, 357.0434551, 681.3087957,
239.359377, 599.0475141, 160.8957335, 698.8849707, 551.6709828, 458.7968116, 676.0877551, 274.8077947, 684.4545154, 734.0259335, 207.6724107, 597.8782689, 595.4058297, 614.039031, 570.6839345, 558.4990546, 722.0721396, 525.4907417, 620.9144101, 644.6847908, 755.7822256, 179.2967997,
149.7049484, 778.4512469, 699.9125905, 769.2882372, 387.5036569, 420.4220629, 725.5860002, 548.3870277, 699.408113, 110.1808841, 189.0591887, 618.8590389, 440.3314424, 301.5701595, 501.4681223, 852.3497092, 873.8824075, 840.0686784, 819.6471169, 504.9002332, 891.8488626, 886.3768571,
115.0047828, 362.0929364, 930.650988, 905.246241, 805.5593669, 919.2413236, 911.5036021, 713.7607987, 837.4647266, 521.9765043, 973.5443757, 570.6870834, 627.9068394, 992.9558635, 1005.14625, 756.2047499, 1006.320808, 694.4396616, 1019.783054, 970.6809554, 943.7101306, 863.1080919,
686.6486563, 256.6722715, 411.8483395, 569.3040236, 1004.630676, 210.7281858, 818.3573969, 339.4433982, 141.5484411, 196.342331, 241.2261979, 539.8388784, 941.0002092, 863.6660389, 411.2303829, 508.8826114, 701.9481063, 1030.048127, 1061.163423, 1066.302574, 1059.079305, 871.569366,
223.829253, 231.8539443, 643.5802469, 782.6993095, 1047.702448, 893.8501779, 1017.86357, 173.383553, 231.5923833, 892.1134128, 968.25905, 953.4567901, 931.0665069, 969.5888256, 717.746405, 979.3926004, 934.2345716, 1008.803925, 288.2377172, 562.4082534, 437.4906381, 259.4349161,
224.6479928, 1072.79621, 823.4552876, 842.3962702, 556.7667765, 1080.182744, 685.6182594, 1032.819053, 632.8490114, 751.320027, 1040.844068, 1047.116537, 1053.389005, 1093.504718, 1067.969218, 859.884287, 457.7834781, 1094.644997, 1027.284302, 906.9783553, 857.5737717, 852.6501305,
851.1171205, 838.3097103, 825.5023001, 812.69489, 1021.54793, 1048.526669, 1096.745617, 1067.411414, 1083.217083, 1123.332712, 1045.811264, 889.0414025, 867.6408057, 915.6452816, 1001.781052, 1017.094368, 929.1495711, 977.9559866, 783.5938083, 686.9358448, 940.7758299, 1104.475942,
950.5706826, 1010.859754, 966.5870944, 1005.911973, 976.5479299, 927.7191347, 935.8317792, 1040.154611, 972.0357077, 713.2597092, 414.1229523, 571.3510031, 972.9707344, 719.7091846, 251.304988, 754.5039573, 1016.337199, 1022.816124, 1012.751703, 244.9033683, 978.9600589, 842.0228235,
1035.681944, 1059.258237, 865.0156451, 1033.009387, 973.8855144, 933.6002209, 718.9103626, 888.0894533, 867.7029266, 697.0495122, 691.5074544, 885.4518682, 691.2718572, 806.6132892, 781.6013252, 924.0530094, 923.9212673, 1069.899909, 1022.810025, 731.1443433, 948.3114631, 705.8202309,

992.3671041, 906.7777909, 878.2499901, 935.2780076, 982.3127241, 446.9539347,
933.7392127, 655.3453915, 972.7627379, 831.8753202, 837.2069197, 949.363597, 852.0609213,
1018.705915, 1034.162825, 950.157623, 1005.339481, 724.6404224, 418.8950625, 256.2674863,
211.5399771, 673.5035662,
935.4316901, 656.1472987, 965.8923435, 1073.257844, 1058.414268, 1029.747572,
628.9397127, 175.7172269, 1017.157499, 968.8627311, 562.1693595, 188.6452133,
456.8813499, 628.5458533, 870.8476928, 986.8817974, 766.8621045, 305.8698474,
743.4297095, 958.2665712, 690.0416238, 453.4037506,
790.8763371, 766.7882558, 741.1493533, 673.5756165, 87.04515956, 286.5685002,
741.2008235, 411.6859867, 789.8625968, 391.9057423, 350.646735, 926.1688563, 922.4594281,
858.7036321, 847.5994977, 719.2933733, 873.1718509, 717.0401855, 409.9256182,
635.3917117, 652.3497875, 845.1506955,
783.7205371, 803.7746329, 654.4170209, 766.5233771, 83.16750386, 230.6414219,
784.4595247, 774.2972372, 748.2756955, 765.2289413, 498.1477009, 276.4031105,
268.0134274, 727.9342156, 423.44716, 609.4522798, 629.1392968, 699.5918663, 696.0683926,
672.4594281, 395.0454697, 362.2407674,
150.2495287, 195.3615393, 197.5989797, 654.3196185, 196.7006765, 709.7149828,
718.3154042, 666.6823777, 402.8085838, 153.8510591, 137.1659088, 379.8159033,
637.9782633, 524.9889098, 524.7144283, 669.5935455, 543.0852834, 415.2961074,
645.3670844, 635.1585893, 599.1377398, 559.0357103,
613.2832428, 608.2011756, 419.5935455, 378.2383276, 79.11167794, 495.9215925,
304.3913538, 611.5472127, 523.7258248, 426.1319681, 423.3447099, 526.4220705,
443.0858931, 400.5403868, 324.294653, 59.80944255, 503.5807736, 562.1587031, 496.3680319,
396.5642776, 90.13651877, 186.20876,
191.20876, 112.096132, 143.4926052, 319.1780432, 568.2423208, 559.8634812, 273.2992036,
578.9562002, 340.8617747, 270.0142207, 476.1746303, 531.0409556, 400.463595, 400.463595,
400.463595, 153.654, 609.322, 609.874, 477.263, 177.386, 602.607, 304.802, 202.382, 475.385,
610.067,
609.962, 311.855, 475.012, 532.216, 435.247, 192.961, 669.957, 108.318, 492.542, 692.03,
167.653, 152.71, 554.195, 609.117, 478.603, 635.811, 746.949, 284.922, 750.093, 750.133,
757.072, 107.417, 208.553, 699.726, 614.803, 730.377, 710.037, 733.854, 771.639, 740.91,
778.346,
787.155, 633.822, 72.724, 238.139, 821.219, 825.214, 582.492, 467.407, 855.863, 736.231,
640.663, 633.687, 879.17, 897.949, 98.424, 362.95, 873.888, 940.395, 447.848, 809.048,
878.951, 824.655, 867.183, 909.149, 275.492, 904.637, 867.153, 842.009, 871.807, 840.34,
718.108,
723.675, 713.143, 96.551, 408.303, 1001.66, 1021.07, 931.982, 216.309, 889.37, 915.327,
769.761, 1032.997, 698.719, 841.614, 982.561, 1066.106, 1117.919, 907.707, 583.654, 904.529,
1027.573, 649.434, 1003.085, 927.638, 662.735, 1067.436, 973.153, 1030.066, 404.393,
943.172,
1097.228, 460.799, 155.767, 666.671, 767.818, 592.461, 442.707, 1151.961, 1147.735,
1152.133, 1147.37, 958.264, 1028.802, 644.204, 419.284, 540.734, 925.019, 1101.386,
1079.686, 905.61, 236.302, 722.521, 198.71, 953.659, 1129.34, 268.127, 661.105, 939.621,
852.932, 847.539,
750.376, 1117.695, 1113.333, 1074.191, 753.225, 275.824, 1084.405, 1112.626, 1099.03,
1003.971, 1149.208, 1166.13, 1153.055, 1028.043, 990.701, 1016.658, 1038.027, 1090.597,
1090.934, 1065.547, 879.919, 192.223, 1060.037, 1085.346, 965.739, 1070.321, 1062.909,
1056.636, 553.412,
880.468, 904.391, 901.77, 585.747, 577.168, 785.65, 1091.391, 1029.541, 970.618, 446.342,
1124.679, 937.715, 1144.004, 685.074, 719.377, 1038.245, 973.431, 976.946, 666.34, 740.128,
1068.468, 1129.533, 1066.506, 926.21, 697.46, 1123.925, 1063.021, 1093.929, 970.357,
910.489,
869.556, 1133.638, 493.428, 1008.239, 1010.553, 844.414, 964.888, 889.008, 882.116,
741.524, 997.3, 879.069, 950.332, 804.352, 952.524, 850.055, 939.533, 900.162, 740.219,

786.235, 902.884, 1041.147, 1116.053, 1076.321, 926.036, 791.382, 939.856, 942.725,
1025.298, 764.594,
669.56, 930.788, 1109.155, 974.629, 933.558, 811.317, 751.493, 924.73, 994.727, 632.073,
394.296, 832.585, 514.416, 460.191, 801.922, 973.379, 656.362, 778.078, 858.955, 915.045,
631.847, 805.775, 1066.135, 943.301, 1039.166, 1014.133, 1023.545, 967.551, 937.285,
976.411,
668.325, 799.35, 778.366, 999.614, 882.574, 593.93, 846.761, 910.432, 750.904, 726.759,
593.513, 252.668, 737.356, 749.562, 869.285, 947.578, 931.425, 877.907, 679.179, 739.097,
902.904, 820.698, 942.49, 775.219, 687.405, 858.958, 364.08, 714.836, 872.582, 772.719,
748.694,
644.891, 829.629, 811.129, 848.475, 832.781, 816.5, 703.43, 757.042, 419.98, 341.276,
777.709, 815.45, 798.104, 697.072, 289.789, 94.995, 91.271, 493.144, 648.515, 743.776,
740.321, 731.545, 627.038, 657.804, 655.541, 734.38, 741.313, 292.638, 689.822, 539.714,
476.268,
309.926, 689.761, 506.097, 382.369, 641.036, 81.132, 680.435, 596.556, 609.796, 185.144,
368.768, 612.402, 513.6, 495.062, 631.76, 164.386, 213.321, 445.394, 591.579, 384.164,
616.981, 576.301, 270.282, 577.937, 534.121, 197.353, 630.321, 288.547, 391.889, 346.292,
560.965,
231.395, 572.355, 220.434, 438.044, 175.936, 320.406, 587.445, 554.601, 301.539, 566.103,
220.731, 268.034, 192.52, 606.547, 506.969, 75.332, 582.947, 342.649, 71.795, 48.52306667,
532.5961838, 585.3088324, 401.7610704, 583.1623414, 598.7400591, 473.2642754,
454.2020681, 599.4973093,
506.9334399, 509.2308116, 342.3852554, 269.749022, 356.1182434, 174.9532573,
399.2074862, 454.612008, 586.4447077, 73.68955131, 545.268518, 75.40617481, 357.0434551,
681.3087957, 239.359377, 599.0475141, 160.8957335, 698.8849707, 551.6709828, 458.7968116,
676.0877551, 274.8077947,
684.4545154, 734.0259335, 207.6724107, 597.8782689, 595.4058297, 614.039031,
570.6839345, 558.4990546, 722.0721396, 525.4907417, 620.9144101, 644.6847908,
755.7822256, 179.2967997, 149.7049484, 778.4512469, 699.9125905, 769.2882372,
387.5036569, 420.4220629, 725.5860002, 548.3870277,
699.408113, 110.1808841, 189.0591887, 618.8590389, 440.3314424, 301.5701595,
501.4681223, 852.3497092, 873.8824075, 840.0686784, 819.6471169, 504.9002332,
891.8488626, 886.3768571, 115.0047828, 362.0929364, 930.650988, 905.246241, 805.5593669,
919.2413236, 911.5036021, 713.7607987,
837.4647266, 521.9765043, 973.5443757, 570.6870834, 627.9068394, 992.9558635,
1005.14625, 756.2047499, 1006.320808, 694.4396616, 1019.783054, 970.6809554, 943.7101306,
863.1080919, 686.6486563, 256.6722715, 411.8483395, 569.3040236, 1004.630676,
210.7281858, 818.3573969, 339.4433982,
141.5484411, 196.342331, 241.2261979, 539.8388784, 941.0002092, 863.6660389,
411.2303829, 508.8826114, 701.9481063, 1030.048127, 1061.163423, 1066.302574,
1059.079305, 871.569366, 223.829253, 231.8539443, 643.5802469, 782.6993095, 1047.702448,
893.8501779, 1017.86357, 173.383553,
231.5923833, 892.1134128, 968.25905, 953.4567901, 931.0665069, 969.5888256, 717.746405,
979.3926004, 934.2345716, 1008.803925, 288.2377172, 562.4082534, 437.4906381,
259.4349161, 224.6479928, 1072.79621, 823.4552876, 842.3962702, 556.7667765, 1080.182744,
685.6182594, 1032.819053,
632.8490114, 751.320027, 1040.844068, 1047.116537, 1053.389005, 1093.504718,
1067.969218, 859.884287, 457.7834781, 1094.644997, 1027.284302, 906.9783553, 857.5737717,
852.6501305, 851.1171205, 838.3097103, 825.5023001, 812.69489, 1021.54793, 1048.526669,
1096.745617, 1067.411414,
1083.217083, 1123.332712, 1045.811264, 889.0414025, 867.6408057, 915.6452816,
1001.781052, 1017.094368, 929.1495711, 977.9559866, 783.5938083, 686.9358448,
940.7758299, 1104.475942, 950.5706826, 1010.859754, 966.5870944, 1005.911973,
976.5479299, 927.7191347, 935.8317792, 1040.154611,

972.0357077, 713.2597092, 414.1229523, 571.3510031, 972.9707344, 719.7091846,
251.304988, 754.5039573, 1016.337199, 1022.816124, 1012.751703, 244.9033683, 978.9600589,
842.0228235, 1035.681944, 1059.258237, 865.0156451, 1033.009387, 973.8855144,
933.6002209, 718.9103626, 888.0894533,
867.7029266, 697.0495122, 691.5074544, 885.4518682, 691.2718572, 806.6132892,
781.6013252, 924.0530094, 923.9212673, 1069.899909, 1022.810025, 731.1443433,
948.3114631, 705.8202309, 992.3671041, 906.7777909, 878.2499901, 935.2780076,
982.3127241, 446.9539347, 933.7392127, 655.3453915,
972.7627379, 831.8753202, 837.2069197, 949.363597, 852.0609213, 1018.705915,
1034.162825, 950.157623, 1005.339481, 724.6404224, 418.8950625, 256.2674863, 211.5399771,
673.5035662, 935.4316901, 656.1472987, 965.8923435, 1073.257844, 1058.414268,
1029.747572, 628.9397127, 175.7172269,
1017.157499, 968.8627311, 562.1693595, 188.6452133, 456.8813499, 628.5458533,
870.8476928, 986.8817974, 766.8621045, 305.8698474, 743.4297095, 958.2665712,
690.0416238, 453.4037506, 790.8763371, 766.7882558, 741.1493533, 673.5756165,
87.04515956, 286.5685002, 741.2008235, 411.6859867,
789.8625968, 391.9057423, 350.646735, 926.1688563, 922.4594281, 858.7036321,
847.5994977, 719.2933733, 873.1718509, 717.0401855, 409.9256182, 635.3917117,
652.3497875, 845.1506955, 783.7205371, 803.7746329, 654.4170209, 766.5233771,
83.16750386, 230.6414219, 784.4595247, 774.2972372,
748.2756955, 765.2289413, 498.1477009, 276.4031105, 268.0134274, 727.9342156, 423.44716,
609.4522798, 629.1392968, 699.5918663, 696.0683926, 672.4594281, 395.0454697,
362.2407674, 150.2495287, 195.3615393, 197.5989797, 654.3196185, 196.7006765,
709.7149828, 718.3154042, 666.6823777,
402.8085838, 153.8510591, 137.1659088, 379.8159033, 637.9782633, 524.9889098,
524.7144283, 669.5935455, 543.0852834, 415.2961074, 645.3670844, 635.1585893,
599.1377398, 559.0357103, 613.2832428, 608.2011756, 419.5935455, 378.2383276,
79.11167794, 495.9215925, 304.3913538, 611.5472127,
523.7258248, 426.1319681, 423.3447099, 526.4220705, 443.0858931, 400.5403868,
324.294653, 59.80944255, 503.5807736, 562.1587031, 496.3680319, 396.5642776, 90.13651877,
186.20876, 191.20876, 112.096132, 143.4926052, 319.1780432, 568.2423208, 559.8634812,
273.2992036, 578.9562002,
340.8617747, 270.0142207, 476.1746303, 531.0409556, 400.463595, 400.463595};
double array[731] = { 1.54, 2.21, 3.41, 4.35, 11.65, 15.43, 5.96, 5.83, 2.24, -1.72, -4.71, 3.51,
5.24, -1.09, -2.54, -5.03, -12.84, -6.29, 0.22, -4.98, -8.86, -5.86, -3.97, -4.24, -8.77, -15.71, -
10.48, -3.13, -6.34, -8.84, -6.08, -3.5, -2.44, -3.78, -8.93, -11.62, -8.68, -10.19, -8.2,
-7.43, -6.49, -6.13, -3.38, -10.7, -8.13, -10.46, -7.98, -4.26, -4.44, -11.61, -4.26, -1.01, -2.69,
-3.43, -6.49, -2.74, -0.86, -1.12, -0.74, -1.79, 1.3, 1.08, 0.97, -1.77, -14.2, -12.4, -11.64, -8.49,
2.49, 4.94, 4.87, 7.69, 13.57, 6.13, -3.9, -2.54, -2.69, -0.29,
1.21, -3.33, 7.68, 8.33, 2.15, 3.16, 3.38, 5.08, 7.19, 3.97, 5.82, 3.59, 3.9, 3.65, 2.9, 1.18,
1.04, -0.26, 0.38, 0.69, 3.86, 3.05, 3.24, 2.19, 4.42, 6.86, 3.69, 6.51, 3.66, 5.31, 7.21, 7.73,
11.92, 12.09, 21.14, 16.43, 9.4, 9.36, 6.74, 8.27, 6.9, 10.02, 12.06,
12.14, 11.33, 11.97, 11.32, 7.68, 10.74, 18.42, 21.04, 20.46, 18.39, 12.69, 11.97, 13.09,
15.33, 10.09, 7.36, 5.56, 9.36, 10.56, 11.54, 11.83, 10.82, 17.46, 22.4, 21.07, 15.1, 18.8, 18.4,
18.45, 15.81, 19.35, 20.05, 12.05, 11.21, 17.79, 14.67, 14.03, 15.25, 13.7,
17.53, 20.57, 19.92, 12.83, 12.84, 14, 17.96, 22.04, 20.06, 17.95, 19.45, 19.77, 17.77, 16.44,
18.21, 22.22, 25.95, 27.59, 26.35, 18.19, 17.87, 16.18, 17.21, 17.24, 18.25, 20.91, 20.76,
20.41, 19.66, 17.21, 17.21, 21.65, 21.4, 20.62, 18.9, 21.6, 20.77, 21.13, 18.73,
18.24, 21.5, 19.68, 19.54, 16.54, 20.9, 23.56, 25.01, 25.75, 22.27, 22.81, 21.98, 23.67, 22.7,
24.6, 27.23, 25.24, 19.53, 18.29, 21.05, 23.22, 18.73, 15.44, 19.07, 22.92, 21.41, 16.69, 19.57,
22.09, 19.35, 16.82, 15.12, 13.88, 13.52, 14.08, 17.43, 25.09, 27.75, 24.34,
18.99, 18.55, 19.74, 22.07, 19.63, 18.34, 15.49, 21.69, 17.92, 13.29, 15.52, 24.46, 25.33,
17.05, 15.22, 15.18, 16.24, 13.78, 15.16, 15.09, 10.24, 9.86, 9.77, 12.19, 16.48, 16.11, 17.78,
18.87, 17.46, 20.14, 24.1, 21.72, 19.8, 16.12, 10.16, 12.25, 11.91, 16.06, 19.35,

16.79, 17.21, 13.57, 12.19, 12.24, 12.73, 13.3, 13.11, 8.54, 8.81, 9.68, 8.71, 7.3, 11.49,
15.88, 16.98, 15.97, 16.85, 19.9, 12.69, 9.14, 8.26, 14.12, 9.66, 3.47, 6.41, 9.69, 12.4, 6.05,
5.71, 6.9, 5.58, 7.38, 3.93, 2.84, 2.32, 1.23, 0.3, 0.57, 5.8, 7.09, 11.68,
3.76, 0.85, 0.14, 0.04, 0.59, 1.49, 5, 0.08, -3.23, 0.79, 4.66, 6.18, 1.56, 3.99, 1.68, -4.5, -
5.95, -1.13, -5.03, -5.05, -6.1, -5.4, -1.27, -2.95, -4.53, -3.31, 1.74, -6.31, -2.47, -7.52, -6.02, -
5.96, -8.21, -6.71, -3.18, -6.76, -3.33, 3.28, 3.68, 0.85, -1.87,
0.45, 0.3, 3.61, 0.31, -0.2, -2.97, -4.58, -14.35, -10.19, -2.24, -0.22, 2.12, 6.47, 9.73, 2.54,
2.25, 2.12, 0.17, -2.37, -4.45, -4.3, -5.97, 2.57, -1.2, -5.23, -8.6, -3.81, -1.06, -4.78, -6.84, -
4.86, -3.22, -2.6, -1.49, 3.31, -3.15, -0.36, 3.4, 0.68, 0.5, 1.41,
1.6, -2.87, -4.04, -3.77, -0.19, -8.7, -7.11, -0.35, -0.66, 0.49, -7.14, -3.21, 8.5, 1.77, -4.02, -
6.72, -5.41, -4.2, -4, -1.41, -1.58, 0.05, -6.74, -9.95, -0.27, -1.64, 0.06, 6.2, 1.87, 0.81, 0.58,
2.41, -0.14, -3.67, -1.54, 0.73, -0.37, 3.12, 1.8, 1.84, 1.18,
0.3, 1.83, 3.39, -0.62, 0.23, -0.47, -1.64, -1, 6.36, 4.93, 1.17, -1.72, 0.3, 0.78, 11.63, 4.53,
4.95, 3.55, 4.69, 3.71, 2.8, 2.56, 4.5, 12.24, 7.01, 6.89, 5.6, 5.61, 5.33, 6.53, 7.09, 11.12,
8.93, 7.82, 9.39, 11.78, 17.94, 16.71, 9.53, 7.69, 8.14, 7.03, 9.56,
6.01, 6.08, 6.64, 7.02, 7.48, 11.09, 12.46, 14.69, 18.04, 10.3, 10.25, 9.37, 9.37, 13.37, 11.36,
14.24, 12.45, 13, 14.09, 11.1, 12.52, 13.23, 12.13, 14.08, 14.31, 14.68, 19.61, 21.06, 12.93,
16.01, 14.05, 17.76, 19.74, 17.64, 19.93, 13.63, 14.04, 12.53, 21.4, 24.69,
23.48, 22.36, 22.14, 19.68, 17.75, 17.81, 13.75, 16.42, 17.72, 17.09, 17.94, 16.39, 18.9,
19.04, 18.92, 20.14, 20.91, 19.14, 21.31, 15.65, 17.3, 23.39, 21.98, 20.42, 22.38, 18.65, 18.42,
21.03, 23.97, 26.55, 26.22, 23.34, 19.9, 20.55, 23.24, 22.38, 21.78, 21.6, 22.61,
21.65, 23.3, 21.31, 20.94, 20.09, 19.03, 20.97, 22.14, 22.29, 20.99, 21.39, 22.69, 21.73,
21.95, 21.93, 20.69, 20.31, 21.4, 19.04, 17.01, 18.65, 18.88, 19.32, 19.03, 18.29, 17.28, 19.95,
18.98, 20.54, 18.87, 21.03, 23.52, 19.4, 16.74, 17.83, 19.71, 19.61, 18.44,
19.98, 16.65, 17.91, 17.36, 18.18, 21.72, 21.01, 19.63, 17.42, 17.38, 21.41, 21.73, 21.43,
20.43, 18.43, 17.66, 15.86, 12.72, 14.49, 19.07, 20.34, 22.4, 13.8, 13.31, 13.47, 8.82, 9.53,
14.32, 12.27, 10.06, 10.9, 11.46, 13.73, 17.27, 17.67, 18.02, 15.25, 15.34, 13.09,
9.92, 9.32, 10, 8.96, 10.24, 10.25, 16.08, 13.37, 10.84, 10.29, 11.25, 13.96, 11.85, 13.56,
9.87, 4.66, 3.75, 6, 8.32, 5.54, 2.3, 4.64, 9.92, 12.83, 9.2, 9.25, 4.61, 2.39, 5.23, 5.94, 1.23,
2.45, 8.56, 10.87, 13.64, 13.67, 13.24, 11.04, 7.13, 4.01, 2.93, 4.05,
11.49, 13.76, 10.99, 3.45, -0.32, -2.98, -3.74, -3.52, -4.7, -4.19, -0.27, 7.88, 2.19, 0.32, 0.84,
0.76, -0.36, 4.71, 4.38, 0.63, 4.18, 0.68, -2.76, -0.65, -9.27, -3.44, 11.11, -0.03, 0.57, -5.11, -
2.87, 10.76, 2.56, -4.61, -3.69, -6.87, -11.35, -9.9, -7.37, -8.51,
1.67, 3.89, -3.33, 2.06, 9.83, 2.47, -0.58, -8};
double array_0[731] = { 38.99966959, 371.5993906, 371.7757953, 184.1153147, 102.2350717,
191.6580392, 388.6423628, 42.76391872, 376.7919489, 310.9417736, 290.0349689,
172.3104246, 87.13824183, 47.56667952, 44.58202544, 156.9034638, 456.1427759,
383.7826973, 240.2632304, 432.9370193, 453.7015621, 285.6874461,
301.0346568, 435.4436735, 516.9398095, 483.585091, 345.1130752, 482.7599721,
498.7843494, 544.0833746, 526.9095214, 341.2574446, 141.4626568, 512.3749676,
605.4936559, 525.6800947, 552.8121185, 618.4203566, 544.561277, 617.5086931, 426.4377798,
637.8863796, 628.4973921, 573.2958791,
176.484297, 748.5990271, 648.0580217, 727.0602597, 484.2911442, 763.7676174,
324.2425924, 740.3883217, 454.6678134, 791.8600414, 838.248807, 819.8693264, 431.2197129,
783.4371879, 731.0423371, 694.60149, 102.8477313, 477.2724888, 549.9199532, 448.9983661,
735.8126096, 712.0645771,
639.11274, 799.5170961, 590.4758586, 644.0222002, 679.5640899, 456.0709628, 546.8436989,
132.230142, 185.3350272, 255.7612688, 635.1117507, 695.0499168, 874.0488825, 911.2102202,
718.1892042, 753.1084079, 764.3632235, 696.276701, 273.266178, 576.6001859, 912.22557,
945.4416813,
632.3584566, 968.1766875, 812.8355841, 175.9925628, 221.052466, 167.8386596,
501.5311329, 836.2160292, 807.7022871, 752.9478415, 858.1880138, 506.8096097,
956.1106917, 141.8818151, 513.7067327, 614.5330062, 115.6909495, 229.823925, 202.4759548,
216.3373735, 743.2836353, 1002.035173,

926.9620589, 872.0212632, 940.691299, 938.8559666, 501.214771, 856.8163899, 151.1364844,
578.7191047, 157.2383688, 381.2052648, 821.1748457, 701.1237721, 931.4340181, 927.45951,
637.7544938, 806.8987846, 948.3804215, 892.4660105, 804.0249692, 1161.201669,
749.3062049, 870.7408957,
979.7424105, 789.1016698, 450.0728335, 233.0949209, 355.8240883, 141.190961,
206.1357025, 231.7655632, 760.9661831, 1035.455286, 814.5532493, 1054.368596,
987.0965101, 752.6974234, 785.5709269, 893.7798654, 965.1931141, 949.6826494,
362.4650631, 658.2344601, 661.3816133, 157.7233042,
132.6206999, 583.5665249, 832.7718081, 749.5783897, 790.6611447, 123.8321767,
909.7100963, 701.5129468, 614.9727632, 284.6448026, 397.1065592, 897.0207447,
905.9323931, 706.3763898, 885.4283262, 793.181479, 357.4957093, 866.5771211, 715.5473472,
758.199015, 962.0047011, 765.5380196,
948.4889187, 884.3929557, 719.418327, 696.4741437, 959.9843295, 687.3285287,
911.6708339, 971.6756753, 689.425896, 543.4441952, 723.2972373, 634.2009242, 390.6688484,
320.4887361, 517.1359146, 619.6314409, 934.983147, 814.9449674, 827.7343482, 595.0538682,
875.8636074, 798.5193901,
164.3602613, 208.9920086, 707.68287, 767.6345173, 976.0508898, 308.0014218, 744.8484458,
904.8011963, 916.12204, 872.1077242, 500.731235, 898.4995955, 543.5163872, 915.6736898,
991.2848342, 922.2753965, 908.7928955, 927.9784375, 979.9904968, 346.9854503,
783.8725259, 521.0592476,
875.1150872, 397.3254031, 897.3461791, 851.0127802, 519.4988858, 752.4600865,
736.2997772, 755.5462708, 748.2767728, 780.9102766, 519.03159, 734.9344606, 673.3001704,
725.127343, 341.1666011, 680.0231354, 691.8152445, 517.1034867, 788.9741775, 790.0530214,
759.7661555, 722.1183641,
400.6127277, 798.6449683, 833.2641307, 803.8865077, 731.9269578, 798.6405036,
481.0719707, 704.455755, 606.7885525, 221.5688901, 116.0974194, 94.04857577, 755.3665506,
772.9350835, 673.9351728, 249.8548978, 688.8940977, 772.2452898, 776.9532994,
918.9927672, 748.1717118, 700.3482454,
461.0344674, 728.105188, 713.2511831, 697.0488436, 629.5963925, 512.6149656,
451.5559425, 667.921243, 555.0852755, 496.9875622, 574.5845771, 391.7114428, 668.880597,
649.3159204, 597, 330.1766169, 159.079602, 573.2114428, 211.4776119, 98.91044776,
277.1791045, 635.7363184, 539.8781095,
552.6517413, 612.4925373, 562.278607, 360.1965174, 158.6840796, 312.2985075,
566.2761194, 555.5273632, 358.4726368, 147.6293532, 381.39801, 445.5373134, 81.23880597,
519.2263682, 490.0348259, 499.3656716, 431.9875622, 185.1214519, 530.8415869,
63.56719969, 500.2711227, 460.3524595,
52.5203342, 420.9622088, 400.4841476, 108.7589221, 448.447961, 483.5279146, 307.5139711,
318.38931, 418.5276379, 127.0154374, 136.5268633, 430.2080452, 438.117634, 309.1130416,
84.6151718, 82.7449787, 99.64034748, 388.2199967, 410.8642727, 371.7008798, 66.11519947,
258.9027832,
356.744868, 87.81884579, 377.5770486, 389.2128208, 178.6416278, 78.60172865,
193.6280754, 306.3687024, 406.5643125, 171.8285379, 281.6899807, 121.2283445,
120.5937488, 259.9256985, 228.5866598, 81.08722848, 287.2867622, 323.2941923, 74.4709807,
374.599492, 379.9024792, 167.4745062,
51.7783085, 146.2096744, 99.69909777, 130.4182304, 350.6111869, 342.8902726,
323.3323856, 78.99835096, 262.1644111, 215.2102051, 266.9033132, 93.2062436, 114.2365971,
225.0826492, 381.5487538, 321.0911529, 304.608151, 161.0807758, 189.9701545, 248.3325068,
192.3330027, 359.3038184,
35.73025144, 317.6607527, 352.4145499, 123.8246368, 406.8198432, 409.7178908,
287.120227, 290.2630999, 310.7686374, 417.0968097, 454.1422852, 314.8452624, 433.9299686,
349.3343068, 475.7637703, 364.188936, 96.39851599, 371.5279997, 390.5275773, 160.0530791,
502.3244623, 84.43504941,
383.5527489, 188.3474259, 294.9694959, 74.95237076, 55.00552999, 110.5629883,
128.9539406, 274.9702094, 231.2704699, 522.3165293, 442.2969068, 72.85953834,

539.5611688, 374.4746513, 562.9269685, 390.6903564, 170.5958115, 439.9168718,
502.8466945, 608.0152699, 169.3496022, 419.4755432,
622.0082771, 587.0919405, 216.6805808, 172.9858361, 530.5954547, 683.8567198,
201.6394643, 607.7169462, 398.0131079, 212.7595432, 153.5150061, 631.3702807,
246.5020327, 107.1604938, 536.8241264, 691.2102054, 689.5051266, 233.3710728,
637.1921053, 443.1536783, 188.7448061, 191.4761755,
738.8110932, 613.5295041, 141.0095507, 150.2067827, 703.933429, 785.6358921,
811.1935402, 721.1301677, 803.1453831, 593.089409, 481.6222013, 99.51775626, 865.8834783,
812.8870355, 260.7542672, 221.6488805, 705.7062147, 718.8156518, 139.8367859, 420.257376,
250.0962544, 582.1301528,
709.6128897, 520.2490061, 700.3912953, 353.3793681, 411.1404059, 432.0820255,
680.085792, 782.7286043, 779.9832601, 800.9374346, 778.5792007, 775.2312199, 766.4657878,
701.1780707, 759.4831555, 742.4251936, 820.5001046, 723.7267211, 861.8957941,
187.1793262, 249.6379996, 133.3040385,
755.9928855, 636.9832235, 269.2536699, 101.1140653, 152.1850659, 789.2712702,
755.1602756, 798.9945327, 587.6703865, 317.6490413, 526.5952666, 826.0148292,
544.0383463, 791.5424655, 834.1578041, 389.9603056, 467.1809467, 575.9549131, 585.625749,
609.1334632, 749.337178, 619.190009,
456.8360545, 678.5013481, 610.6875374, 889.3929748, 832.916417, 634.7232624,
901.3761983, 901.1047034, 794.4577591, 336.150015, 717.2099963, 794.0637822, 681.6728833,
129.1495711, 166.9041402, 143.7803059, 809.5953003, 861.8183514, 759.0619172,
839.4330474, 804.1383812, 834.7127937,
831.6057441, 765.3692652, 214.9906751, 219.5486759, 623.0193957, 720.1697128,
597.9093622, 582.5438269, 813.2189482, 549.3024991, 190.2163372, 632.0589332,
801.1357702, 340.0317046, 569.3640433, 241.2196941, 371.2234241, 727.4785528,
694.8794405, 505.1518498, 745.4076937, 556.3003865,
584.7634824, 769.2508743, 711.4283085, 813.6996135, 749.6723725, 885.8255108,
632.6836002, 740.2171912, 840.6497331, 524.2076201, 790.3920486, 865.4353028,
750.4748758, 583.6683232, 648.3121664, 211.7191239, 282.9468066, 713.0333149,
255.1150377, 271.0491441, 829.3024112, 745.225474,
492.3614946, 716.9464384, 804.8223817, 706.7807841, 570.9221425, 697.1883989,
165.8056508, 353.2095992, 644.7688852, 382.7264846, 105.1582141, 275.6275367,
458.8347716, 794.8989242, 592.6567364, 190.4381921, 409.1756315, 804.774008, 668.4714505,
741.4016629, 554.5139299, 801.1821728,
759.8869055, 481.1541947, 773.3715569, 780.6103952, 748.2622059, 748.8158569,
674.5438783, 474.774402, 726.4392954, 744.0379083, 676.6245025, 646.6308074, 500.8078181,
716.8183788, 812.9682675, 634.8543168, 739.1263483, 762.0373271, 731.7034418,
284.9975384, 711.0638679, 749.9776216,
194.454639, 756.6777067, 669.2163094, 211.6255651, 679.5528801, 140.4489102,
684.5119277, 370.9730117, 581.2782527, 686.8549434, 684.7088574, 646.4530278,
558.6895224, 375.9880052, 597.9277626, 607.3177281, 487.2398514, 112.5251757, 157.071566,
139.98344, 394.729893, 270.8566441,
147.0078246, 450.1400696, 407.5782457, 517.7767581, 517.2913447, 512.7680641,
621.2132921, 585.7563756, 529.7357998, 598.2974304, 591.250483, 571.0780526, 382.7955951,
464.2001546, 536.7054675, 137.809119, 525.5457883, 483.2254637, 443.9601043, 527.9221406,
264.3305641, 116.5233771,
509.9642581, 497.0150696, 242.1053903, 420.4332496, 460.8698802, 100.1931994,
226.1205564, 458.1650889, 453.7698029, 362.8008207, 518.9974493, 210.1558168,
435.0837307, 261.2260175, 97.21914162, 122.8401908, 172.2856826, 300.2079406,
310.9543085, 311.1095708, 402.3427969, 157.9433293,
109.659532, 70.42253521, 220.8744594, 369.9706111, 391.6768326, 409.5763558,
394.4521459, 414.489298, 355.1097926, 403.9065099, 332.3999113, 47.7348342, 279.6107353,
326.4860818, 87.28512809, 362.0660974, 61.21492736, 108.4101251, 305.8959044,
354.5136519, 152.1587031, 345.0483504,

266.1405006, 68.27076223, 369.5847554, 115.8503982, 53.96473265, 47.38054608,
154.8805461, 340.6968146, 264.6956769, 166.0210466, 111.0750853, 76.85153584,
190.1678043, 107.7189989, 182.3407281, 67.02787258, 343.1911263, 351.2599545,
86.95108077, 330.986917, 274.9089875, 93.54095563,
213.2309443, 341.5159272, 299.1211604, 73.81683732};
double array_1[730] = { 3.606666667, 3.750722222, 3.894777778, 4.038833333, 4.182888889,
4.326944444, 4.471, 4.615055556, 4.759111111, 4.903166667, 5.047222222, 5.191277778,
5.335333333, 5.479388889, 5.623444444, 5.7675, 5.911555556, 6.055611111, 6.199666667,
6.343722222, 6.487777778, 6.631833333, 6.775888889,
6.919944444, 7.064, 7.208055556, 7.352111111, 7.496166667, 7.640222222, 7.784277778,
7.928333333, 8.072388889, 8.216444444, 8.3605, 8.504555556, 8.648611111, 8.792666667,
8.936722222, 9.080777778, 9.224833333, 9.368888889, 9.512944444, 9.657, 9.801055556,
9.945111111, 10.08916667,
10.23322222, 10.37727778, 10.52133333, 10.66538889, 10.80944444, 10.9535, 11.09755556,
11.24161111, 11.38566667, 11.52972222, 11.67377778, 11.81783333, 11.96188889, 12.25,
12.37342441, 12.49684882, 12.62027322, 12.74369763, 12.86712204, 12.99054645,
13.11397086, 13.23739526, 13.36081967,
13.48424408, 13.60766849, 13.7310929, 13.8545173, 13.97794171, 14.10136612, 14.22479053,
14.34821494, 14.47163934, 14.59506375, 14.71848816, 14.84191257, 14.96533698,
15.08876138, 15.21218579, 15.3356102, 15.45903461, 15.58245902, 15.70588342, 15.82930783,
15.95273224, 16.07615665,
16.19958106, 16.32300546, 16.44642987, 16.56985428, 16.69327869, 16.8167031, 16.9401275,
17.06355191, 17.18697632, 17.31040073, 17.43382514, 17.55724954, 17.68067395,
17.80409836, 17.92752277, 18.05094718, 18.17437158, 18.29779599, 18.4212204, 18.54464481,
18.66806922, 18.79149362,
18.91491803, 19.03834244, 19.16176685, 19.28519126, 19.40861566, 19.53204007,
19.65546448, 19.77888889, 20.02878136, 20.27867384, 20.52856631, 20.77845878,
21.02835125, 21.27824373, 21.5281362, 21.77802867, 22.02792115, 22.27781362, 22.52770609,
22.77759857, 23.02749104, 23.27738351,
23.52727599, 23.77716846, 24.02706093, 24.27695341, 24.52684588, 24.77673835,
25.02663082, 25.2765233, 25.52641577, 25.77630824, 26.02620072, 26.27609319, 26.52598566,
26.77587814, 27.02577061, 27.27566308, 27.52555556, 27.49922222, 27.47288889,
27.44655556, 27.42022222, 27.39388889,
27.36755556, 27.34122222, 27.31488889, 27.28855556, 27.26222222, 27.23588889,
27.20955556, 27.18322222, 27.15688889, 27.13055556, 27.10422222, 27.07788889,
27.05155556, 27.02522222, 26.99888889, 26.97255556, 26.94622222, 26.91988889,
26.89355556, 26.86722222, 26.84088889, 26.81455556,
26.78822222, 26.76188889, 24.83, 24.97802867, 25.12605735, 25.27408602, 25.4221147,
25.57014337, 25.71817204, 25.86620072, 26.01422939, 26.16225806, 26.31028674,
26.45831541, 26.60634409, 26.75437276, 26.90240143, 27.05043011, 27.19845878,
27.34648746, 27.49451613, 27.6425448, 27.79057348,
27.93860215, 28.08663082, 28.2346595, 28.38268817, 28.53071685, 28.67874552,
28.82677419, 28.97480287, 29.12283154, 29.27086022, 29.41888889, 29.29326165,
29.16763441, 29.04200717, 28.91637993, 28.79075269, 28.66512545, 28.53949821,
28.41387097, 28.28824373, 28.16261649, 28.03698925,
27.91136201, 27.78573477, 27.66010753, 27.53448029, 27.40885305, 27.28322581,
27.15759857, 27.03197133, 26.90634409, 26.78071685, 26.65508961, 26.52946237,
26.40383513, 26.27820789, 26.15258065, 26.02695341, 25.90132616, 25.77569892,
25.65007168, 25.52444444, 25.21118519, 24.89792593,
24.58466667, 24.27140741, 23.95814815, 23.64488889, 23.33162963, 23.01837037,
22.70511111, 22.39185185, 22.07859259, 21.76533333, 21.45207407, 21.13881481,
20.82555556, 20.5122963, 20.19903704, 19.88577778, 19.57251852, 19.25925926, 18.946,
18.63274074, 18.31948148, 18.00622222,
17.69296296, 17.3797037, 17.06644444, 16.75318519, 16.43992593, 16.12666667,
15.99057971, 15.85449275, 15.7184058, 15.58231884, 15.44623188, 15.31014493, 15.17405797,

15.03797101, 14.90188406, 14.7657971, 14.62971014, 14.49362319, 14.35753623, 14.22144928,
14.08536232, 13.94927536,
13.81318841, 13.67710145, 13.54101449, 13.40492754, 13.26884058, 13.13275362,
12.99666667, 12.86057971, 12.72449275, 12.5884058, 12.45231884, 12.31623188, 12.18014493,
12.04405797, 11.90797101, 11.77188406, 11.6357971, 11.49971014, 11.36362319, 11.22753623,
11.09144928, 10.95536232,
10.81927536, 10.68318841, 10.54710145, 10.41101449, 10.27492754, 10.13884058,
10.00275362, 9.866666667, 9.73057971, 9.594492754, 9.458405797, 9.322318841, 9.186231884,
9.050144928, 8.914057971, 8.777971014, 8.641884058, 8.505797101, 8.369710145,
8.233623188, 8.097536232, 7.961449275,
7.825362319, 7.689275362, 7.553188406, 7.417101449, 7.281014493, 7.144927536,
7.00884058, 6.872753623, 6.736666667, 6.60057971, 6.464492754, 6.328405797, 6.192318841,
6.056231884, 5.920144928, 5.784057971, 5.647971014, 5.511884058, 5.375797101,
5.239710145, 5.103623188, 4.967536232,
4.831449275, 4.695362319, 4.559275362, 4.423188406, 4.287101449, 4.151014493,
4.014927536, 3.87884058, 3.742753623, 3.606666667, 3.750722222, 3.894777778, 4.038833333,
4.182888889, 4.326944444, 4.471, 4.615055556, 4.759111111, 4.903166667, 5.047222222,
5.191277778, 5.335333333,
5.479388889, 5.623444444, 5.7675, 5.911555556, 6.055611111, 6.199666667, 6.343722222,
6.487777778, 6.631833333, 6.775888889, 6.919944444, 7.064, 7.208055556, 7.352111111,
7.496166667, 7.640222222, 7.784277778, 7.928333333, 8.072388889, 8.216444444, 8.3605,
8.504555556, 8.648611111,
8.792666667, 8.936722222, 9.080777778, 9.224833333, 9.368888889, 9.512944444, 9.657,
9.801055556, 9.945111111, 10.08916667, 10.23322222, 10.37727778, 10.52133333,
10.66538889, 10.80944444, 10.9535, 11.09755556, 11.24161111, 11.38566667, 11.52972222,
11.67377778, 11.81783333, 11.96188889,
12.25, 12.37342441, 12.49684882, 12.62027322, 12.74369763, 12.86712204, 12.99054645,
13.11397086, 13.23739526, 13.36081967, 13.48424408, 13.60766849, 13.7310929, 13.8545173,
13.97794171, 14.10136612, 14.22479053, 14.34821494, 14.47163934, 14.59506375,
14.71848816, 14.84191257, 14.96533698,
15.08876138, 15.21218579, 15.3356102, 15.45903461, 15.58245902, 15.70588342,
15.82930783, 15.95273224, 16.07615665, 16.19958106, 16.32300546, 16.44642987,
16.56985428, 16.69327869, 16.8167031, 16.9401275, 17.06355191, 17.18697632, 17.31040073,
17.43382514, 17.55724954, 17.68067395,
17.80409836, 17.92752277, 18.05094718, 18.17437158, 18.29779599, 18.4212204,
18.54464481, 18.66806922, 18.79149362, 18.91491803, 19.03834244, 19.16176685,
19.28519126, 19.40861566, 19.53204007, 19.65546448, 19.77888889, 20.02878136,
20.27867384, 20.52856631, 20.77845878, 21.02835125,
21.27824373, 21.5281362, 21.77802867, 22.02792115, 22.27781362, 22.52770609,
22.77759857, 23.02749104, 23.27738351, 23.52727599, 23.77716846, 24.02706093,
24.27695341, 24.52684588, 24.77673835, 25.02663082, 25.2765233, 25.52641577, 25.77630824,
26.02620072, 26.27609319, 26.52598566,
26.77587814, 27.02577061, 27.27566308, 27.52555556, 27.49922222, 27.47288889,
27.44655556, 27.42022222, 27.39388889, 27.36755556, 27.34122222, 27.31488889,
27.28855556, 27.26222222, 27.23588889, 27.20955556, 27.18322222, 27.15688889,
27.13055556, 27.10422222, 27.07788889, 27.05155556,
27.02522222, 26.99888889, 26.97255556, 26.94622222, 26.91988889, 26.89355556,
26.86722222, 26.84088889, 26.81455556, 26.78822222, 26.76188889, 26.73555556,
24.97802867, 25.12605735, 25.27408602, 25.4221147, 25.57014337, 25.71817204, 25.86620072,
26.01422939, 26.16225806, 26.31028674,
26.45831541, 26.60634409, 26.75437276, 26.90240143, 27.05043011, 27.19845878,
27.34648746, 27.49451613, 27.6425448, 27.79057348, 27.93860215, 28.08663082, 28.2346595,
28.38268817, 28.53071685, 28.67874552, 28.82677419, 28.97480287, 29.12283154,
29.27086022, 29.41888889, 29.29326165,

29.16763441, 29.04200717, 28.91637993, 28.79075269, 28.66512545, 28.53949821,
28.41387097, 28.28824373, 28.16261649, 28.03698925, 27.91136201, 27.78573477,
27.66010753, 27.53448029, 27.40885305, 27.28322581, 27.15759857, 27.03197133,
26.90634409, 26.78071685, 26.65508961, 26.52946237,
26.40383513, 26.27820789, 26.15258065, 26.02695341, 25.90132616, 25.77569892,
25.65007168, 25.52444444, 25.21118519, 24.89792593, 24.58466667, 24.27140741,
23.95814815, 23.64488889, 23.33162963, 23.01837037, 22.70511111, 22.39185185,
22.07859259, 21.76533333, 21.45207407, 21.13881481,
20.82555556, 20.5122963, 20.19903704, 19.88577778, 19.57251852, 19.25925926, 18.946,
18.63274074, 18.31948148, 18.00622222, 17.69296296, 17.3797037, 17.06644444, 16.75318519,
16.43992593, 16.12666667, 15.99057971, 15.85449275, 15.7184058, 15.58231884, 15.44623188,
15.31014493, 15.17405797,
15.03797101, 14.90188406, 14.7657971, 14.62971014, 14.49362319, 14.35753623,
14.22144928, 14.08536232, 13.94927536, 13.81318841, 13.67710145, 13.54101449,
13.40492754, 13.26884058, 13.13275362, 12.99666667, 12.86057971, 12.72449275, 12.5884058,
12.45231884, 12.31623188, 12.18014493,
12.04405797, 11.90797101, 11.77188406, 11.6357971, 11.49971014, 11.36362319,
11.22753623, 11.09144928, 10.95536232, 10.81927536, 10.68318841, 10.54710145,
10.41101449, 10.27492754, 10.13884058, 10.00275362, 9.866666667, 9.73057971, 9.594492754,
9.458405797, 9.322318841, 9.186231884,
9.050144928, 8.914057971, 8.777971014, 8.641884058, 8.505797101, 8.369710145,
8.233623188, 8.097536232, 7.961449275, 7.825362319, 7.689275362, 7.553188406,
7.417101449, 7.281014493, 7.144927536, 7.00884058, 6.872753623, 6.736666667, 6.60057971,
6.464492754, 6.328405797, 6.192318841,
6.056231884, 5.920144928, 5.784057971, 5.647971014, 5.511884058, 5.375797101,
5.239710145, 5.103623188, 4.967536232, 4.831449275, 4.695362319, 4.559275362,
4.423188406, 4.287101449, 4.151014493, 4.014927536, 3.87884058, 3.742753623};
double array_2[731] = { 14.0875, 13.20104167, 12.86145833, 13.27222222, 13.10103717,
12.92985212, 12.75866707, 12.58748202, 12.41629697, 12.24511192, 12.07392687,
11.90274182, 11.73155676, 11.56037171, 11.38918666, 11.21800161, 11.04681656,
10.87563151, 10.70444646, 10.53326141, 10.36207636, 10.19089131,
10.01970626, 9.848521205, 9.677336155, 9.506151104, 9.334966053, 9.163781002,
8.992595951, 8.8214109, 8.650225849, 8.479040799, 8.307855748, 8.136670697, 7.965485646,
9.654285714, 6.273958333, 7.503125, 8.079166667, 7.715625, 7.680208333, 7.266666667,
8.384375, 9.995833333, 10.18958333,
8.135416667, 7.398958333, 7.345833333, 7.573958333, 6.557291667, 8.217708333, 9.0625,
10.684375, 9.445833333, 7.938461538, 8.329291311, 8.720121083, 9.110950855, 9.501780627,
9.892610399, 10.28344017, 10.67426994, 11.06509972, 11.45592949, 13.1, 13.06666667,
13.14166667, 11.89270833,
13.11458333, 14.571875, 14.509375, 15.47083333, 16.496875, 17.16666667, 15.84895833,
13.06354167, 12.16979167, 12.803125, 14.34479167, 15.346875, 16.52916667, 17.90833333,
19.54270833, 18.565625, 17.51041667, 18.68645833, 19.16875, 16.58125, 15.46354167,
16.50416667, 18.07291667,
19.33541667, 19.70729167, 20.33645833, 18.55416667, 15.85833333, 14.65833333,
13.79166667, 13.74166667, 14.321875, 15.17604167, 17.184375, 17.04270833, 17.46458333,
18.05625, 14.79270833, 15.52916667, 16.18854167, 16.29375, 16.75902778, 17.22430556,
17.68958333, 18.15486111, 18.62013889,
19.08541667, 19.55069444, 20.01597222, 20.48125, 20.94652778, 21.41180556, 21.87708333,
21.8625, 22.671875, 20.56770833, 20.03541667, 20.41875, 17.28541667, 18.16875,
20.18958333, 21.96666667, 23.334375, 23.84583333, 22.73541667, 20.7625, 21.85208333,
22.1125, 21.571875, 21.48229167,
21.18645833, 22.05416667, 22.640625, 23.51770833, 23.18020833, 22.95473684, 23.54583333,
24.128125, 24.5875, 24.9875, 25.21875, 25.3375, 25.10104167, 25.28020833, 24.33541667,
23.76041667, 24.91666667, 26.11666667, 25.98333333, 26.77708333, 27.17916667,
26.98333333, 26.34270833,

25.15729167, 25.37708333, 25.44166667, 24.38958333, 23.41041667, 24.890625, 25.81041667, 26.96041667, 27.84375, 27.21666667, 26.86770833, 27.103125, 27.46770833, 27.95104167, 27.98020833, 28.33958333, 28.765625, 28.584375, 28.63541667, 27.84166667, 27.70104167, 26.09583333, 26.04895833,
26.728125, 27.61666667, 28.453125, 28.296875, 27.40729167, 27.5625, 28.25520833, 28.51145833, 28.32083333, 28.865625, 27.96041667, 28.19791667, 28.84583333, 29.08125, 29.46145833, 29.56666667, 29.17708333, 27.61145833, 25.89375, 25.80104167, 26.69479167, 26.7125, 27.73020833, 28.58333333,
28.08333333, 27.61875, 28.25208333, 28.32291667, 28.79270833, 29.05104167, 29.10729167, 29.17395833, 29.42291667, 29.31458333, 29.74479167, 30.10833333, 30.4625, 30.13020833, 28.99479167, 28.54166667, 28.39818182, 30.43818182, 29.91770833, 29.64375, 28.95833333, 29.20416667, 28.99375,
29.590625, 29.878125, 30.175, 30.01875, 29.771875, 29.91979167, 29.834375, 29.21041667, 29.03125, 28.83125, 29.00104167, 28.58541667, 27.98645833, 27.075, 27.14479167, 27.42291667, 27.73052632, 28.07395833, 27.728125, 27.12291667, 26.54166667, 27.07395833, 27.97083333, 27.54166667,
27.909375, 27.70520833, 27.39166667, 25.99375, 24.71458333, 24.26458333, 24.215625, 24.04479167, 24.721875, 26.24479167, 26.99166667, 26.76145833, 26.26979167, 26.95, 27.42916667, 27.325, 24.93229167, 23.97916667, 23.91458333, 23.80729167, 24.86041667, 25.32708333, 25.22083333,
25.296875, 25.81666667, 26.35208333, 26.34895833, 26.05416667, 25.09270833, 22.93125, 22.775, 22.92083333, 23.13473684, 23.165625, 24.12083333, 24.896875, 25.44583333, 24.75729167, 23.496875, 23.59375, 24.34479167, 25.21666667, 25.16145833, 24.50104167, 24.03541667, 22.34270833,
19.96875, 19.36875, 19.984375, 20.55729167, 19.396875, 18.096875, 17.53229167, 17.496875, 16.57916667, 15.0875, 14.77291667, 14.784375, 14.52083333, 14.82083333, 15.19368421, 15.74895833, 16.79479167, 16.434375, 13.53645833, 13.67708333, 14.1, 15.38229167, 16.21145833, 16.73958333,
17.49583333, 15.83020833, 14.0125, 13.78958333, 15.32395833, 17.01979167, 14.9, 14.71979167, 14.64583333, 13.11041667, 12.92916667, 14.26770833, 11.5, 11.57604167, 10.73854167, 10.89270833, 12.36458333, 13.90729167, 15.14791667, 16.13958333, 16.68125, 17.08229167, 17.11875, 14.94166667,
15.17083333, 10.81770833, 10.4875, 10.821875, 11.50416667, 12.06770833, 12.59895833, 13.70729167, 13.734375, 12.7375, 11.98854167, 11.93645833, 12.59479167, 14.41770833, 15.03020833, 14.73541667, 13.54166667, 9.320833333, 6.9, 7.236458333, 7.508333333, 9.432291667, 11.46875, 12.815625,
14.12526316, 14.83333333, 15.709375, 15.37604167, 13.321875, 12.434375, 10.9125, 9.4875, 9.959375, 9.81875, 9.573958333, 8.698958333, 7.427083333, 8.38125, 10.45625, 10.56666667, 8.71875, 8.285416667, 8.988541667, 8.922916667, 9.633333333, 12.30416667, 10.13020833, 11.30208333,
11.38125, 11.89583333, 12.9125, 14.60520833, 16.13125, 16.159375, 14.09166667, 13.37291667, 12.875, 11.54270833, 11.753125, 14.30416667, 13.26354167, 12.409375, 12.40625, 11.86, 11.86, 11.86, 11.86, 12.5987013, 12.5987013, 12.5987013, 12.5987013, 12.490625, 13.246875, 12.68333333,
10.39166667, 11.03229167, 12.79270833, 12.57604167, 13.70208333, 15.05833333, 16.31875, 15.403125, 14.59791667, 15.54270833, 13.04375, 13.90736842, 14.53645833, 15.39583333, 15.73229167, 15.73645833, 16.10520833, 15.7875, 14.19791667, 14.309375, 16.08541667, 17.22291667, 15.396875,
16.09791667, 15.596875, 14.390625, 13.74479167, 14.6875, 16.2875, 17.95, 16.30208333, 13.27291667, 14.55208333, 16.70104167, 17.18020833, 14.28541667, 16.11979167, 17.90104167, 18.22916667, 16.65729167, 16.22421053, 17.12708333, 18.48229167, 19.94791667, 20.68645833, 18.86458333,
16.91979167, 15.38125, 14.89895833, 15.77291667, 17.059375, 18.35104167, 19.653125, 19.46666667, 18.584375, 19.46875, 21.10416667, 22.55625, 23.13958333, 22.78125,

21.82291667, 20.57708333, 18.7125, 19.48333333, 20.34791667, 20.734375, 21.340625,
21.20947368, 21.61041667, 22.52708333,
22.375, 23.58541667, 23.33645833, 21.209375, 20.334375, 19.74375, 20.2375, 20.959375,
22.61041667, 22.40625, 21.94166667, 22.71458333, 24.03229167, 24.00104167, 24.025,
23.81770833, 22.71354167, 22.98333333, 23.65208333, 24.29270833, 23.38020833, 22.121875,
23.77291667, 24.96875,
25.38020833, 24.98645833, 25.68947368, 26.41354167, 27.25104167, 27.54375, 27.94375,
28.33645833, 28.515625, 28.53125, 28.55, 28.396875, 28.13645833, 28.43958333, 27.36458333,
27.4375, 27.703125, 25.884375, 26.39895833, 25.77604167, 24.9, 25.82083333, 26.72291667,
27.61979167,
28.59166667, 28.51354167, 28.27291667, 28.171875, 27.63854167, 26.81979167, 26.33541667,
26.48229167, 27.20625, 27.86770833, 27.93333333, 27.94166667, 27.89895833, 27.97291667,
27.50729167, 26.49166667, 26.63333333, 27.01458333, 27.71979167, 28.19375, 27.965625,
28.00416667, 27.92604167,
28.10625, 27.64270833, 27.54583333, 28.609375, 29.83645833, 29.55, 28.46458333,
28.32604167, 28.27708333, 28.18958333, 28.303125, 28.48958333, 28.315625, 28.45208333,
28.39479167, 28.534375, 28.01979167, 28.44583333, 29.14791667, 29.35729167, 28.76458333,
28.22916667, 27.88125,
27.55833333, 26.93541667, 26.65833333, 25.03645833, 24.68854167, 26.25520833,
26.34583333, 26.05208333, 26.46458333, 27.79270833, 28.56145833, 27.13854167,
26.71145833, 26.96041667, 27.71666667, 27.93541667, 27.98958333, 28.19375, 28.29895833,
28.61875, 29.06666667, 29.03541667,
28.27708333, 27.45157895, 27.53684211, 27.97916667, 26.99166667, 26.81041667,
27.97916667, 28.71458333, 28.634375, 27.99895833, 27.40416667, 28.04375, 28.48125,
28.5125, 28.66875, 28.06458333, 26.06354167, 25.60520833, 25.11041667, 23.88020833,
23.975, 23.765625, 23.40625, 22.08854167,
21.19791667, 21.925, 23.40625, 23.45333333, 0, 0, 25.17708333, 23.109375, 22.671875,
22.77083333, 22.87395833, 22.74895833, 22.41875, 21.79166667, 22.175, 22.34166667,
22.06666667, 21.65208333, 21.871875, 22.47083333, 22.70104167, 23.06041667, 23.065625,
21.26458333, 18.93854167,
18.16979167, 18.175, 17.5, 17.08854167, 17.64375, 18.84583333, 19.42708333, 18.68854167,
16.07708333, 14.84791667, 14.928125, 15.22947368, 16.14270833, 16.91145833, 16.80520833,
17.25, 17.521875, 17.471875, 17.67604167, 17.70208333, 16.753125, 16.003125, 15.7375,
15.53229167,
17.18645833, 18.07083333, 19.00833333, 16.71770833, 15.39583333, 13.27263158, 11.109375,
11.36770833, 10.61041667, 9.186458333, 9.691666667, 10.52604167, 11.515625, 11.25520833,
11.459375, 12.196875, 12.6, 13.21770833, 13.41770833, 11.49375, 10.72916667, 10.95104167,
11.63541667,
11.4625, 10.50104167, 9.714583333, 11.22708333, 13.11458333, 14.16041667, 13.84375,
11.26041667, 11.27083333, 12.59479167, 13.43125, 14.03645833, 14.42291667, 15.16875,
15.74895833, 15.08020833, 10.78125, 9.505208333, 11.27395833, 14.31145833, 13.77083333,
14.00416667, 15.61979167,
15.25416667, 13.64479167, 12.61979167};
double array_3[731] = { 5.349375, 5.511145833, 4.907395833, 4.418854167, 4.727291667,
5.224270833, 5.633229167, 5.986041667, 6.826041667, 7.287604167, 7.422916667, 8.0421875,
8.335625, 8.772395833, 8.533020833, 8.2584375, 8.015, 7.133020833, 6.492916667,
7.156354167, 6.978645833, 5.5396875, 4.161354167,
3.583229167, 1.898229167, 1.592083333, 2.0190625, 2.5946875, 3.847708333, 5.808020833,
7.801875, 5.276979167, 4.180520833, 4.45375, 4.233125, 4.310625, 5.003541667, 5.265368421,
6.1171875, 5.5034375, 4.9171875, 5.800520833, 6.9975, 7.185, 6.971145833, 7.342916667,
6.7678125, 4.788333333,
2.777708333, 4.4471875, 5.256979167, 4.466875, 4.154583333, 5.005833333, 5.754791667,
5.497916667, 5.520625, 6.567604167, 6.7925, 6.318645833, 5.849270833, 4.9321875,
4.603020833, 5.009895833, 5.2296875, 4.966354167, 4.811354167, 5.2234375, 6.116979167,
6.67875, 6.937291667, 7.500625,

6.951354167, 7.055416667, 7.974583333, 6.909895833, 6.2159375, 7.4425, 8.043854167,
6.9884375, 6.033125, 7.494583333, 6.7475, 6.0709375, 6.615104167, 7.397916667,
7.627604167, 7.816041667, 9.16625, 9.158645833, 9.787395833, 9.343854167, 9.706145833,
9.28875, 9.353645833, 9.398958333,
10.42302083, 11.94020833, 13.53895833, 15.38, 16.245625, 15.4440625, 15.8134375,
14.34208333, 14.346875, 15.0084375, 16.3453125, 17.41625, 17.57333333, 16.15854167,
14.61354167, 12.92739583, 11.8321875, 12.66229167, 14.01302083, 14.92739583, 15.6609375,
15.76666667, 15.15020833,
14.89185567, 14.35822917, 14.33, 14.2540625, 13.44770833, 13.1890625, 13.9928125,
15.4440625, 16.18020833, 17.51729167, 19.21572917, 19.22114583, 18.90229167, 17.36666667,
16.71947917, 17.78364583, 20.53083333, 21.090625, 19.38041667, 19.9059375, 20.86291667,
21.37791667, 21.08572917,
19.14895833, 17.88, 16.173125, 17.07541667, 18.13479167, 18.83333333, 19.94875,
21.3809375, 22.27447917, 22.68322917, 22.71625, 21.66270833, 21.15989583, 21.89333333,
21.80416667, 22.19458333, 21.41885417, 22.86541667, 23.41270833, 22.63083333, 23.2371875,
25.06104167, 22.201875,
22.595, 23.66052083, 23.46364583, 22.95083333, 21.8678125, 22.0803125, 22.22697917,
23.041875, 24.656875, 25.52645833, 26.6003125, 26.959375, 26.37895833, 26.39645833,
25.19291667, 23.78041667, 23.10625, 22.77510417, 23.84822917, 24.56572917, 25.39,
26.03635417, 25.90385417, 26.35052083,
24.53697917, 24.70510417, 24.06322917, 22.66479167, 23.3234375, 24.40864583,
26.15104167, 26.84302083, 27.72489583, 28.66572917, 28.89677083, 28.9403125, 26.860625,
26.42989583, 26.016875, 26.4390625, 24.10645833, 23.9121875, 24.91541667, 25.2709375,
25.53447917, 25.8521875, 25.88645833,
25.07989583, 25.5471875, 26.81114583, 26.0096875, 25.34197917, 24.3553125, 23.9065625,
25.135, 26.34666667, 27.12583333, 27.15208333, 27.87229167, 28.47197917, 26.57385417,
24.291875, 23.50020833, 23.28354167, 23.06083333, 22.94177083, 23.82802083, 25.07541667,
25.97197917, 26.20614583,
24.91239583, 24.26489583, 24.06552083, 24.24510417, 25.29958333, 25.2034375,
25.17427083, 25.9203125, 26.474375, 26.5621875, 26.801875, 26.26625, 26.4553125,
25.36177083, 24.44270833, 24.08885417, 24.555, 24.81270833, 25.48635417, 26.09873684,
25.83510417, 24.1725, 23.71666667,
23.21239583, 21.883125, 20.83927083, 21.28291667, 21.81104167, 22.2721875, 22.55614583,
20.99177083, 20.43864583, 20.62458333, 20.88041667, 20.4234375, 19.94729167, 19.70552083,
19.4159375, 20.20104167, 21.46520833, 22.26322917, 22.64989583, 23.1859375, 23.51572917,
23.284375, 20.90604167,
19.0890625, 18.70645833, 19.48083333, 20.12625, 20.104375, 19.63114583, 19.10979167,
19.5334375, 20.3365625, 20.03770833, 19.36177083, 18.6546875, 18.1753125, 17.66041667,
17.33989583, 15.025625, 13.53666667, 13.04489583, 12.548125, 13.6228125, 14.47604167,
15.37239583, 16.2975,
16.9534375, 17.44291667, 15.88875, 13.90166667, 13.23520833, 13.73885417, 14.58354167,
12.67166667, 11.86447917, 11.37802083, 11.26020833, 10.82666667, 9.3203125, 8.273333333,
8.416145833, 9.366666667, 10.20739583, 11.82333333, 10.974375, 9.756875, 9.910104167,
10.58072917, 11.0221875,
7.8634375, 5.757395833, 6.3740625, 8.089270833, 5.971354167, 5.899375, 6.079166667,
6.670625, 7.1096875, 7.374895833, 8.046770833, 8.776041667, 10.06479167, 9.3584375,
7.938958333, 7.406145833, 7.149479167, 6.291354167, 5.723645833, 4.960729167,
5.659166667, 6.863333333, 6.629583333,
6.84125, 6.645833333, 6.224791667, 6.713052632, 7.9671875, 10.440625, 11.273125,
8.978854167, 7.079270833, 6.135208333, 5.9196875, 6.113645833, 7.366875, 7.703541667,
6.99, 6.371979167, 6.62, 5.395104167, 3.6625, 4.5546875, 5.929375, 1.563229167,
0.079895833, 1.687916667, 3.040833333,
5.6471875, 7.027708333, 6.857924528, 8.772395833, 8.533020833, 8.2584375, 8.015,
7.133020833, 6.492916667, 7.156354167, 6.978645833, 5.5396875, 4.161354167, 3.583229167,

1.898229167, 1.592083333, 2.0190625, 2.5946875, 3.847708333, 5.808020833, 7.801875, 5.276979167, 4.180520833,
4.45375, 4.233125, 4.310625, 5.003541667, 5.265368421, 6.1171875, 5.5034375, 4.9171875, 5.800520833, 6.9975, 7.185, 6.971145833, 7.342916667, 6.7678125, 4.788333333, 2.777708333, 4.4471875, 5.256979167, 4.466875, 4.154583333, 5.005833333, 5.754791667, 5.497916667, 5.520625, 6.567604167,
6.7925, 6.318645833, 5.849270833, 4.9321875, 4.603020833, 5.009895833, 5.2296875, 4.966354167, 4.811354167, 5.2234375, 6.116979167, 6.67875, 6.937291667, 7.500625, 6.951354167, 7.055416667, 7.974583333, 6.909895833, 6.2159375, 7.4425, 8.043854167, 6.9884375, 6.033125, 7.494583333,
6.7475, 6.0709375, 6.615104167, 7.397916667, 7.627604167, 7.816041667, 9.16625, 9.158645833, 9.787395833, 9.343854167, 9.706145833, 9.28875, 9.353645833, 9.398958333, 10.42302083, 11.94020833, 13.53895833, 15.38, 16.245625, 15.4440625, 15.8134375, 14.34208333, 14.346875, 15.0084375,
16.3453125, 17.41625, 17.57333333, 16.15854167, 14.61354167, 12.92739583, 11.8321875, 12.66229167, 14.01302083, 14.92739583, 15.6609375, 15.76666667, 15.15020833, 14.89185567, 14.35822917, 14.33, 14.2540625, 13.44770833, 13.1890625, 13.9928125, 15.4440625, 16.18020833, 17.51729167,
19.21572917, 19.22114583, 18.90229167, 17.36666667, 16.71947917, 17.78364583, 20.53083333, 21.090625, 19.38041667, 19.9059375, 20.86291667, 21.37791667, 21.08572917, 19.14895833, 17.88, 16.173125, 17.07541667, 18.13479167, 18.83333333, 19.94875, 21.3809375, 22.27447917, 22.68322917,
22.71625, 21.66270833, 21.15989583, 21.89333333, 21.80416667, 22.19458333, 21.41885417, 22.86541667, 23.41270833, 22.63083333, 23.2371875, 25.06104167, 22.201875, 22.595, 23.66052083, 23.46364583, 22.95083333, 21.8678125, 22.0803125, 22.22697917, 23.041875, 24.656875, 25.52645833,
26.6003125, 26.959375, 26.37895833, 26.39645833, 25.19291667, 23.78041667, 23.10625, 22.77510417, 23.84822917, 24.56572917, 25.39, 26.03635417, 25.90385417, 26.35052083, 24.53697917, 24.70510417, 24.06322917, 22.66479167, 23.3234375, 24.40864583, 26.15104167, 26.84302083, 27.72489583,
28.66572917, 28.89677083, 28.9403125, 26.860625, 26.42989583, 26.016875, 26.4390625, 24.10645833, 23.9121875, 24.91541667, 25.2709375, 25.53447917, 25.8521875, 25.88645833, 25.07989583, 25.5471875, 26.81114583, 26.0096875, 25.34197917, 24.3553125, 23.9065625, 25.135, 26.34666667,
27.12583333, 27.15208333, 27.87229167, 28.47197917, 26.57385417, 24.291875, 23.50020833, 23.28354167, 23.06083333, 22.94177083, 23.82802083, 25.07541667, 25.97197917, 26.20614583, 24.91239583, 24.26489583, 24.06552083, 24.24510417, 25.29958333, 25.2034375, 25.17427083, 25.9203125,
26.474375, 26.5621875, 26.801875, 26.26625, 26.4553125, 25.36177083, 24.44270833, 24.08885417, 24.555, 24.81270833, 25.48635417, 26.09873684, 25.83510417, 24.1725, 23.71666667, 23.21239583, 21.883125, 20.83927083, 21.28291667, 21.81104167, 22.2721875, 22.55614583, 20.99177083,
20.43864583, 20.62458333, 20.88041667, 20.4234375, 19.94729167, 19.70552083, 19.4159375, 20.20104167, 21.46520833, 22.26322917, 22.64989583, 23.1859375, 23.51572917, 23.284375, 20.90604167, 19.0890625, 18.70645833, 19.48083333, 20.12625, 20.104375, 19.63114583, 19.10979167, 19.5334375,
20.3365625, 20.03770833, 19.36177083, 18.6546875, 18.1753125, 17.66041667, 17.33989583, 15.025625, 13.53666667, 13.04489583, 12.548125, 13.6228125, 14.47604167, 15.37239583, 16.2975, 16.9534375, 17.44291667, 15.88875, 13.90166667, 13.23520833, 13.73885417, 14.58354167, 12.67166667,
11.86447917, 11.37802083, 11.26020833, 10.82666667, 9.3203125, 8.273333333, 8.416145833, 9.366666667, 10.20739583, 11.82333333, 10.974375, 9.756875, 9.910104167, 10.58072917, 11.0221875, 7.8634375, 5.757395833, 6.3740625, 8.089270833, 5.971354167, 5.899375, 6.079166667, 6.670625,

7.1096875, 7.374895833, 8.046770833, 8.776041667, 10.06479167, 9.3584375, 7.938958333, 7.406145833, 7.149479167, 6.291354167, 5.723645833, 4.960729167, 5.659166667, 6.863333333, 6.629583333, 6.84125, 6.645833333, 6.224791667, 6.713052632, 7.9671875, 10.440625, 11.273125, 8.978854167,
7.079270833, 6.135208333, 5.9196875, 6.113645833, 7.366875, 7.703541667, 6.99, 6.371979167};
double array_4[730] = { 153.654, 609.322, 609.874, 477.263, 177.386, 602.607, 304.802, 202.382, 475.385, 610.067, 609.962, 311.855, 475.012, 532.216, 435.247, 192.961, 669.957, 108.318, 492.542, 692.03, 167.653, 152.71, 554.195, 609.117, 478.603, 635.811, 746.949, 284.922, 750.093, 750.133, 757.072,
107.417, 208.553, 699.726, 614.803, 730.377, 710.037, 733.854, 771.639, 740.91, 778.346, 787.155, 633.822, 72.724, 238.139, 821.219, 825.214, 582.492, 467.407, 855.863, 736.231, 640.663, 633.687, 879.17, 897.949, 98.424, 362.95, 873.888, 940.395, 447.848, 809.048, 878.951,
824.655, 867.183, 909.149, 275.492, 904.637, 867.153, 842.009, 871.807, 840.34, 718.108, 723.675, 713.143, 96.551, 408.303, 1001.66, 1021.07, 931.982, 216.309, 889.37, 915.327, 769.761, 1032.997, 698.719, 841.614, 982.561, 1066.106, 1117.919, 907.707, 583.654, 904.529, 1027.573,
649.434, 1003.085, 927.638, 662.735, 1067.436, 973.153, 1030.066, 404.393, 943.172, 1097.228, 460.799, 155.767, 666.671, 767.818, 592.461, 442.707, 1151.961, 1147.735, 1152.133, 1147.37, 958.264, 1028.802, 644.204, 419.284, 540.734, 925.019, 1101.386, 1079.686, 905.61, 236.302,
722.521, 198.71, 953.659, 1129.34, 268.127, 661.105, 939.621, 852.932, 847.539, 750.376, 1117.695, 1113.333, 1074.191, 753.225, 275.824, 1084.405, 1112.626, 1099.03, 1003.971, 1149.208, 1166.13, 1153.055, 1028.043, 990.701, 1016.658, 1038.027, 1090.597, 1090.934, 1065.547,
879.919, 192.223, 1060.037, 1085.346, 965.739, 1070.321, 1062.909, 1056.636, 553.412, 880.468, 904.391, 901.77, 585.747, 577.168, 785.65, 1091.391, 1029.541, 970.618, 446.342, 1124.679, 937.715, 1144.004, 685.074, 719.377, 1038.245, 973.431, 976.946, 666.34, 740.128, 1068.468,
1129.533, 1066.506, 926.21, 697.46, 1123.925, 1063.021, 1093.929, 970.357, 910.489, 869.556, 1133.638, 493.428, 1008.239, 1010.553, 844.414, 964.888, 889.008, 882.116, 741.524, 997.3, 879.069, 950.332, 804.352, 952.524, 850.055, 939.533, 900.162, 740.219, 786.235, 902.884,
1041.147, 1116.053, 1076.321, 926.036, 791.382, 939.856, 942.725, 1025.298, 764.594, 669.56, 930.788, 1109.155, 974.629, 933.558, 811.317, 751.493, 924.73, 994.727, 632.073, 394.296, 832.585, 514.416, 460.191, 801.922, 973.379, 656.362, 778.078, 858.955, 915.045, 631.847,
805.775, 1066.135, 943.301, 1039.166, 1014.133, 1023.545, 967.551, 937.285, 976.411, 668.325, 799.35, 778.366, 999.614, 882.574, 593.93, 846.761, 910.432, 750.904, 726.759, 593.513, 252.668, 737.356, 749.562, 869.285, 947.578, 931.425, 877.907, 679.179, 739.097, 902.904,
820.698, 942.49, 775.219, 687.405, 858.958, 364.08, 714.836, 872.582, 772.719, 748.694, 644.891, 829.629, 811.129, 848.475, 832.781, 816.5, 703.43, 757.042, 419.98, 341.276, 777.709, 815.45, 798.104, 697.072, 289.789, 94.995, 91.271, 493.144, 648.515, 743.776, 740.321,
731.545, 627.038, 657.804, 655.541, 734.38, 741.313, 292.638, 689.822, 539.714, 476.268, 309.926, 689.761, 506.097, 382.369, 641.036, 81.132, 680.435, 596.556, 609.796, 185.144, 368.768, 612.402, 513.6, 495.062, 631.76, 164.386, 213.321, 445.394, 591.579, 384.164, 616.981,
576.301, 270.282, 577.937, 534.121, 197.353, 630.321, 288.547, 391.889, 346.292, 560.965, 231.395, 572.355, 220.434, 438.044, 175.936, 320.406, 587.445, 554.601, 301.539, 566.103, 220.731, 268.034, 192.52, 606.547, 506.969, 75.332, 582.947, 342.649, 71.795, 48.52306667,
532.5961838, 585.3088324, 401.7610704, 583.1623414, 598.7400591, 473.2642754, 454.2020681, 599.4973093, 506.9334399, 509.2308116, 342.3852554, 269.749022, 356.1182434,

174.9532573, 399.2074862, 454.612008, 586.4447077, 73.68955131, 545.268518, 75.40617481,
357.0434551, 681.3087957,
239.359377, 599.0475141, 160.8957335, 698.8849707, 551.6709828, 458.7968116,
676.0877551, 274.8077947, 684.4545154, 734.0259335, 207.6724107, 597.8782689,
595.4058297, 614.039031, 570.6839345, 558.4990546, 722.0721396, 525.4907417, 620.9144101,
644.6847908, 755.7822256, 179.2967997,
149.7049484, 778.4512469, 699.9125905, 769.2882372, 387.5036569, 420.4220629,
725.5860002, 548.3870277, 699.408113, 110.1808841, 189.0591887, 618.8590389, 440.3314424,
301.5701595, 501.4681223, 852.3497092, 873.8824075, 840.0686784, 819.6471169,
504.9002332, 891.8488626, 886.3768571,
115.0047828, 362.0929364, 930.650988, 905.246241, 805.5593669, 919.2413236, 911.5036021,
713.7607987, 837.4647266, 521.9765043, 973.5443757, 570.6870834, 627.9068394,
992.9558635, 1005.14625, 756.2047499, 1006.320808, 694.4396616, 1019.783054, 970.6809554,
943.7101306, 863.1080919,
686.6486563, 256.6722715, 411.8483395, 569.3040236, 1004.630676, 210.7281858,
818.3573969, 339.4433982, 141.5484411, 196.342331, 241.2261979, 539.8388784, 941.0002092,
863.6660389, 411.2303829, 508.8826114, 701.9481063, 1030.048127, 1061.163423,
1066.302574, 1059.079305, 871.569366,
223.829253, 231.8539443, 643.5802469, 782.6993095, 1047.702448, 893.8501779, 1017.86357,
173.383553, 231.5923833, 892.1134128, 968.25905, 953.4567901, 931.0665069, 969.5888256,
717.746405, 979.3926004, 934.2345716, 1008.803925, 288.2377172, 562.4082534, 437.4906381,
259.4349161,
224.6479928, 1072.79621, 823.4552876, 842.3962702, 556.7667765, 1080.182744,
685.6182594, 1032.819053, 632.8490114, 751.320027, 1040.844068, 1047.116537, 1053.389005,
1093.504718, 1067.969218, 859.884287, 457.7834781, 1094.644997, 1027.284302, 906.9783553,
857.5737717, 852.6501305,
851.1171205, 838.3097103, 825.5023001, 812.69489, 1021.54793, 1048.526669, 1096.745617,
1067.411414, 1083.217083, 1123.332712, 1045.811264, 889.0414025, 867.6408057,
915.6452816, 1001.781052, 1017.094368, 929.1495711, 977.9559866, 783.5938083,
686.9358448, 940.7758299, 1104.475942,
950.5706826, 1010.859754, 966.5870944, 1005.911973, 976.5479299, 927.7191347,
935.8317792, 1040.154611, 972.0357077, 713.2597092, 414.1229523, 571.3510031,
972.9707344, 719.7091846, 251.304988, 754.5039573, 1016.337199, 1022.816124, 1012.751703,
244.9033683, 978.9600589, 842.0228235,
1035.681944, 1059.258237, 865.0156451, 1033.009387, 973.8855144, 933.6002209,
718.9103626, 888.0894533, 867.7029266, 697.0495122, 691.5074544, 885.4518682,
691.2718572, 806.6132892, 781.6013252, 924.0530094, 923.9212673, 1069.899909,
1022.810025, 731.1443433, 948.3114631, 705.8202309,
992.3671041, 906.7777909, 878.2499901, 935.2780076, 982.3127241, 446.9539347,
933.7392127, 655.3453915, 972.7627379, 831.8753202, 837.2069197, 949.363597, 852.0609213,
1018.705915, 1034.162825, 950.157623, 1005.339481, 724.6404224, 418.8950625, 256.2674863,
211.5399771, 673.5035662,
935.4316901, 656.1472987, 965.8923435, 1073.257844, 1058.414268, 1029.747572,
628.9397127, 175.7172269, 1017.157499, 968.8627311, 562.1693595, 188.6452133,
456.8813499, 628.5458533, 870.8476928, 986.8817974, 766.8621045, 305.8698474,
743.4297095, 958.2665712, 690.0416238, 453.4037506,
790.8763371, 766.7882558, 741.1493533, 673.5756165, 87.04515956, 286.5685002,
741.2008235, 411.6859867, 789.8625968, 391.9057423, 350.646735, 926.1688563, 922.4594281,
858.7036321, 847.5994977, 719.2933733, 873.1718509, 717.0401855, 409.9256182,
635.3917117, 652.3497875, 845.1506955,
783.7205371, 803.7746329, 654.4170209, 766.5233771, 83.16750386, 230.6414219,
784.4595247, 774.2972372, 748.2756955, 765.2289413, 498.1477009, 276.4031105,
268.0134274, 727.9342156, 423.44716, 609.4522798, 629.1392968, 699.5918663, 696.0683926,
672.4594281, 395.0454697, 362.2407674,

150.2495287, 195.3615393, 197.5989797, 654.3196185, 196.7006765, 709.7149828,
718.3154042, 666.6823777, 402.8085838, 153.8510591, 137.1659088, 379.8159033,
637.9782633, 524.9889098, 524.7144283, 669.5935455, 543.0852834, 415.2961074,
645.3670844, 635.1585893, 599.1377398, 559.0357103,
613.2832428, 608.2011756, 419.5935455, 378.2383276, 79.11167794, 495.9215925,
304.3913538, 611.5472127, 523.7258248, 426.1319681, 423.3447099, 526.4220705,
443.0858931, 400.5403868, 324.294653, 59.80944255, 503.5807736, 562.1587031, 496.3680319,
396.5642776, 90.13651877, 186.20876,
191.20876, 112.096132, 143.4926052, 319.1780432, 568.2423208, 559.8634812, 273.2992036,
578.9562002, 340.8617747, 270.0142207, 476.1746303, 531.0409556, 400.463595, 400.463595};
double array_5[731] = { 206.1402977, 668.0930484, 681.6363787, 257.6960001, 427.3006039,
700.9697671, 403.9298236, 214.8637957, 708.9961819, 713.3010261, 715.5316923,
577.8791049, 603.2187896, 626.5952604, 574.0636415, 240.9916111, 442.8242194,
198.4780733, 491.5688272, 769.7306203, 426.7486279, 160.044606,
431.9980909, 532.1106155, 557.1657764, 800.6412798, 781.7716651, 661.9188832,
823.4515484, 618.6712741, 840.7733539, 71.17855954, 355.006936, 652.38699, 815.7209707,
726.8739134, 898.7253542, 855.9631746, 817.9771945, 807.9259609, 910.1440832,
940.8453557, 751.2652129, 755.7747938,
305.728369, 925.6280287, 800.8161321, 812.8082344, 996.4166944, 986.1705138,
825.0612215, 284.0548589, 886.9941368, 1009.11978, 1052.240317, 423.4820405, 1086.972404,
1065.579847, 1080.243869, 583.7099579, 728.3055268, 503.5252057, 896.1681732,
945.7297897, 946.8984875, 799.6309454,
701.323245, 668.6461753, 654.7124162, 774.2100253, 773.452579, 969.103746, 739.32335,
926.3893511, 335.2234264, 936.568407, 1037.246031, 1045.203864, 1008.200522, 904.2003568,
1026.665018, 1039.548575, 999.2738079, 1016.548695, 943.7200012, 1007.101527,
887.2205483, 275.2248505,
1118.536748, 1032.833791, 895.7094028, 803.5592462, 861.3451322, 993.4934016,
1065.558911, 790.3435525, 1076.661914, 1045.660613, 601.3021445, 1004.048188, 174.839792,
1058.254418, 1103.574344, 992.3818012, 436.3259101, 1033.504515, 1148.057862,
704.5813063, 978.3296986, 1060.960829,
1158.597481, 1158.92251, 1162.892202, 901.9131298, 1088.518985, 1048.282517,
938.0412263, 1121.949337, 1116.042471, 1157.626727, 1023.549202, 985.5849955,
764.8748047, 347.5030069, 453.7547279, 840.9466186, 923.9268292, 897.1924002,
614.0508452, 1061.120028, 1013.143006, 1007.858519,
504.355584, 1095.758903, 1098.964089, 1075.837916, 282.1396921, 814.8516765,
981.1279654, 1083.872198, 1075.460038, 1037.772942, 1109.846994, 1122.97004, 1118.142883,
1106.779394, 1086.592924, 970.9911288, 1078.355169, 1060.821601, 941.3499411, 976.927095,
390.911126, 498.878815,
1043.500485, 1011.47489, 1018.881054, 1017.047235, 1012.872174, 869.6944258,
845.7260652, 578.8280725, 989.9354526, 969.2989329, 705.9361242, 836.9151556, 992.894187,
862.2621446, 1027.5763, 946.5991344, 467.000597, 1059.400418, 921.9349302, 867.4800388,
614.5343631, 693.0023879,
967.0061936, 1079.089247, 952.5632415, 988.3889262, 517.0789493, 687.8091754,
766.7872114, 1004.935285, 1032.860286, 882.6838506, 1059.001385, 478.6309874,
401.8548071, 580.4862847, 825.5381919, 846.1680026, 909.6684561, 911.3250723,
499.3367897, 807.3211134, 994.8094082, 785.9047899,
933.0301637, 885.6437221, 600.0959087, 574.9464995, 677.8599794, 999.9730353,
950.9033804, 626.858239, 1058.638525, 771.4302961, 376.5363166, 540.2753469, 654.8288351,
804.2092587, 1087.131275, 1100.014419, 780.4673614, 1006.593852, 798.0112728,
709.5237253, 779.1772185, 845.317866,
899.3575829, 761.6211168, 746.913488, 1000.762813, 778.7261764, 951.7150347,
857.7621576, 813.7042863, 664.9823044, 954.7057281, 871.4051645, 903.3031852,
963.4421287, 922.4013632, 644.6650937, 924.8394285, 719.4324289, 836.6627343,
436.2105125, 945.1566391, 843.5705859, 887.0494167,

545.7202779, 555.4067327, 848.0667917, 1098.758818, 1074.426288, 1068.622198,
661.2197518, 1072.41718, 1063.93428, 625.0558086, 1031.11885, 302.7055987, 812.6361282,
812.6361282, 812.6361282, 812.6361282, 812.6361282, 980.5339762, 825.5647826,
535.8737387, 201.8930262, 816.7247076,
964.0146442, 981.6278239, 692.9859809, 965.8898116, 863.7824806, 409.2686847,
818.6333601, 941.7514957, 915.6263952, 945.6144279, 551.4228856, 874.3233831,
416.6442786, 493.4228856, 606.5472637, 1167.783582, 1115.597015, 956.1119403,
740.7039801, 956.7835821, 943.8706468, 916.7363184,
893.6368159, 891.6641791, 644.6318408, 844.7437811, 604.3059701, 705.7636816,
813.2985075, 676.7686567, 436.7935323, 797.3731343, 734.6940299, 576.7960199,
118.2860697, 266.2587065, 460.5547264, 726.8134328, 800.9179104, 720.5671642,
687.4149283, 632.5070547, 867.3158856, 860.4354562,
838.7982073, 611.69424, 834.4713108, 827.643446, 732.1086704, 721.4048581, 771.5127538,
775.6487578, 703.6158911, 717.2024567, 210.6623139, 761.2681901, 744.0602003,
702.1496154, 712.2115863, 697.5294638, 699.6099153, 533.2208266, 351.5437393, 597.79782,
194.5664804, 316.2064959,
563.1245504, 700.3070879, 439.3293864, 622.0190339, 681.1268433, 252.7218621,
604.2408734, 417.3847568, 234.9649342, 621.4924751, 493.7947799, 673.455874, 673.3354183,
737.9907313, 692.6377042, 701.9098525, 654.4003753, 585.5643694, 214.8605899,
417.5992267, 695.64322, 684.5348573,
443.3032905, 659.577031, 216.4764585, 680.5862618, 648.2894537, 609.9669244, 498.848042,
332.5369612, 621.5571098, 418.974184, 422.0149551, 225.842621, 700.8727586, 673.3405881,
671.9114919, 694.0994912, 688.1439749, 660.441714, 535.8085845, 686.9113082, 618.2662957,
565.4466729,
569.141826, 462.8136536, 228.4731941, 171.4573806, 647.5314526, 673.1726266,
491.3870369, 100.1079031, 331.6186383, 89.5519496, 718.4795306, 741.5244182, 566.4174335,
635.0539056, 339.7234926, 724.2984003, 192.7116278, 725.0841185, 726.9459291,
684.9014638, 566.9327052, 371.0013408,
243.0432766, 627.5325556, 659.9115202, 447.3391844, 555.1025723, 627.8422348,
733.1206964, 729.0773841, 698.9111278, 749.5911378, 750.6525384, 433.0340005,
350.2747155, 738.7348817, 588.3131756, 644.722252, 607.1886261, 313.0432766, 808.567555,
783.4385815, 546.5614185, 254.5988084,
597.5586, 497.0826644, 743.9944343, 402.8377752, 619.1162724, 872.9258982, 856.1607621,
780.2055869, 811.5595761, 639.826174, 586.8876483, 824.1692823, 849.4118614, 109.0393238,
654.6089289, 911.5036021, 873.4821989, 637.8291932, 888.1560697, 861.982184, 542.3077153,
747.7442232,
625.7337907, 908.0054106, 696.8096284, 623.8503273, 952.7753116, 956.3175201,
831.9927211, 895.004185, 670.8372941, 990.9380325, 957.3530775, 965.4915852, 905.5149762,
494.9709294, 122.0011658, 309.291095, 498.5750157, 388.1920904, 115.994978, 533.6430216,
426.4888052, 536.8173258,
583.4672526, 713.571877, 739.2864616, 706.5369324, 772.2284997, 847.5245867,
694.7122829, 605.9928855, 899.9979075, 970.3808328, 966.4260306, 921.7932622,
950.3368906, 888.6252354, 775.8087466, 672.3686964, 783.0362, 919.2948316, 888.5059636,
873.8041431, 679.395271, 510.3410755,
948.2297552, 918.7947269, 848.363541, 721.8375524, 686.3447424, 805.0891252, 432.06074,
802.5801378, 878.0388706, 578.4189635, 780.3943229, 791.5799131, 403.9713152, 821.44997,
948.5489065, 758.7870731, 624.0020222, 686.7379419, 986.9850959, 765.308568, 956.1357849,
908.1804224,
761.1593769, 717.2895446, 705.2351708, 375.4680947, 892.5423158, 889.9677951,
909.5360246, 455.4280258, 781.1769772, 926.9023367, 858.6522618, 813.9630735,
689.7631481, 851.6076091, 852.1689668, 972.0458784, 1082.306975, 1088.964938, 1058.85677,
1055.7348, 1042.817978, 1062.284595,
1056.954495, 1051.920925, 1122.008579, 515.7030959, 1046.50317, 826.079821, 775.4289444,
954.4143976, 317.3498695, 320.4737038, 760.8560239, 655.8317792, 1029.488997,

927.1316673, 963.9071242, 924.1999254, 1059.255875, 867.1782917, 911.8314062, 858.021351,
986.1862691, 1053.423523,
974.4432174, 922.7369777, 861.6436591, 923.0756488, 836.8396834, 540.4564697,
524.7064237, 555.0174857, 1060.67182, 1065.111357, 981.7540953, 887.9348426, 998.5956194,
1008.919566, 855.1794589, 599.0226394, 694.4763482, 914.9236149, 1003.152954,
762.8437327, 715.5383766, 685.2604454,
680.6478925, 806.8157556, 858.722621, 859.3410639, 887.7655071, 909.4128474,
856.8506916, 932.681168, 878.5967608, 890.0165504, 881.1936005, 951.9269417, 916.3494503,
746.867242, 948.809946, 524.4138393, 849.6906648, 977.0559956, 195.5530599, 1009.445561,
858.3126453, 574.4158096,
411.1774441, 941.2854159, 970.9678055, 985.191315, 465.6874335, 603.4263309,
956.6260787, 922.9045987, 504.5769792, 619.9885723, 632.1216062, 566.3888561,
800.2423454, 863.6107499, 713.6994129, 935.4943383, 1009.969565, 1001.239762, 996.752898,
463.098062, 877.7424697, 895.1461308,
890.0192454, 768.5762879, 383.3594414, 804.0840532, 885.1452356, 859.4235331,
876.046189, 917.9854988, 566.5420937, 401.389697, 909.1370899, 849.9977622, 399.6464217,
528.5973235, 358.1255874, 788.6496889, 615.7745155, 146.5492548, 554.8247773,
732.9924361, 734.3485655, 790.0617643,
390.4019156, 809.9232032, 917.5207689, 896.3461167, 886.8503671, 887.3164606,
907.9332496, 793.4577859, 698.7031491, 343.76449, 358.7760819, 230.3395479, 461.5243431,
904.3131762, 827.0503284, 808.2592736, 787.3695904, 546.9667697, 148.2974304,
787.7608192, 783.0226043, 767.8757728,
765.5718702, 777.827956, 213.2993624, 574.98551, 731.4697643, 611.8455371, 695.3777048,
716.7986862, 706.1509853, 712.2488408, 800.4158811, 811.1622491, 85.69646224,
251.3724077, 339.078962, 760.4164356, 747.0500166, 623.9630698, 739.8247754, 734.6040812,
725.4713319, 412.5429744,
400.609959, 231.4184319, 413.5244538, 704.0617722, 691.8847732, 501.8964179,
683.6336919, 659.6678496, 437.1853166, 657.6743928, 626.0923811, 552.6838195,
616.7184208, 627.8945325, 618.1878674, 574.523123, 84.8092492, 244.8541644, 626.6865757,
396.9055745, 628.0176337, 580.4664391,
385.6598407, 454.8691695, 598.3276451, 600.1450512, 330.3384528, 360.2872582,
226.0267349, 501.8714448, 584.013083, 431.0409556, 364.5022753, 390.6001138, 466.9226394,
219.0187713, 453.486917, 427.6279863, 225.5432309, 579.5875995, 559.0216155, 331.4163823,
162.1103527, 392.8697383,
399.0358362, 570.1109215, 164.2690557, 591.1575654, 585.2957907};
double array_6[623] = { 13.84922917, 8.867895833, 4.84115625, 6.01696875, 8.120729167,
11.09189583, 13.12505208, 14.3118125, 15.48094792, 15.9866875, 16.81670833, 15.48052083,
13.73097917, 12.53708333, 10.00202083, 8.160260417, 9.826854167, 11.0585, 10.36289583,
8.532552083, 6.214791667, 7.604864583, 11.26375,
10.31080208, 7.317177083, 8.172645833, 8.585322917, 8.599364583, 9.565864583,
11.3366875, 10.07169792, 11.51039583, 13.01390625, 13.74336458, 15.08461458, 15.21529167,
15.72745833, 15.83222917, 14.38879167, 13.85019792, 13.046875, 10.56817708, 12.1341875,
14.53445833, 12.64639583,
12.83751042, 12.52713542, 12.92685417, 14.63127083, 13.64329167, 13.25052083,
10.39365625, 11.87057292, 13.76933333, 12.09073958, 12.34348958, 13.021125, 11.75995833,
8.269697917, 10.0990625, 12.28966667, 12.78232292, 14.53736458, 15.67559375, 16.33433333,
16.02575, 14.44230208,
15.14823958, 12.73959375, 13.62136458, 14.27780208, 14.66551042, 15.38313542, 15.602125,
16.38273958, 15.51288542, 13.88980208, 13.76297917, 15.94617708, 16.74434375,
15.17428125, 16.21516667, 14.39440625, 12.89855208, 12.30571875, 13.17730208,
16.00490625, 17.12648958, 13.39789583,
11.86520833, 14.04025, 17.22586458, 16.14992708, 13.88169792, 16.57653125, 18.03329167,
18.64675, 16.18647917, 15.16355208, 15.16159375, 16.62715625, 18.34153125, 19.25232292,
17.79702083, 16.89695833, 14.85125, 13.64821875, 14.81115625, 17.25454167, 18.79808333,
18.12576042, 18.25601042,

17.76867708, 19.61566667, 21.41322917, 22.5575, 22.27877083, 21.9875, 21.22075,
20.50152083, 18.98309375, 19.83695833, 21.18002083, 22.15692708, 22.29538542,
21.52698958, 21.37086458, 22.22454167, 22.10388542, 22.36977083, 22.39377083, 20.6625,
19.12933333, 17.41786458, 18.76476042,
20.17678125, 21.560375, 21.49388542, 21.44975, 22.061625, 22.57926042, 23.22158333,
23.6481875, 23.95261458, 22.22779167, 22.38871875, 22.99241667, 23.03008333, 22.12952083,
20.81296875, 23.35817708, 24.79523958, 25.94533333, 25.15064583, 25.02309375,
25.91803125, 26.81538542, 26.89642708,
27.63426042, 27.92405208, 27.884375, 27.89017582, 28.901875, 28.108125, 27.72507292,
28.46047917, 27.71436458, 27.66891667, 28.11044792, 26.6035, 27.21230208, 26.36907292,
25.1970625, 26.45308333, 26.20282292, 27.08971875, 28.08283333, 27.94436458, 28.11717708,
28.63173958, 28.40721875,
27.53073958, 28.109375, 27.44632292, 28.08514583, 28.33113542, 28.2973125, 27.31585417,
27.43135417, 27.37690625, 26.87379167, 27.02108333, 27.0024375, 27.316, 28.29830208,
28.48489583, 28.4934375, 28.00816667, 27.70251042, 28.02013542, 28.44367708, 27.30232292,
28.31273958, 28.96870833,
28.02802083, 27.80113542, 28.03552083, 28.4979375, 28.32869792, 29.15208333,
28.24602083, 28.30348958, 28.73953125, 28.86892708, 28.94402083, 28.90927083,
29.26815625, 28.67809375, 29.93155208, 30.13884375, 30.39761458, 29.969875, 29.02523958,
28.54554167, 29.24590625, 29.75757292,
29.83461458, 28.9781875, 27.95044792, 28.16563542, 27.89554167, 28.53671875, 28.918875,
28.95073958, 28.89986458, 29.16801042, 29.59483333, 29.87908333, 29.43444792,
29.22808333, 28.82926042, 28.626, 28.41941667, 27.73133333, 26.67436458, 26.67047917,
26.80090625, 27.30340625, 27.51442708,
27.37401042, 27.05673958, 26.49816667, 27.23265625, 27.88880208, 27.18872917,
27.17430208, 27.27458333, 26.93933333, 25.04772917, 24.22538542, 23.89459375,
24.33419792, 24.50970833, 24.60244792, 25.50845833, 26.36553125, 25.85636458,
26.18894792, 26.67722917, 26.9231875, 26.99066667,
25.0385625, 23.90726042, 24.05644792, 24.30779167, 24.53617647, 25.49901667,
25.35829167, 25.33204167, 25.82525, 26.17907292, 26.23428125, 25.82635417, 25.27469792,
22.77302083, 22.80465625, 23.04045833, 22.79878125, 22.90201042, 23.565875, 24.19358333,
24.97270833, 24.40304167,
23.11952083, 23.67053125, 24.40882292, 25.12734375, 24.62560417, 24.17958333,
23.78320833, 22.12361458, 19.98558333, 19.44383333, 19.5618125, 20.26267708, 18.61691667,
18.19570833, 18.69635417, 18.29601042, 17.69103125, 15.88655208, 15.28348958,
15.35685417, 15.088625, 14.602875,
15.12794792, 15.54671875, 16.70779167, 16.2370625, 13.030125, 13.55295833, 13.99584375,
15.17151042, 15.47058333, 16.12153125, 17.07128125, 15.33745833, 13.06320833,
12.84977083, 14.93502083, 16.098625, 13.37008333, 14.57358333, 13.41373958, 12.73295833,
13.01813542, 14.28560417,
11.04923958, 11.47863542, 10.32078125, 10.67061458, 12.13454167, 13.29320833,
14.37982292, 14.83265625, 15.55279167, 15.70051042, 15.93986458, 14.37619792,
15.10394792, 10.30432292, 11.06731959, 12.08796875, 12.3929375, 11.716375, 12.99622917,
14.22408333, 14.51357292, 13.29980208,
12.79008333, 12.86564583, 13.524375, 14.68129167, 14.68373958, 14.59866667, 13.85192708,
9.75178125, 7.399166667, 9.131302083, 9.782291667, 11.03259375, 11.98633333, 12.80114583,
13.84788542, 14.31271875, 15.17671875, 14.64430208, 14.03809375, 13.26611458,
11.42188542, 10.08592708,
10.38592708, 11.4866875, 10.73825, 9.689072917, 8.217395833, 9.067322917, 11.19666667,
10.52695833, 8.6408125, 9.171072917, 9.576375, 9.736416667, 10.086375, 11.72073958,
10.61398958, 11.97140625, 13.01965625, 13.61823958, 13.66558333, 14.04963542, 14.8115,
15.08795833, 14.42110417,
14.08128125, 12.9568125, 11.31451042, 12.27586458, 13.99486458, 12.90435417, 13.059,
12.80770833, 12.89005208, 14.5556875, 13.90602083, 13.37054167, 11.54754167, 12.06066667,

```
13.43214583,  12.50109375,  12.96247917,  13.48745833,  12.768125,  10.80441667,  12.05776042,
13.4270625,  13.69179167,
  14.8131875,  15.26255208,  15.95321875,  16.09141667,  14.8604375,  15.08477083,  13.65907292,
14.474,  14.70247917,  15.03846875,  15.37482292,  15.481375,  16.23480208,  15.37248958,
14.49520833,  14.31867708,  15.59657292,  16.81709375,  15.65026042,  16.25241667,  15.232875,
14.10134375,  13.98655208,
  13.62366667,  15.76461458,  17.22292708,  14.63407292,  12.74464583,  14.53338542,
16.56616667,  16.06788542,  14.3585625,  15.96067708,  16.80655208,  18.0775625,  15.82916667,
15.49247917,  15.68578125,  16.88746875,  18.25428125,  19.0849375,  18.58034375,  17.50401042,
16.27891667,  15.66126042,
  16.25060417,  17.11692708,  17.96808333,  18.08003125,  18.53216667,  18.25923958,
18.7630625,  19.5306875,  20.62573958,  21.17641667,  21.20946875,  20.66035417,  20.61930208,
19.71445833,  20.001875,  20.69684375,  21.59030208,  21.90023958,  21.60333333,  22.06865625,
22.50767708,  22.25548958,
  22.50815625,  22.8451875,  21.3410625,  19.74035417,  18.75961458,  19.83114583,  20.30135417,
21.42178125,  21.57021875,  21.29584375,  21.7515,  22.33782292,  22.89734375,  23.23935417,
23.60639583,  22.40063542,  23.05729167,  23.93426042,  23.95503125,  22.48659375,
22.07705208,  23.70335417,
  24.82154167,  25.70838542,  25.40366667,  25.46689583,  26.12975,  27.02747917,  27.15886458,
27.67567708,  28.07211458,  28.18422917,  28.36678125,  28.67704348,  28.20345833,
27.74511458,  28.23230208,  27.16820833,  27.44621875,  27.52991667,  26.17838542,  26.9433125,
26.07832292,  25.10767708,
  26.35154167,  26.28254167,  27.53025,  28.384,  28.00897917,  27.77596875,  27.84917708,
27.53798958,  26.84592708,  27.24361458,  26.67758333,  27.15541667,  27.75032292,  27.9425,
27.50071875,  27.66332292,  27.50652083,  26.87529167,  27.05045833,  26.77452083,
27.28679167,  27.58746875,  28.09123958,
  28.2404375,  28.00148958,  27.82670833,  28.2935,  28.3676875,  27.93655208,  28.35359375,
29.37557292,  28.04086458,  27.918125,  27.86046875,  28.1141875,  28.36529167,  28.79295833,
29.28204167,  29.08813542,  28.9829375,  28.81372917,  29.26165625,  28.69492708,  29.16132292,
29.24111458,  29.47523958,
  29.33694792,  28.43736458,  27.89536458,  27.39603125,  27.06913542,  26.86216667,
25.63642708,  25.88532292,  26.68582292,  26.76176042,  26.63702083,  27.09860417,  27.6638125,
28.00996875,  27.37613542,  26.41144792,  26.35965625,  26.89927083,  27.61889583,
27.68419792,  27.72770833,  28.02897917,
  28.41764583,  29.12664583,  28.9355,  28.17391667,  27.27848958,  27.74438542,  27.66101042,
27.12733333,  27.13732292,  27.78622917,  28.42789583,  28.36023958,  27.8020625,  27.87063542,
27.95416667,  28.5325625,  28.44573958};
double array_7[232] = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,  15.73,  16.43,  18.37,  18.73,  18.51,  19.16,  19.81,  20.17,  18.32,  19.69,  20.44,  20.56,  20.3,
19.84,  20.43,  20.43,  20.43,  20.43,  20.43,  20.81,  20.75,  20.76,
  21.13,  20.6,  19.73,  20.43,  20.43,  20.43,  20.43,  22.68,  21.5,  20.2,  21.41,  21.77,  22.24,  21.89,
21.86,  21.26,  21.69,  22.37,  22.32,  21.81,  22.58,  22.8,  22.27,  21.59,  21.58,  22.08,  22.44,  22.95,
22.95,  22.95,  22.95,  23.27,  22.48,  23.09,  23.47,  23.42,  23.73,  23.49,  23.4,
  23.48,  23.62,  23.79,  23.5,  23.38,  22.95,  22.95,  22.95,  23.6,  23.8,  24.1,  23.7,  23.5,  21.6,  22.9,
23.2,  23.3,  21.8,  22.6,  22.6,  22.6,  24.1,  23.1,  22.6,  22.6,  22.6, 23, 22};
double array_8[232] = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,  15.73,  16.43,  18.37,  18.73,  18.51,  19.16,  19.81,  20.17,  18.32,  19.69,  20.44,  20.56,  20.3,
19.84,  20.43,  20.43,  20.43,  20.43,  20.43,  20.81,  20.75,  20.76,
```

```
  21.13,  20.6,  19.73,  20.43,  20.43,  20.43,  20.43,  22.68,  21.5,  20.2,  21.41,  21.77,  22.24,  21.89,
21.86,  21.26,  21.69,  22.37,  22.32,  21.81,  22.58,  22.8,  22.27,  21.59,  21.58,  22.08,  22.44,  22.95,
22.95,  22.95,  22.95,  23.27,  22.48,  23.09,  23.47,  23.42,  23.73,  23.49,  23.4,
  23.48,  23.62,  23.79,  23.5,  23.38,  22.95,  22.95,  22.95,  23.6,  23.8,  24.1,  23.7,  23.5,  21.6,  22.9,
23.2,  23.3,  21.8,  22.6,  22.6,  22.6,  24.1,  23.1,  22.6,  22.6,  22.6, 23, 22};
/* STRUCTURE TYPE DECLARATIONS */

class Eve_Plastochrone_Intervalstype : public submodeltype {
public:
   Eve_Plastochrone_Intervalstype () {
   }; /* end(procedure,structor) */
   ~Eve_Plastochrone_Intervalstype () {
   }; /* end(procedure,structor) */
   void* baseptrs[1];
   Eve_Plastochrone_Intervalstype* next;
   int instanceid[1];
   BOOLEAN new_instance;
   BOOLEAN I_am_Eve;
   double Eve_PI;
   int count;
   double NewNodes;
   double Leaves_Shed;
   double Nodessquared;
   double above;
   double below;
void* get_pointer (int id, int** dims) {
   switch (id) {
      case 1:
      return(&(next));
      break; // end(case,1)
      case 2:
      return(&(instanceid[step_list(dims, 2)]));
      break; // end(case,2)
      case 3:
      return(&(new_instance));
      break; // end(case,3)
      case 4:
      return(&(I_am_Eve));
      break; // end(case,4)
      case 5:
      return(&(Eve_PI));
      break; // end(case,5)
      case 6:
      return(&(count));
      break; // end(case,6)
      case 7:
      return(&(NewNodes));
      break; // end(case,7)
      case 8:
      return(&(Leaves_Shed));
      break; // end(case,8)
      case 9:
      return(&(Nodessquared));
      break; // end(case,9)
      case 10:
```

```
      return(&(above));
      break; // end(case,10)
      case 11:
      return(&(below));
      break; // end(case,11)
   }; /* end(switch,id) */
   return(NULL);
}; /* end(procedure,get_pointer) */

}; /* end(class,Eve_Plastochrone_Intervalstype) */

class EveStatstype : public submodeltype {
public:
   EveStatstype () {
   }; /* end(procedure,structor) */
   ~EveStatstype () {
   }; /* end(procedure,structor) */
   double Tagged_PI;
   double GreatestPI;
   double mean_number__of_new_nodes;
   double Greatest_New_Nodes;
   double mean_new_leaves;
   double stdevnodes;
   double leastnodes;
   double aboveavg;
   double abovestdev;
   double avgbelow;
   double belowstdev;
   int Tagged_PI_sum_0;
   double Tagged_PI_sum;
   double GreatestPI_greatest;
   int mean_number__of_new_nodes_sum_0;
   double mean_number__of_new_nodes_sum;
   double Greatest_New_Nodes_greatest;
   int mean_new_leaves_sum_0;
   double mean_new_leaves_sum;
   double stdevnodes_sum_1;
   double stdevnodes_sum_0;
   int stdevnodes_sum;
   double leastnodes_least;
   int aboveavg_count;
   double aboveavg_sum;
   double abovestdev_sum_0;
   int abovestdev_sum;
   int avgbelow_count;
   double avgbelow_sum;
   double belowstdev_sum_0;
   int belowstdev_sum;
void* get_pointer (int id, int** dims) {
   switch (id) {
      case 1:
      return(&(Tagged_PI));
      break; // end(case,1)
      case 2:
      return(&(GreatestPI));
```

```
break; // end(case,2)
case 3:
return(&(mean_number__of_new_nodes));
break; // end(case,3)
case 4:
return(&(Greatest_New_Nodes));
break; // end(case,4)
case 5:
return(&(mean_new_leaves));
break; // end(case,5)
case 6:
return(&(stdevnodes));
break; // end(case,6)
case 7:
return(&(leastnodes));
break; // end(case,7)
case 8:
return(&(aboveavg));
break; // end(case,8)
case 9:
return(&(abovestdev));
break; // end(case,9)
case 10:
return(&(avgbelow));
break; // end(case,10)
case 11:
return(&(belowstdev));
break; // end(case,11)
case 12:
return(&(Tagged_PI_sum_0));
break; // end(case,12)
case 13:
return(&(Tagged_PI_sum));
break; // end(case,13)
case 14:
return(&(GreatestPI_greatest));
break; // end(case,14)
case 15:
return(&(mean_number__of_new_nodes_sum_0));
break; // end(case,15)
case 16:
return(&(mean_number__of_new_nodes_sum));
break; // end(case,16)
case 17:
return(&(Greatest_New_Nodes_greatest));
break; // end(case,17)
case 18:
return(&(mean_new_leaves_sum_0));
break; // end(case,18)
case 19:
return(&(mean_new_leaves_sum));
break; // end(case,19)
case 20:
return(&(stdevnodes_sum_1));
break; // end(case,20)
```

```
      case 21:
      return(&(stdevnodes_sum_0));
      break; // end(case,21)
      case 22:
      return(&(stdevnodes_sum));
      break; // end(case,22)
      case 23:
      return(&(leastnodes_least));
      break; // end(case,23)
      case 24:
      return(&(aboveavg_count));
      break; // end(case,24)
      case 25:
      return(&(aboveavg_sum));
      break; // end(case,25)
      case 26:
      return(&(abovestdev_sum_0));
      break; // end(case,26)
      case 27:
      return(&(abovestdev_sum));
      break; // end(case,27)
      case 28:
      return(&(avgbelow_count));
      break; // end(case,28)
      case 29:
      return(&(avgbelow_sum));
      break; // end(case,29)
      case 30:
      return(&(belowstdev_sum_0));
      break; // end(case,30)
      case 31:
      return(&(belowstdev_sum));
      break; // end(case,31)
   }; /* end(switch,id) */
   return(NULL);
}; /* end(procedure,get_pointer) */

}; /* end(class,EveStatstype) */

class Physical_Settingtype : public submodeltype {
public:
   Physical_Settingtype () {
   }; /* end(procedure,structor) */
   ~Physical_Settingtype () {
   }; /* end(procedure,structor) */
   double water_depth_SA;
   double sediments_SA;
   double Nitrogen_SA;
   double k_SA;
void* get_pointer (int id, int** dims) {
   switch (id) {
      case 1:
      return(&(water_depth_SA));
      break; // end(case,1)
      case 2:
```

```
      return(&(sediments_SA));
      break; // end(case,2)
      case 3:
      return(&(Nitrogen_SA));
      break; // end(case,3)
      case 4:
      return(&(k_SA));
      break; // end(case,4)
   }; /* end(switch,id) */
   return(NULL);
}; /* end(procedure,get_pointer) */

}; /* end(class,Physical_Settingtype) */

class clocktype : public submodeltype {
public:
   clocktype () {
   }; /* end(procedure,structor) */
   ~clocktype () {
   }; /* end(procedure,structor) */
   int month;
   int days;
   int year;
   int dayofyear;
   double dayreal;
void* get_pointer (int id, int** dims) {
   switch (id) {
      case 1:
      return(&(month));
      break; // end(case,1)
      case 2:
      return(&(days));
      break; // end(case,2)
      case 3:
      return(&(year));
      break; // end(case,3)
      case 4:
      return(&(dayofyear));
      break; // end(case,4)
      case 5:
      return(&(dayreal));
      break; // end(case,5)
   }; /* end(switch,id) */
   return(NULL);
}; /* end(procedure,get_pointer) */

}; /* end(class,clocktype) */

class Temperaturetype : public submodeltype {
public:
   Temperaturetype () {
   }; /* end(procedure,structor) */
   ~Temperaturetype () {
   }; /* end(procedure,structor) */
   diffs daysabove_20box_extras;
```

```
    double flow1;
    double daysabove_20box;
    double Temperature_SA;
    double Daysabove20;
    diffs FDegreeDays_extras;
    double FDegreeDays;
    double FDays;
    diffs GDegreeDays_extras;
    double GDegreeDays;
    double GDays;
void* get_pointer (int id, int** dims) {
    switch (id) {
        case 1:
        return(&(daysabove_20box_extras));
        break; // end(case,1)
        case 2:
        return(&(flow1));
        break; // end(case,2)
        case 3:
        return(&(daysabove_20box));
        break; // end(case,3)
        case 4:
        return(&(Temperature_SA));
        break; // end(case,4)
        case 5:
        return(&(Daysabove20));
        break; // end(case,5)
        case 6:
        return(&(FDegreeDays_extras));
        break; // end(case,6)
        case 7:
        return(&(FDegreeDays));
        break; // end(case,7)
        case 8:
        return(&(FDays));
        break; // end(case,8)
        case 9:
        return(&(GDegreeDays_extras));
        break; // end(case,9)
        case 10:
        return(&(GDegreeDays));
        break; // end(case,10)
        case 11:
        return(&(GDays));
        break; // end(case,11)
    }; /* end(switch,id) */
    return(NULL);
}; /* end(procedure,get_pointer) */

}; /* end(class,Temperaturetype) */

class SGR_Adulttype : public submodeltype {
public:
    SGR_Adulttype () {
    }; /* end(procedure,structor) */
```

```
    ~SGR_Adulttype () {
    }; /* end(procedure,structor) */
    double sediment__limitation_A;
    double SGR_A;
    double umax_A;
    double alpha_A;
    double ro_A;
    double Set_GMAXNW_A;
void* get_pointer (int id, int** dims) {
    switch (id) {
        case 1:
        return(&(sediment__limitation_A));
        break; // end(case,1)
        case 2:
        return(&(SGR_A));
        break; // end(case,2)
        case 3:
        return(&(umax_A));
        break; // end(case,3)
        case 4:
        return(&(alpha_A));
        break; // end(case,4)
        case 5:
        return(&(ro_A));
        break; // end(case,5)
        case 6:
        return(&(Set_GMAXNW_A));
        break; // end(case,6)
    }; /* end(switch,id) */
    return(NULL);
}; /* end(procedure,get_pointer) */

}; /* end(class,SGR_Adulttype) */

class SGR_Seedlingtype : public submodeltype {
public:
    SGR_Seedlingtype () {
    }; /* end(procedure,structor) */
    ~SGR_Seedlingtype () {
    }; /* end(procedure,structor) */
    double SGR_S;
    double umax_S;
    double sediment_limitation_S;
    double F_I_T__S;
void* get_pointer (int id, int** dims) {
    switch (id) {
        case 1:
        return(&(SGR_S));
        break; // end(case,1)
        case 2:
        return(&(umax_S));
        break; // end(case,2)
        case 3:
        return(&(sediment_limitation_S));
        break; // end(case,3)
```

```
      case 4:
      return(&(F_I_T__S));
      break; // end(case,4)
   }; /* end(switch,id) */
   return(NULL);
}; /* end(procedure,get_pointer) */

}; /* end(class,SGR_Seedlingtype) */

class Grass_Machinetype : public submodeltype {
public:
   Grass_Machinetype () {
   }; /* end(procedure,structor) */
   ~Grass_Machinetype () {
   }; /* end(procedure,structor) */
   SGR_Adulttype SGR_Adult;
   SGR_Seedlingtype SGR_Seedling;
void* get_pointer (int id, int** dims) {
   switch (id) {
      case 1:
      return(&(SGR_Adult));
      break; // end(case,1)
      case 2:
      return(&(SGR_Seedling));
      break; // end(case,2)
   }; /* end(switch,id) */
   return(NULL);
}; /* end(procedure,get_pointer) */

}; /* end(class,Grass_Machinetype) */

class Light_Adulttype : public submodeltype {
public:
   Light_Adulttype () {
   }; /* end(procedure,structor) */
   ~Light_Adulttype () {
   }; /* end(procedure,structor) */
   double kcanopy_A;
   double Percent_Irradiance_A;
   double Izleaf_A;
   double Izleaf_A_at_phase;
void* get_pointer (int id, int** dims) {
   switch (id) {
      case 1:
      return(&(kcanopy_A));
      break; // end(case,1)
      case 2:
      return(&(Percent_Irradiance_A));
      break; // end(case,2)
      case 3:
      return(&(Izleaf_A));
      break; // end(case,3)
      case 4:
      return(&(Izleaf_A_at_phase));
      break; // end(case,4)
```

```
   }; /* end(switch,id) */
   return(NULL);
}; /* end(procedure,get_pointer) */

}; /* end(class,Light_Adulttype) */

class Light_Seedlingtype : public submodeltype {
public:
   Light_Seedlingtype () {
   }; /* end(procedure,structor) */
   ~Light_Seedlingtype () {
   }; /* end(procedure,structor) */
   double Surface_PAR_S;
   double Iz_SA;
   double Percent_Irradiance_S;
void* get_pointer (int id, int** dims) {
   switch (id) {
      case 1:
      return(&(Surface_PAR_S));
      break; // end(case,1)
      case 2:
      return(&(Iz_SA));
      break; // end(case,2)
      case 3:
      return(&(Percent_Irradiance_S));
      break; // end(case,3)
   }; /* end(switch,id) */
   return(NULL);
}; /* end(procedure,get_pointer) */

}; /* end(class,Light_Seedlingtype) */

class Lighttype : public submodeltype {
public:
   Lighttype () {
   }; /* end(procedure,structor) */
   ~Lighttype () {
   }; /* end(procedure,structor) */
   Light_Adulttype Light_Adult;
   Light_Seedlingtype Light_Seedling;
void* get_pointer (int id, int** dims) {
   switch (id) {
      case 1:
      return(&(Light_Adult));
      break; // end(case,1)
      case 2:
      return(&(Light_Seedling));
      break; // end(case,2)
   }; /* end(switch,id) */
   return(NULL);
}; /* end(procedure,get_pointer) */

}; /* end(class,Lighttype) */

class Specific_Growth_Ratetype : public submodeltype {
```

```
public:
   Specific_Growth_Ratetype () {
   }; /* end(procedure,structor) */
   ~Specific_Growth_Ratetype () {
   }; /* end(procedure,structor) */
   Physical_Settingtype Physical_Setting;
   clocktype clock;
   Temperaturetype Temperature;
   Grass_Machinetype Grass_Machine;
   Lighttype Light;
void* get_pointer (int id, int** dims) {
   switch (id) {
      case 1:
      return(&(Physical_Setting));
      break; // end(case,1)
      case 2:
      return(&(clock));
      break; // end(case,2)
      case 3:
      return(&(Temperature));
      break; // end(case,3)
      case 4:
      return(&(Grass_Machine));
      break; // end(case,4)
      case 5:
      return(&(Light));
      break; // end(case,5)
   }; /* end(switch,id) */
   return(NULL);
}; /* end(procedure,get_pointer) */

}; /* end(class,Specific_Growth_Ratetype) */

class Node_births_and_lengthstype : public submodeltype {
public:
   Node_births_and_lengthstype () {
   }; /* end(procedure,structor) */
   ~Node_births_and_lengthstype () {
   }; /* end(procedure,structor) */
   void* baseptrs[1];
   int channelId;
   Node_births_and_lengthstype* next;
   int instanceid[1];
   BOOLEAN new_instance;
   double my_internode_length;
   int index;
   double node_age;
   double node_birthday;
   double NodeXNW;
   double NodeY;
   double node_age_at_phase;
   double node_birthday_at_phase;
void* get_pointer (int id, int** dims) {
   switch (id) {
      case 1:
```

```
      return(&(next));
      break; // end(case,1)
      case 2:
      return(&(instanceid[step_list(dims, 2)]));
      break; // end(case,2)
      case 3:
      return(&(new_instance));
      break; // end(case,3)
      case 4:
      return(&(my_internode_length));
      break; // end(case,4)
      case 5:
      return(&(index));
      break; // end(case,5)
      case 6:
      return(&(node_age));
      break; // end(case,6)
      case 7:
      return(&(node_birthday));
      break; // end(case,7)
      case 8:
      return(&(NodeXNW));
      break; // end(case,8)
      case 9:
      return(&(NodeY));
      break; // end(case,9)
      case 10:
      return(&(node_age_at_phase));
      break; // end(case,10)
      case 11:
      return(&(node_birthday_at_phase));
      break; // end(case,11)
   }; /* end(switch,id) */
   return(NULL);
}; /* end(procedure,get_pointer) */

}; /* end(class,Node_births_and_lengthstype) */

class Adulthoodtype : public submodeltype {
public:
   Adulthoodtype () {
   }; /* end(procedure,structor) */
   ~Adulthoodtype () {
   }; /* end(procedure,structor) */
   BOOLEAN is_adult;
void* get_pointer (int id, int** dims) {
   switch (id) {
      case 1:
      return(&(is_adult));
      break; // end(case,1)
   }; /* end(switch,id) */
   return(NULL);
}; /* end(procedure,get_pointer) */

}; /* end(class,Adulthoodtype) */
```

```
class Flowertype : public submodeltype {
public:
   Flowertype () {
   }; /* end(procedure,structor) */
   ~Flowertype () {
   }; /* end(procedure,structor) */
   BOOLEAN flower;
void* get_pointer (int id, int** dims) {
   switch (id) {
      case 1:
      return(&(flower));
      break; // end(case,1)
   }; /* end(switch,id) */
   return(NULL);
}; /* end(procedure,get_pointer) */

}; /* end(class,Flowertype) */

class Ramettype : public submodeltype {
public:
   Ramettype () {
      Node_births_and_lengths = 0;
   }; /* end(procedure,structor) */
   ~Ramettype () {
      delete_list(Node_births_and_lengths);
   }; /* end(procedure,structor) */
   void* baseptrs[1];
   int channelId;
   Ramettype* next;
   int instanceid[1];
   BOOLEAN new_instance;
   Specific_Growth_Ratetype Specific_Growth_Rate;
   Node_births_and_lengthstype *Node_births_and_lengths;
   Adulthoodtype Adulthood;
   Flowertype Flower;
   int Branching_Rate_Fix;
   BOOLEAN Negative_Root_biomass;
   int Negative_Root_biomass_0;
   int countflowers;
   int adult_status;
   diffs LeavesProduced_extras;
   int Me_0;
   int seeds_perflowerShoot;
   double Initial_Density;
   double LeavesProduced;
   double Pl;
   double init_y;
   double init_XNW;
   double check_b_adult___fix;
   double RHIZOME_DETRITUS;
   double LEAFDETRITUS;
   double Gr;
   double Gl;
   diffs RHIZOME_DETRITUS_extras;
```

```
double Decomp;
double Rhizome_Length;
double My_birthday;
diffs Y_extras;
double Y;
diffs XNW_extras;
double XNW;
double Move_YAPEXNW;
double Move_XNWAPEXNW;
double My_Birthplace[3];
double MyCoords[3];
double age_of__youngest_node;
diffs NODES_extras;
diffs GROWING_INTERNODE___extras;
double GROWING_INTERNODE;
double NODES;
double In;
diffs OLD__NODES_extras;
double Io;
double b_A;
double Canopy_Height__for_Light;
diffs LEAFBUNDLE_extras;
double LEAFBUNDLE;
diffs OLDESTLEAF_extras;
double Ol;
double OLD__NODES;
double Nutrient__Canopy_Factor;
double malb;
diffs ROOTS___RHIZOMES_extras;
double Gi;
double ROOTS___RHIZOMES;
double OLDESTLEAF;
double BRANCH;
diffs BRANCH_extras;
double BRANCH_0;
BOOLEAN Death;
double Death_0;
diffs LEAFDETRITUS_extras;
double Sl;
double maol;
int my_number__of_nodes;
int one;
double average_internode_length;
double youngest__birthday;
double Canopy_Height;
BOOLEAN Eve;
int Eve_Status;
int My_Mother;
int Me;
double Internode_Length;
double Plastochrone_Interval;
double Di__branch;
double Timed_Square_Meter_Density;
double DIRECTION;
double Squared_Distance_to_rcm;
```

```
double Square_Meter__Density;
double initial_ch;
double randomflower;
double b_S;
double b;
double GI_A;
double GI_S;
double delay_death;
int flower_time;
double Initial___Nodes;
double Add_a__node_;
diffs Add_a__node__extras;
double Add_a__node__0;
int Node_births_and_lengthscount;
int Branching_Rate_Fix_in_progenitor;
double _array__for_Pl_last[1000];
double _array__for_Pl[1000];
int ptr_for_Pl;
int ptw_for_Pl_last;
int ptw_for_Pl;
double init_y_last;
double init_y_at_phase;
double init_XNW_last;
double init_XNW_at_phase;
double Rhizome_Length_sum;
double My_birthday_at_phase;
double My_Birthplace_in_progenitor[3];
double MyCoords_last[3];
double MyCoords_at_phase_0;
double MyCoords_at_phase;
double age_of__youngest_node_least;
double _array__for_In_last[1000];
double _array__for_In[1000];
int ptr_for_In;
int ptw_for_In_last;
int ptw_for_In;
double Canopy_Height__for_Light_at_phase;
double _array__for_Gi_last[1000];
double _array__for_Gi[1000];
int ptr_for_Gi;
int ptw_for_Gi_last;
int ptw_for_Gi;
double _array__for_BRANCH_0_0_last[1000];
double _array__for_BRANCH_0_0[1000];
int ptr_for_BRANCH_0_0;
int ptw_for_BRANCH_0_0_last;
int ptw_for_BRANCH_0_0;
double _array__for_BRANCH_0_last[1000];
double _array__for_BRANCH_0[1000];
int ptr_for_BRANCH_0;
int ptw_for_BRANCH_0_last;
int ptw_for_BRANCH_0;
int my_number__of_nodes_count;
double average_internode_length_sum;
double youngest__birthday_greatest;
```

```
    int My_Mother_at_phase;
    int My_Mother_at_phase_in_progenitor;
    double _array__for_Internode_Length_last[1000];
    double _array__for_Internode_Length[1000];
    int ptr_for_Internode_Length;
    int ptw_for_Internode_Length_last;
    int ptw_for_Internode_Length;
    double _array__for_Plastochrone_Interval_0_last[1000];
    double _array__for_Plastochrone_Interval_0[1000];
    int ptr_for_Plastochrone_Interval_0;
    int ptw_for_Plastochrone_Interval_0_last;
    int ptw_for_Plastochrone_Interval_0;
    double _array__for_Plastochrone_Interval_last[1000];
    double _array__for_Plastochrone_Interval[1000];
    int ptr_for_Plastochrone_Interval;
    int ptw_for_Plastochrone_Interval_last;
    int ptw_for_Plastochrone_Interval;
    double Di__branch_in_progenitor;
    int Timed_Square_Meter_Density_count;
    double _array__for_Square_Meter__Density_last[1000];
    double _array__for_Square_Meter__Density[1000];
    double ptr_for_Square_Meter__Density;
    int ptw_for_Square_Meter__Density_last;
    int ptw_for_Square_Meter__Density;
    double randomflower_at_phase;
    int _array__for_delay_death_last[1000];
    int _array__for_delay_death[1000];
    int ptr_for_delay_death;
    int ptw_for_delay_death_last;
    int ptw_for_delay_death;
    double _array__for_Add_a__node__0_last[1000];
    double _array__for_Add_a__node__0[1000];
    int ptr_for_Add_a__node__0;
    int ptw_for_Add_a__node__0_last;
    int ptw_for_Add_a__node__0;
void* get_pointer (int id, int** dims) {
    switch (id) {
        case 1:
        return(&(next));
        break; // end(case,1)
        case 2:
        return(&(instanceid[step_list(dims, 2)]));
        break; // end(case,2)
        case 3:
        return(&(new_instance));
        break; // end(case,3)
        case 4:
        return(&(Specific_Growth_Rate));
        break; // end(case,4)
        case 5:
        return(&(Node_births_and_lengths));
        break; // end(case,5)
        case 6:
        return(&(Adulthood));
        break; // end(case,6)
```

```
case 7:
return(&(Flower));
break; // end(case,7)
case 8:
return(&(Branching_Rate_Fix));
break; // end(case,8)
case 9:
return(&(Negative_Root_biomass));
break; // end(case,9)
case 10:
return(&(Negative_Root_biomass_0));
break; // end(case,10)
case 11:
return(&(countflowers));
break; // end(case,11)
case 12:
return(&(adult_status));
break; // end(case,12)
case 13:
return(&(LeavesProduced_extras));
break; // end(case,13)
case 14:
return(&(Me_0));
break; // end(case,14)
case 15:
return(&(seeds_perflowerShoot));
break; // end(case,15)
case 16:
return(&(Initial_Density));
break; // end(case,16)
case 17:
return(&(LeavesProduced));
break; // end(case,17)
case 18:
return(&(Pl));
break; // end(case,18)
case 19:
return(&(init_y));
break; // end(case,19)
case 20:
return(&(init_XNW));
break; // end(case,20)
case 21:
return(&(check_b_adult___fix));
break; // end(case,21)
case 22:
return(&(RHIZOME_DETRITUS));
break; // end(case,22)
case 23:
return(&(LEAFDETRITUS));
break; // end(case,23)
case 24:
return(&(Gr));
break; // end(case,24)
case 25:
```

```
return(&(Gl));
break; // end(case,25)
case 26:
return(&(RHIZOME_DETRITUS_extras));
break; // end(case,26)
case 27:
return(&(Decomp));
break; // end(case,27)
case 28:
return(&(Rhizome_Length));
break; // end(case,28)
case 29:
return(&(My_birthday));
break; // end(case,29)
case 30:
return(&(Y_extras));
break; // end(case,30)
case 31:
return(&(Y));
break; // end(case,31)
case 32:
return(&(XNW_extras));
break; // end(case,32)
case 33:
return(&(XNW));
break; // end(case,33)
case 34:
return(&(Move_YAPEXNW));
break; // end(case,34)
case 35:
return(&(Move_XNWAPEXNW));
break; // end(case,35)
case 36:
return(&(My_Birthplace[step_list(dims, 2)]));
break; // end(case,36)
case 37:
return(&(MyCoords[step_list(dims, 2)]));
break; // end(case,37)
case 38:
return(&(age_of__youngest_node));
break; // end(case,38)
case 39:
return(&(NODES_extras));
break; // end(case,39)
case 40:
return(&(GROWING_INTERNODE___extras));
break; // end(case,40)
case 41:
return(&(GROWING_INTERNODE));
break; // end(case,41)
case 42:
return(&(NODES));
break; // end(case,42)
case 43:
return(&(In));
```

```
break; // end(case,43)
case 44:
return(&(OLD__NODES_extras));
break; // end(case,44)
case 45:
return(&(Io));
break; // end(case,45)
case 46:
return(&(b_A));
break; // end(case,46)
case 47:
return(&(Canopy_Height__for_Light));
break; // end(case,47)
case 48:
return(&(LEAFBUNDLE_extras));
break; // end(case,48)
case 49:
return(&(LEAFBUNDLE));
break; // end(case,49)
case 50:
return(&(OLDESTLEAF_extras));
break; // end(case,50)
case 51:
return(&(Ol));
break; // end(case,51)
case 52:
return(&(OLD__NODES));
break; // end(case,52)
case 53:
return(&(Nutrient__Canopy_Factor));
break; // end(case,53)
case 54:
return(&(malb));
break; // end(case,54)
case 55:
return(&(ROOTS___RHIZOMES_extras));
break; // end(case,55)
case 56:
return(&(Gi));
break; // end(case,56)
case 57:
return(&(ROOTS___RHIZOMES));
break; // end(case,57)
case 58:
return(&(OLDESTLEAF));
break; // end(case,58)
case 59:
return(&(BRANCH));
break; // end(case,59)
case 60:
return(&(BRANCH_extras));
break; // end(case,60)
case 61:
return(&(BRANCH_0));
break; // end(case,61)
```

```
case 62:
return(&(Death));
break; // end(case,62)
case 63:
return(&(Death_0));
break; // end(case,63)
case 64:
return(&(LEAFDETRITUS_extras));
break; // end(case,64)
case 65:
return(&(Sl));
break; // end(case,65)
case 66:
return(&(maol));
break; // end(case,66)
case 67:
return(&(my_number__of_nodes));
break; // end(case,67)
case 68:
return(&(one));
break; // end(case,68)
case 69:
return(&(average_internode_length));
break; // end(case,69)
case 70:
return(&(youngest__birthday));
break; // end(case,70)
case 71:
return(&(Canopy_Height));
break; // end(case,71)
case 72:
return(&(Eve));
break; // end(case,72)
case 73:
return(&(Eve_Status));
break; // end(case,73)
case 74:
return(&(My_Mother));
break; // end(case,74)
case 75:
return(&(Me));
break; // end(case,75)
case 76:
return(&(Internode_Length));
break; // end(case,76)
case 77:
return(&(Plastochrone_Interval));
break; // end(case,77)
case 78:
return(&(Di__branch));
break; // end(case,78)
case 79:
return(&(Timed_Square_Meter_Density));
break; // end(case,79)
case 80:
```

```
return(&(DIRECTION));
break; // end(case,80)
case 81:
return(&(Squared_Distance_to_rcm));
break; // end(case,81)
case 82:
return(&(Square_Meter__Density));
break; // end(case,82)
case 83:
return(&(initial_ch));
break; // end(case,83)
case 84:
return(&(randomflower));
break; // end(case,84)
case 85:
return(&(b_S));
break; // end(case,85)
case 86:
return(&(b));
break; // end(case,86)
case 87:
return(&(GI_A));
break; // end(case,87)
case 88:
return(&(GI_S));
break; // end(case,88)
case 89:
return(&(delay_death));
break; // end(case,89)
case 90:
return(&(flower_time));
break; // end(case,90)
case 91:
return(&(Initial___Nodes));
break; // end(case,91)
case 92:
return(&(Add_a__node_));
break; // end(case,92)
case 93:
return(&(Add_a__node__extras));
break; // end(case,93)
case 94:
return(&(Add_a__node__0));
break; // end(case,94)
case 95:
return(&(Node_births_and_lengthscount));
break; // end(case,95)
case 96:
return(&(Branching_Rate_Fix_in_progenitor));
break; // end(case,96)
case 97:
return(&(_array__for_Pl_last[step_list(dims, 2)]));
break; // end(case,97)
case 98:
return(&(_array__for_Pl[step_list(dims, 2)]));
```

```
break; // end(case,98)
case 99:
return(&(ptr_for_Pl));
break; // end(case,99)
case 100:
return(&(ptw_for_Pl_last));
break; // end(case,100)
case 101:
return(&(ptw_for_Pl));
break; // end(case,101)
case 102:
return(&(init_y_last));
break; // end(case,102)
case 103:
return(&(init_y_at_phase));
break; // end(case,103)
case 104:
return(&(init_XNW_last));
break; // end(case,104)
case 105:
return(&(init_XNW_at_phase));
break; // end(case,105)
case 106:
return(&(Rhizome_Length_sum));
break; // end(case,106)
case 107:
return(&(My_birthday_at_phase));
break; // end(case,107)
case 108:
return(&(My_Birthplace_in_progenitor[step_list(dims, 2)]));
break; // end(case,108)
case 109:
return(&(MyCoords_last[step_list(dims, 2)]));
break; // end(case,109)
case 110:
return(&(MyCoords_at_phase_0));
break; // end(case,110)
case 111:
return(&(MyCoords_at_phase));
break; // end(case,111)
case 112:
return(&(age_of__youngest_node_least));
break; // end(case,112)
case 113:
return(&(_array__for_In_last[step_list(dims, 2)]));
break; // end(case,113)
case 114:
return(&(_array__for_In[step_list(dims, 2)]));
break; // end(case,114)
case 115:
return(&(ptr_for_In));
break; // end(case,115)
case 116:
return(&(ptw_for_In_last));
break; // end(case,116)
```

```
case 117:
return(&(ptw_for_In));
break; // end(case,117)
case 118:
return(&(Canopy_Height__for_Light_at_phase));
break; // end(case,118)
case 119:
return(&(_array__for_Gi_last[step_list(dims, 2)]));
break; // end(case,119)
case 120:
return(&(_array__for_Gi[step_list(dims, 2)]));
break; // end(case,120)
case 121:
return(&(ptr_for_Gi));
break; // end(case,121)
case 122:
return(&(ptw_for_Gi_last));
break; // end(case,122)
case 123:
return(&(ptw_for_Gi));
break; // end(case,123)
case 124:
return(&(_array__for_BRANCH_0_0_last[step_list(dims, 2)]));
break; // end(case,124)
case 125:
return(&(_array__for_BRANCH_0_0[step_list(dims, 2)]));
break; // end(case,125)
case 126:
return(&(ptr_for_BRANCH_0_0));
break; // end(case,126)
case 127:
return(&(ptw_for_BRANCH_0_0_last));
break; // end(case,127)
case 128:
return(&(ptw_for_BRANCH_0_0));
break; // end(case,128)
case 129:
return(&(_array__for_BRANCH_0_last[step_list(dims, 2)]));
break; // end(case,129)
case 130:
return(&(_array__for_BRANCH_0[step_list(dims, 2)]));
break; // end(case,130)
case 131:
return(&(ptr_for_BRANCH_0));
break; // end(case,131)
case 132:
return(&(ptw_for_BRANCH_0_last));
break; // end(case,132)
case 133:
return(&(ptw_for_BRANCH_0));
break; // end(case,133)
case 134:
return(&(my_number__of_nodes_count));
break; // end(case,134)
case 135:
```

```
return(&(average_internode_length_sum));
break; // end(case,135)
case 136:
return(&(youngest__birthday_greatest));
break; // end(case,136)
case 137:
return(&(My_Mother_at_phase));
break; // end(case,137)
case 138:
return(&(My_Mother_at_phase_in_progenitor));
break; // end(case,138)
case 139:
return(&(_array__for_Internode_Length_last[step_list(dims, 2)]));
break; // end(case,139)
case 140:
return(&(_array__for_Internode_Length[step_list(dims, 2)]));
break; // end(case,140)
case 141:
return(&(ptr_for_Internode_Length));
break; // end(case,141)
case 142:
return(&(ptw_for_Internode_Length_last));
break; // end(case,142)
case 143:
return(&(ptw_for_Internode_Length));
break; // end(case,143)
case 144:
return(&(_array__for_Plastochrone_Interval_0_last[step_list(dims, 2)]));
break; // end(case,144)
case 145:
return(&(_array__for_Plastochrone_Interval_0[step_list(dims, 2)]));
break; // end(case,145)
case 146:
return(&(ptr_for_Plastochrone_Interval_0));
break; // end(case,146)
case 147:
return(&(ptw_for_Plastochrone_Interval_0_last));
break; // end(case,147)
case 148:
return(&(ptw_for_Plastochrone_Interval_0));
break; // end(case,148)
case 149:
return(&(_array__for_Plastochrone_Interval_last[step_list(dims, 2)]));
break; // end(case,149)
case 150:
return(&(_array__for_Plastochrone_Interval[step_list(dims, 2)]));
break; // end(case,150)
case 151:
return(&(ptr_for_Plastochrone_Interval));
break; // end(case,151)
case 152:
return(&(ptw_for_Plastochrone_Interval_last));
break; // end(case,152)
case 153:
return(&(ptw_for_Plastochrone_Interval));
```

```
break; // end(case,153)
case 154:
return(&(Di__branch_in_progenitor));
break; // end(case,154)
case 155:
return(&(Timed_Square_Meter_Density_count));
break; // end(case,155)
case 156:
return(&(_array__for_Square_Meter__Density_last[step_list(dims, 2)]));
break; // end(case,156)
case 157:
return(&(_array__for_Square_Meter__Density[step_list(dims, 2)]));
break; // end(case,157)
case 158:
return(&(ptr_for_Square_Meter__Density));
break; // end(case,158)
case 159:
return(&(ptw_for_Square_Meter__Density_last));
break; // end(case,159)
case 160:
return(&(ptw_for_Square_Meter__Density));
break; // end(case,160)
case 161:
return(&(randomflower_at_phase));
break; // end(case,161)
case 162:
return(&(_array__for_delay_death_last[step_list(dims, 2)]));
break; // end(case,162)
case 163:
return(&(_array__for_delay_death[step_list(dims, 2)]));
break; // end(case,163)
case 164:
return(&(ptr_for_delay_death));
break; // end(case,164)
case 165:
return(&(ptw_for_delay_death_last));
break; // end(case,165)
case 166:
return(&(ptw_for_delay_death));
break; // end(case,166)
case 167:
return(&(_array__for_Add_a__node__0_last[step_list(dims, 2)]));
break; // end(case,167)
case 168:
return(&(_array__for_Add_a__node__0[step_list(dims, 2)]));
break; // end(case,168)
case 169:
return(&(ptr_for_Add_a__node__0));
break; // end(case,169)
case 170:
return(&(ptw_for_Add_a__node__0_last));
break; // end(case,170)
case 171:
return(&(ptw_for_Add_a__node__0));
break; // end(case,171)
```

```
    }; /* end(switch,id) */
    return(NULL);
}; /* end(procedure,get_pointer) */

}; /* end(class,Ramettype) */

class Neighbourstype : public submodeltype {
public:
    Neighbourstype () {
    }; /* end(procedure,structor) */
    ~Neighbourstype () {
    }; /* end(procedure,structor) */
    void* baseptrs[2];
    Neighbourstype* next;
    int instanceid[2];
    BOOLEAN new_instance;
    BOOLEAN cond1;
    int index_0;
void* get_pointer (int id, int** dims) {
    switch (id) {
        case 1:
        return(&(next));
        break; // end(case,1)
        case 2:
        return(&(instanceid[step_list(dims, 2)]));
        break; // end(case,2)
        case 3:
        return(&(new_instance));
        break; // end(case,3)
        case 4:
        return(&(cond1));
        break; // end(case,4)
        case 5:
        return(&(index_0));
        break; // end(case,5)
    }; /* end(switch,id) */
    return(NULL);
}; /* end(procedure,get_pointer) */

}; /* end(class,Neighbourstype) */

class Virtual_Eelgrass_Meadow_v_2_VAntype : public submodeltype {
public:
    Virtual_Eelgrass_Meadow_v_2_VAntype () {
        Eve_Plastochrone_Intervals = 0;
        Ramet = 0;
        Neighbours = 0;
    }; /* end(procedure,structor) */
    ~Virtual_Eelgrass_Meadow_v_2_VAntype () {
        delete_list(Eve_Plastochrone_Intervals);
        delete_list(Ramet);
        delete_list(Neighbours);
    }; /* end(procedure,structor) */
    Eve_Plastochrone_Intervalstype *Eve_Plastochrone_Intervals;
    EveStatstype EveStats;
```

```
Ramettype *Ramet;
Neighbourstype *Neighbours;
int countME;
int sum_adults;
double sum_Roots;
double NH_GB_TEMP;
double NH_GB_PAR;
double VA_HIB_TEMP;
double abovebelow_ratio;
double NC_NERRS_TEMP;
double South_Bay_2013_TEMP;
double VA_HIB_PAR;
double NC_NERRS_PAR;
double Average_Iz;
double averageIleaf;
double avgkcanopy;
int Total_Branches_of_Eves;
double NC1_NC2Comb_TEMP;
int sum_Eve;
double average_Eve_Branching;
double shoot_density;
int Number_of__Shoots;
double sum_LEAFbundle;
double total_biomass;
double Total_Rhizome__Length;
double Average_Internode__Length;
double Greatest_Y;
double Greatest_XNW;
double Least_XNW;
double Least_Y;
double averageLL;
double Average_Density;
double rcmXNWY[2];
double Radius_of_Gyration;
int sumflowers;
int sum_seeds;
double Seeds_that_will_Germinate__Seed_Bank_;
double Set_Initial_Density;
double delay__seeds;
double BINTZ_MESO_PAR;
double BINTZ_MESO_TEMP;
double sum_BRANCH;
int how_many_adults_true;
double Germinated_Seeds;
diffs Germinated_Seeds_extras;
double Germinated_Seeds_0;
double Initial_germinated_Seedlings;
double Adult__Initialization;
int Rametcount;
int countME_count;
int sum_adults_sum;
double sum_Roots_sum;
int Average_Iz_count;
double Average_Iz_sum;
int averageIleaf_count;
```

```
    double averageIleaf_sum;
    int avgkcanopy_count;
    double avgkcanopy_sum;
    int Total_Branches_of_Eves_sum;
    int sum_Eve_sum;
    int Number_of__Shoots_count;
    double sum_LEAFbundle_sum;
    double Total_Rhizome__Length_sum;
    int Average_Internode__Length_count;
    double Average_Internode__Length_sum;
    double Greatest_Y_last;
    double Greatest_Y_last_greatest;
    double Greatest_XNW_last;
    double Greatest_XNW_last_greatest;
    double Least_XNW_last;
    double Least_XNW_last_least;
    double Least_Y_last;
    double Least_Y_last_least;
    int averageLL_count;
    double averageLL_sum;
    int Average_Density_count;
    double Average_Density_sum;
    int rcmXNWY_count_0;
    double rcmXNWY_sum_0;
    int rcmXNWY_count;
    double rcmXNWY_sum;
    int Radius_of_Gyration_count;
    double Radius_of_Gyration_sum;
    int sumflowers_sum;
    int sum_seeds_sum;
    double _array__for_delay__seeds_last[1000];
    double _array__for_delay__seeds[1000];
    int ptr_for_delay__seeds;
    int ptw_for_delay__seeds_last;
    int ptw_for_delay__seeds;
    double sum_BRANCH_sum;
    int how_many_adults_true_sum;
  void* get_pointer (int id, int** dims) {
    switch (id) {
       case 1:
       return(&(Eve_Plastochrone_Intervals));
       break; // end(case,1)
       case 2:
       return(&(EveStats));
       break; // end(case,2)
       case 3:
       return(&(Ramet));
       break; // end(case,3)
       case 4:
       return(&(Neighbours));
       break; // end(case,4)
       case 5:
       return(&(countME));
       break; // end(case,5)
       case 6:
```

```
return(&(sum_adults));
break; // end(case,6)
case 7:
return(&(sum_Roots));
break; // end(case,7)
case 8:
return(&(NH_GB_TEMP));
break; // end(case,8)
case 9:
return(&(NH_GB_PAR));
break; // end(case,9)
case 10:
return(&(VA_HIB_TEMP));
break; // end(case,10)
case 11:
return(&(abovebelow_ratio));
break; // end(case,11)
case 12:
return(&(NC_NERRS_TEMP));
break; // end(case,12)
case 13:
return(&(South_Bay_2013_TEMP));
break; // end(case,13)
case 14:
return(&(VA_HIB_PAR));
break; // end(case,14)
case 15:
return(&(NC_NERRS_PAR));
break; // end(case,15)
case 16:
return(&(Average_Iz));
break; // end(case,16)
case 17:
return(&(averageIleaf));
break; // end(case,17)
case 18:
return(&(avgkcanopy));
break; // end(case,18)
case 19:
return(&(Total_Branches_of_Eves));
break; // end(case,19)
case 20:
return(&(NC1_NC2Comb_TEMP));
break; // end(case,20)
case 21:
return(&(sum_Eve));
break; // end(case,21)
case 22:
return(&(average_Eve_Branching));
break; // end(case,22)
case 23:
return(&(shoot_density));
break; // end(case,23)
case 24:
return(&(Number_of__Shoots));
```

```
break; // end(case,24)
case 25:
return(&(sum_LEAFbundle));
break; // end(case,25)
case 26:
return(&(total_biomass));
break; // end(case,26)
case 27:
return(&(Total_Rhizome__Length));
break; // end(case,27)
case 28:
return(&(Average_Internode__Length));
break; // end(case,28)
case 29:
return(&(Greatest_Y));
break; // end(case,29)
case 30:
return(&(Greatest_XNW));
break; // end(case,30)
case 31:
return(&(Least_XNW));
break; // end(case,31)
case 32:
return(&(Least_Y));
break; // end(case,32)
case 33:
return(&(averageLL));
break; // end(case,33)
case 34:
return(&(Average_Density));
break; // end(case,34)
case 35:
return(&(rcmXNWY[step_list(dims, 2)]));
break; // end(case,35)
case 36:
return(&(Radius_of_Gyration));
break; // end(case,36)
case 37:
return(&(sumflowers));
break; // end(case,37)
case 38:
return(&(sum_seeds));
break; // end(case,38)
case 39:
return(&(Seeds_that_will_Germinate__Seed_Bank_));
break; // end(case,39)
case 40:
return(&(Set_Initial_Density));
break; // end(case,40)
case 41:
return(&(delay__seeds));
break; // end(case,41)
case 42:
return(&(BINTZ_MESO_PAR));
break; // end(case,42)
```

```
case 43:
return(&(BINTZ_MESO_TEMP));
break; // end(case,43)
case 44:
return(&(sum_BRANCH));
break; // end(case,44)
case 45:
return(&(how_many_adults_true));
break; // end(case,45)
case 46:
return(&(Germinated_Seeds));
break; // end(case,46)
case 47:
return(&(Germinated_Seeds_extras));
break; // end(case,47)
case 48:
return(&(Germinated_Seeds_0));
break; // end(case,48)
case 49:
return(&(Initial_germinated_Seedlings));
break; // end(case,49)
case 50:
return(&(Adult__Initialization));
break; // end(case,50)
case 51:
return(&(Rametcount));
break; // end(case,51)
case 52:
return(&(countME_count));
break; // end(case,52)
case 53:
return(&(sum_adults_sum));
break; // end(case,53)
case 54:
return(&(sum_Roots_sum));
break; // end(case,54)
case 55:
return(&(Average_Iz_count));
break; // end(case,55)
case 56:
return(&(Average_Iz_sum));
break; // end(case,56)
case 57:
return(&(averageIleaf_count));
break; // end(case,57)
case 58:
return(&(averageIleaf_sum));
break; // end(case,58)
case 59:
return(&(avgkcanopy_count));
break; // end(case,59)
case 60:
return(&(avgkcanopy_sum));
break; // end(case,60)
case 61:
```

```
return(&(Total_Branches_of_Eves_sum));
break; // end(case,61)
case 62:
return(&(sum_Eve_sum));
break; // end(case,62)
case 63:
return(&(Number_of__Shoots_count));
break; // end(case,63)
case 64:
return(&(sum_LEAFbundle_sum));
break; // end(case,64)
case 65:
return(&(Total_Rhizome__Length_sum));
break; // end(case,65)
case 66:
return(&(Average_Internode__Length_count));
break; // end(case,66)
case 67:
return(&(Average_Internode__Length_sum));
break; // end(case,67)
case 68:
return(&(Greatest_Y_last));
break; // end(case,68)
case 69:
return(&(Greatest_Y_last_greatest));
break; // end(case,69)
case 70:
return(&(Greatest_XNW_last));
break; // end(case,70)
case 71:
return(&(Greatest_XNW_last_greatest));
break; // end(case,71)
case 72:
return(&(Least_XNW_last));
break; // end(case,72)
case 73:
return(&(Least_XNW_last_least));
break; // end(case,73)
case 74:
return(&(Least_Y_last));
break; // end(case,74)
case 75:
return(&(Least_Y_last_least));
break; // end(case,75)
case 76:
return(&(averageLL_count));
break; // end(case,76)
case 77:
return(&(averageLL_sum));
break; // end(case,77)
case 78:
return(&(Average_Density_count));
break; // end(case,78)
case 79:
return(&(Average_Density_sum));
```

```
      break; // end(case,79)
      case 80:
      return(&(rcmXNWY_count_0));
      break; // end(case,80)
      case 81:
      return(&(rcmXNWY_sum_0));
      break; // end(case,81)
      case 82:
      return(&(rcmXNWY_count));
      break; // end(case,82)
      case 83:
      return(&(rcmXNWY_sum));
      break; // end(case,83)
      case 84:
      return(&(Radius_of_Gyration_count));
      break; // end(case,84)
      case 85:
      return(&(Radius_of_Gyration_sum));
      break; // end(case,85)
      case 86:
      return(&(sumflowers_sum));
      break; // end(case,86)
      case 87:
      return(&(sum_seeds_sum));
      break; // end(case,87)
      case 88:
      return(&(_array__for_delay__seeds_last[step_list(dims, 2)]));
      break; // end(case,88)
      case 89:
      return(&(_array__for_delay__seeds[step_list(dims, 2)]));
      break; // end(case,89)
      case 90:
      return(&(ptr_for_delay__seeds));
      break; // end(case,90)
      case 91:
      return(&(ptw_for_delay__seeds_last));
      break; // end(case,91)
      case 92:
      return(&(ptw_for_delay__seeds));
      break; // end(case,92)
      case 93:
      return(&(sum_BRANCH_sum));
      break; // end(case,93)
      case 94:
      return(&(how_many_adults_true_sum));
      break; // end(case,94)
   }; /* end(switch,id) */
   return(NULL);
}; /* end(procedure,get_pointer) */

}; /* end(class,Virtual_Eelgrass_Meadow_v_2_VAntype) */

class AME_model : public InstanceOfModel {
public:
   AME_model () {
```

```
   }; /* end(procedure,structor) */
   ~AME_model () {
   }; /* end(procedure,structor) */
   Virtual_Eelgrass_Meadow_v_2_VAntype Virtual_Eelgrass_Meadow_v_2_VAn;
void* get_pointer (int id, int** dims) {
   switch (id) {
      case 1:
      return(&(Virtual_Eelgrass_Meadow_v_2_VAn));
      break; // end(case,1)
   }; /* end(switch,id) */
   return(NULL);
}; /* end(procedure,get_pointer) */

/* EVALUATION PROCEDURE DECLARATION */

void advancemodel (int phase) {

/* CONSTANT DECLARATIONS */

/* global this
 */
/* global array_9
 */
/* global array_10
 */
/* global array_11
 */
/* global array
 */
/* global array_0
 */
/* global array_1
 */
/* global array_2
 */
/* global array_3
 */
/* global array_4
 */
/* global array_5
 */
/* global array_6
 */
/* global array_7
 */
/* global array_8
 */

/* STRUCTURE TYPE DECLARATIONS */


   /* UPDATE FUNCTION VALUES */


}; /* end(procedure,advancemodel) */
```

```
/* EVALUATION PROCEDURE DECLARATION */

void updatemodel (int phase) {

/* CONSTANT DECLARATIONS */

/* global this
 */
/* global array_9
 */
/* global array_10
 */
/* global array_11
 */
/* global array
 */
/* global array_0
 */
/* global array_1
 */
/* global array_2
 */
/* global array_3
 */
/* global array_4
 */
/* global array_5
 */
/* global array_6
 */
/* global array_7
 */
/* global array_8
 */

/* STRUCTURE TYPE DECLARATIONS */


   /* UPDATE FUNCTION VALUES */

   if (2>=phase) {
      Virtual_Eelgrass_Meadow_v_2_VAntype* Virtual_Eelgrass_Meadow_v_2_VAnpointer;
      Virtual_Eelgrass_Meadow_v_2_VAnpointer = &(this->Virtual_Eelgrass_Meadow_v_2_VAn);
      Virtual_Eelgrass_Meadow_v_2_VAnpointer->Germinated_Seeds =
Virtual_Eelgrass_Meadow_v_2_VAnpointer-
>Germinated_Seeds+stage_incr(&(Virtual_Eelgrass_Meadow_v_2_VAnpointer-
>Germinated_Seeds_extras), 2, Virtual_Eelgrass_Meadow_v_2_VAnpointer->Germinated_Seeds_0,
100, 345);
      Ramettype* Rametpointer;
      Rametpointer = Virtual_Eelgrass_Meadow_v_2_VAnpointer->Ramet;
      while ( Rametpointer != 0 ) {
         abort_check(this);
         Ramettype* Rametprogen;
         Rametprogen = (Ramettype*)Rametpointer->baseptrs[0];
```

```
        Specific_Growth_Ratetype* Specific_Growth_Ratepointer;
        Specific_Growth_Ratepointer = &(Rametpointer->Specific_Growth_Rate);
        Temperaturetype* Temperaturepointer;
        Temperaturepointer = &(Specific_Growth_Ratepointer->Temperature);
        Temperaturepointer->daysabove_20box = Temperaturepointer-
>daysabove_20box+stage_incr(&(Temperaturepointer->daysabove_20box_extras), 2,
Temperaturepointer->flow1, 100, 150);
        Temperaturepointer->FDegreeDays = Temperaturepointer-
>FDegreeDays+stage_incr(&(Temperaturepointer->FDegreeDays_extras), 2, Temperaturepointer-
>FDays, 100, 154);
        Temperaturepointer->GDegreeDays = Temperaturepointer-
>GDegreeDays+stage_incr(&(Temperaturepointer->GDegreeDays_extras), 2, Temperaturepointer-
>GDays, 100, 157);
        Rametpointer->LeavesProduced = Rametpointer-
>LeavesProduced+stage_incr(&(Rametpointer->LeavesProduced_extras), 2, Rametpointer->Pl, 100,
219);
        Rametpointer->RHIZOME_DETRITUS = Rametpointer-
>RHIZOME_DETRITUS+stage_incr(&(Rametpointer->RHIZOME_DETRITUS_extras), 2,
Rametpointer->Decomp, 100, 224);
        Rametpointer->Y = Rametpointer->Y+stage_incr(&(Rametpointer->Y_extras), 2,
Rametpointer->Move_YAPEXNW, 100, 233);
        Rametpointer->XNW = Rametpointer->XNW+stage_incr(&(Rametpointer->XNW_extras), 2,
Rametpointer->Move_XNWAPEXNW, 100, 235);
        Rametpointer->OLD__NODES = Rametpointer-
>OLD__NODES+stage_incr(&(Rametpointer->OLD__NODES_extras), 2, Rametpointer->Io, 100,
254);
        Rametpointer->BRANCH = Rametpointer->BRANCH+stage_incr(&(Rametpointer-
>BRANCH_extras), 2, Rametpointer->BRANCH_0, 100, 261);
        Rametpointer->LEAFDETRITUS = Rametpointer-
>LEAFDETRITUS+stage_incr(&(Rametpointer->LEAFDETRITUS_extras), 2, Rametpointer->Sl,
100, 225);
        Rametpointer->DIRECTION = Rametpointer->DIRECTION;
        Rametpointer->Add_a__node_ = Rametpointer->Add_a__node_+stage_incr(&(Rametpointer-
>Add_a__node__extras), 2, Rametpointer->Add_a__node__0, 100, 294);
        Node_births_and_lengthstype* Node_births_and_lengthspointer;
        Node_births_and_lengthspointer = Rametpointer->Node_births_and_lengths;
        while ( Node_births_and_lengthspointer != 0 ) {
          abort_check(this);
          Node_births_and_lengthstype* Node_births_and_lengthsprogen;
          Node_births_and_lengthsprogen =
(Node_births_and_lengthstype*)Node_births_and_lengthspointer->baseptrs[0];
          Node_births_and_lengthspointer->NodeXNW = Node_births_and_lengthspointer-
>NodeXNW;
          Node_births_and_lengthspointer->NodeY = Node_births_and_lengthspointer->NodeY;
          Node_births_and_lengthspointer = Node_births_and_lengthspointer->next;
        }; /* end(while,Node_births_and_lengthspointer) */
        Rametpointer = Rametpointer->next;
      }; /* end(while,Rametpointer) */
    }; /* end(cond,2>=phase) */
    Virtual_Eelgrass_Meadow_v_2_VAntype* Virtual_Eelgrass_Meadow_v_2_VAnpointer;
    Virtual_Eelgrass_Meadow_v_2_VAnpointer = &(this->Virtual_Eelgrass_Meadow_v_2_VAn);
    Ramettype* Rametpointer;
    Rametpointer = Virtual_Eelgrass_Meadow_v_2_VAnpointer->Ramet;
    while ( Rametpointer != 0 ) {
      abort_check(this);
```

```
      Ramettype* Rametprogen;
      Rametprogen = (Ramettype*)Rametpointer->baseptrs[0];
      Rametpointer->NODES = Rametpointer->NODES+stage_incr(&(Rametpointer-
>NODES_extras), 2, Rametpointer->In+ -Rametpointer->Io, 100, 244);
      Rametpointer->GROWING_INTERNODE = Rametpointer-
>GROWING_INTERNODE+stage_incr(&(Rametpointer->GROWING_INTERNODE___extras), 2,
Rametpointer->Gr+ -Rametpointer->Gi, 100, 243);
      Rametpointer->LEAFBUNDLE = Rametpointer->LEAFBUNDLE+stage_incr(&(Rametpointer-
>LEAFBUNDLE_extras), 2, Rametpointer->Gl+ -Rametpointer->Ol, 100, 251);
      Rametpointer->OLDESTLEAF = Rametpointer->OLDESTLEAF+stage_incr(&(Rametpointer-
>OLDESTLEAF_extras), 2, Rametpointer->Ol+ -Rametpointer->Sl, 100, 260);
      Rametpointer->ROOTS___RHIZOMES = Rametpointer-
>ROOTS___RHIZOMES+stage_incr(&(Rametpointer->ROOTS___RHIZOMES_extras), 2,
Rametpointer->Gi+ -Rametpointer->Decomp, 100, 259);
      Node_births_and_lengthstype* Node_births_and_lengthspointer;
      Node_births_and_lengthspointer = Rametpointer->Node_births_and_lengths;
      while ( Node_births_and_lengthspointer != 0 ) {
         abort_check(this);
         Node_births_and_lengthstype* Node_births_and_lengthsprogen;
         Node_births_and_lengthsprogen =
(Node_births_and_lengthstype*)Node_births_and_lengthspointer->baseptrs[0];
         Node_births_and_lengthspointer->my_internode_length = Node_births_and_lengthspointer-
>my_internode_length;
         Node_births_and_lengthspointer = Node_births_and_lengthspointer->next;
      }; /* end(while,Node_births_and_lengthspointer) */
      Rametpointer = Rametpointer->next;
   }; /* end(while,Rametpointer) */

}; /* end(procedure,updatemodel) */

/* EVALUATION PROCEDURE DECLARATION */

void evalmodel (int phase) {

/* CONSTANT DECLARATIONS */

/* global this
 */
/* global array_9
 */
/* global array_10
 */
/* global array_11
 */
/* global array
 */
/* global array_0
 */
/* global array_1
 */
/* global array_2
 */
/* global array_3
 */
/* global array_4
```

```
 */
/* global array_5
 */
/* global array_6
 */
/* global array_7
 */
/* global array_8
 */

/* STRUCTURE TYPE DECLARATIONS */


   /* UPDATE FUNCTION VALUES */

   if (-2>=phase) {
      Virtual_Eelgrass_Meadow_v_2_VAntype* Virtual_Eelgrass_Meadow_v_2_VAnpointer;
      Virtual_Eelgrass_Meadow_v_2_VAnpointer = &(this->Virtual_Eelgrass_Meadow_v_2_VAn);
      Virtual_Eelgrass_Meadow_v_2_VAnpointer->Eve_Plastochrone_Intervals = 0;
      Virtual_Eelgrass_Meadow_v_2_VAnpointer->Ramet = 0;
      delete_list(Virtual_Eelgrass_Meadow_v_2_VAnpointer->Eve_Plastochrone_Intervals);
      Virtual_Eelgrass_Meadow_v_2_VAnpointer->Eve_Plastochrone_Intervals = 0;
      Virtual_Eelgrass_Meadow_v_2_VAnpointer->Neighbours = 0;
      delete_list(Virtual_Eelgrass_Meadow_v_2_VAnpointer->Neighbours);
      Virtual_Eelgrass_Meadow_v_2_VAnpointer->Neighbours = 0;
      Virtual_Eelgrass_Meadow_v_2_VAnpointer->Set_Initial_Density = 62.5;
   }; /* end(cond,-2>=phase) */
   if (0>=phase) {
      Virtual_Eelgrass_Meadow_v_2_VAntype* Virtual_Eelgrass_Meadow_v_2_VAnpointer;
      Virtual_Eelgrass_Meadow_v_2_VAnpointer = &(this->Virtual_Eelgrass_Meadow_v_2_VAn);
      Virtual_Eelgrass_Meadow_v_2_VAnpointer->Greatest_Y_last = 0;
      Virtual_Eelgrass_Meadow_v_2_VAnpointer->Greatest_XNW_last = 0;
      Virtual_Eelgrass_Meadow_v_2_VAnpointer->Least_XNW_last = 0;
      Virtual_Eelgrass_Meadow_v_2_VAnpointer->Least_Y_last = 0;
      Virtual_Eelgrass_Meadow_v_2_VAnpointer->ptw_for_delay__seeds_last = 0;
      Virtual_Eelgrass_Meadow_v_2_VAnpointer->Germinated_Seeds = ame_rand(0, 1);
      Virtual_Eelgrass_Meadow_v_2_VAnpointer->Initial_germinated_Seedlings = 0;
      Virtual_Eelgrass_Meadow_v_2_VAnpointer->Adult__Initialization = 0;
      int arraybuild0;
      for ( arraybuild0 = 1; 1000>=arraybuild0; ++arraybuild0 ) {
         Virtual_Eelgrass_Meadow_v_2_VAnpointer->_array__for_delay__seeds_last[arraybuild0-1] =
0;
      }; /* end(for,arraybuild0) */
   }; /* end(cond,0>=phase) */
   if (2>=phase) {
      Virtual_Eelgrass_Meadow_v_2_VAntype* Virtual_Eelgrass_Meadow_v_2_VAnpointer;
      Virtual_Eelgrass_Meadow_v_2_VAnpointer = &(this->Virtual_Eelgrass_Meadow_v_2_VAn);
      EveStatstype* EveStatspointer;
      EveStatspointer = &(Virtual_Eelgrass_Meadow_v_2_VAnpointer->EveStats);
      Virtual_Eelgrass_Meadow_v_2_VAnpointer->Least_Y =
Virtual_Eelgrass_Meadow_v_2_VAnpointer->Least_Y_last;
      Virtual_Eelgrass_Meadow_v_2_VAnpointer->Least_XNW =
Virtual_Eelgrass_Meadow_v_2_VAnpointer->Least_XNW_last;
      Virtual_Eelgrass_Meadow_v_2_VAnpointer->Greatest_XNW =
Virtual_Eelgrass_Meadow_v_2_VAnpointer->Greatest_XNW_last;
```

```
      Virtual_Eelgrass_Meadow_v_2_VAnpointer->Greatest_Y =
Virtual_Eelgrass_Meadow_v_2_VAnpointer->Greatest_Y_last;
      Ramettype* Rametpointer;
      Ramettype** Rametmeta;
      Rametmeta = &(Virtual_Eelgrass_Meadow_v_2_VAnpointer->Ramet);
      while ( *Rametmeta != 0 ) {
         abort_check(this);
         Rametpointer = *Rametmeta;
         Rametpointer->new_instance = 0;
         if (Rametpointer->Negative_Root_biomass || Rametpointer->Death) {
            *Rametmeta = Rametpointer->next;
            delete Rametpointer;
         } else { /* Rametpointer->Negative_Root_biomass || Rametpointer->Death */
            Rametmeta = &(Rametpointer->next);
         }; /* end(cond,Rametpointer->Negative_Root_biomass || Rametpointer->Death) */
      }; /* end(while,*Rametmeta) */
      while ( Virtual_Eelgrass_Meadow_v_2_VAnpointer->Germinated_Seeds>=1 ) {
         abort_check(this);
         Virtual_Eelgrass_Meadow_v_2_VAnpointer->Germinated_Seeds =
Virtual_Eelgrass_Meadow_v_2_VAnpointer->Germinated_Seeds-1;
         ++Virtual_Eelgrass_Meadow_v_2_VAnpointer->Rametcount;
         Rametpointer = new Ramettype;
         init_pop_member(Rametpointer, Virtual_Eelgrass_Meadow_v_2_VAnpointer->Rametcount,
345);
         *Rametmeta = Rametpointer;
         Rametmeta = &(Rametpointer->next);
      }; /* end(while,New instances) */
      Rametpointer = Virtual_Eelgrass_Meadow_v_2_VAnpointer->Ramet;
      while ( Rametpointer != 0 ) {
         abort_check(this);
         if (!Rametpointer->new_instance) {
         while ( Rametpointer->BRANCH>=1 ) {
            abort_check(this);
            Rametpointer->BRANCH = Rametpointer->BRANCH-1;
            ++Virtual_Eelgrass_Meadow_v_2_VAnpointer->Rametcount;
            *Rametmeta = new Ramettype;
            init_pop_member(*Rametmeta, Virtual_Eelgrass_Meadow_v_2_VAnpointer->Rametcount,
261);
         (*Rametmeta)->baseptrs[0] = Rametpointer;
            Rametmeta = &((*Rametmeta)->next);
         }; /* end(while,Rametpointer->BRANCH) */
         }; /* end(cond,!Rametpointer->new_instance) */
         Rametpointer = Rametpointer->next;
      }; /* end(while,Rametpointer) */
      EveStatspointer->Tagged_PI_sum_0 = 0;
      EveStatspointer->Tagged_PI_sum = 0;
      EveStatspointer->GreatestPI_greatest = -1.0e+100;
      EveStatspointer->mean_number__of_new_nodes_sum_0 = 0;
      EveStatspointer->mean_number__of_new_nodes_sum = 0;
      EveStatspointer->Greatest_New_Nodes_greatest = -1.0e+100;
      EveStatspointer->mean_new_leaves_sum_0 = 0;
      EveStatspointer->mean_new_leaves_sum = 0;
      EveStatspointer->stdevnodes_sum_1 = 0;
      EveStatspointer->stdevnodes_sum_0 = 0;
      EveStatspointer->stdevnodes_sum = 0;
```

```
  EveStatspointer->leastnodes_least = 1.0e+100;
  EveStatspointer->aboveavg_count = 0;
  EveStatspointer->aboveavg_sum = 0;
  EveStatspointer->abovestdev_sum_0 = 0;
  EveStatspointer->abovestdev_sum = 0;
  EveStatspointer->avgbelow_count = 0;
  EveStatspointer->avgbelow_sum = 0;
  EveStatspointer->belowstdev_sum_0 = 0;
  EveStatspointer->belowstdev_sum = 0;
  Virtual_Eelgrass_Meadow_v_2_VAnpointer->countME_count = 0;
  Virtual_Eelgrass_Meadow_v_2_VAnpointer->sum_adults_sum = 0;
  Virtual_Eelgrass_Meadow_v_2_VAnpointer->sum_Roots_sum = 0;
  Virtual_Eelgrass_Meadow_v_2_VAnpointer->NH_GB_TEMP = array[(int)(ts[2])-1];
  Virtual_Eelgrass_Meadow_v_2_VAnpointer->NH_GB_PAR = array_0[(int)(ts[2])-1];
  Virtual_Eelgrass_Meadow_v_2_VAnpointer->VA_HIB_TEMP = array_1[(int)(ts[2])-1];
  Virtual_Eelgrass_Meadow_v_2_VAnpointer->NC_NERRS_TEMP = array_2[(int)(ts[2])-1];
  Virtual_Eelgrass_Meadow_v_2_VAnpointer->South_Bay_2013_TEMP = array_3[(int)(ts[2])-1];
  Virtual_Eelgrass_Meadow_v_2_VAnpointer->VA_HIB_PAR = array_4[(int)(ts[2])-1];
  Virtual_Eelgrass_Meadow_v_2_VAnpointer->NC_NERRS_PAR = array_5[(int)(ts[2])-1];
  Virtual_Eelgrass_Meadow_v_2_VAnpointer->Average_Iz_count = 0;
  Virtual_Eelgrass_Meadow_v_2_VAnpointer->Average_Iz_sum = 0;
  Virtual_Eelgrass_Meadow_v_2_VAnpointer->averageIleaf_count = 0;
  Virtual_Eelgrass_Meadow_v_2_VAnpointer->averageIleaf_sum = 0;
  Virtual_Eelgrass_Meadow_v_2_VAnpointer->avgkcanopy_count = 0;
  Virtual_Eelgrass_Meadow_v_2_VAnpointer->avgkcanopy_sum = 0;
  Virtual_Eelgrass_Meadow_v_2_VAnpointer->Total_Branches_of_Eves_sum = 0;
  Virtual_Eelgrass_Meadow_v_2_VAnpointer->NC1_NC2Comb_TEMP = array_6[(int)(ts[2])-1];
  Virtual_Eelgrass_Meadow_v_2_VAnpointer->sum_Eve_sum = 0;
  Virtual_Eelgrass_Meadow_v_2_VAnpointer->Number_of__Shoots_count = 0;
  Virtual_Eelgrass_Meadow_v_2_VAnpointer->sum_LEAFbundle_sum = 0;
  Virtual_Eelgrass_Meadow_v_2_VAnpointer->Total_Rhizome__Length_sum = 0;
  Virtual_Eelgrass_Meadow_v_2_VAnpointer->Average_Internode__Length_count = 0;
  Virtual_Eelgrass_Meadow_v_2_VAnpointer->Average_Internode__Length_sum = 0;
  Virtual_Eelgrass_Meadow_v_2_VAnpointer->Greatest_Y_last_greatest = -1.0e+100;
  Virtual_Eelgrass_Meadow_v_2_VAnpointer->Greatest_XNW_last_greatest = -1.0e+100;
  Virtual_Eelgrass_Meadow_v_2_VAnpointer->Least_XNW_last_least = 1.0e+100;
  Virtual_Eelgrass_Meadow_v_2_VAnpointer->Least_Y_last_least = 1.0e+100;
  Virtual_Eelgrass_Meadow_v_2_VAnpointer->averageLL_count = 0;
  Virtual_Eelgrass_Meadow_v_2_VAnpointer->averageLL_sum = 0;
  Virtual_Eelgrass_Meadow_v_2_VAnpointer->Average_Density_count = 0;
  Virtual_Eelgrass_Meadow_v_2_VAnpointer->Average_Density_sum = 0;
  Virtual_Eelgrass_Meadow_v_2_VAnpointer->rcmXNWY_count_0 = 0;
  Virtual_Eelgrass_Meadow_v_2_VAnpointer->rcmXNWY_sum_0 = 0;
  Virtual_Eelgrass_Meadow_v_2_VAnpointer->rcmXNWY_count = 0;
  Virtual_Eelgrass_Meadow_v_2_VAnpointer->rcmXNWY_sum = 0;
  Virtual_Eelgrass_Meadow_v_2_VAnpointer->Radius_of_Gyration_count = 0;
  Virtual_Eelgrass_Meadow_v_2_VAnpointer->Radius_of_Gyration_sum = 0;
  Virtual_Eelgrass_Meadow_v_2_VAnpointer->sumflowers_sum = 0;
  Virtual_Eelgrass_Meadow_v_2_VAnpointer->BINTZ_MESO_PAR = array_7[(int)(ts[2])-1];
  Virtual_Eelgrass_Meadow_v_2_VAnpointer->BINTZ_MESO_TEMP = array_8[(int)(ts[2])-1];
  Virtual_Eelgrass_Meadow_v_2_VAnpointer->sum_BRANCH_sum = 0;
  Virtual_Eelgrass_Meadow_v_2_VAnpointer->how_many_adults_true_sum = 0;
}; /* end(cond,2>=phase) */
Virtual_Eelgrass_Meadow_v_2_VAntype* Virtual_Eelgrass_Meadow_v_2_VAnpointer;
Virtual_Eelgrass_Meadow_v_2_VAnpointer = &(this->Virtual_Eelgrass_Meadow_v_2_VAn);
```

```
   Virtual_Eelgrass_Meadow_v_2_VAnpointer->ptw_for_delay__seeds =
(Virtual_Eelgrass_Meadow_v_2_VAnpointer-
>ptw_for_delay__seeds_last==1000?1:Virtual_Eelgrass_Meadow_v_2_VAnpointer-
>ptw_for_delay__seeds_last+1);
   Virtual_Eelgrass_Meadow_v_2_VAnpointer->ptr_for_delay__seeds =
Virtual_Eelgrass_Meadow_v_2_VAnpointer->ptw_for_delay__seeds-171-
1000*(int)(floor((double)(Virtual_Eelgrass_Meadow_v_2_VAnpointer->ptw_for_delay__seeds-171-
1)/1000));
   Virtual_Eelgrass_Meadow_v_2_VAnpointer->sum_seeds_sum = 0;
   int arraybuild0;
   for ( arraybuild0 = 1; 1000>=arraybuild0; ++arraybuild0 ) {
   }; /* end(for,arraybuild0) */
   if (0>=phase) {
      Virtual_Eelgrass_Meadow_v_2_VAnpointer = &(this->Virtual_Eelgrass_Meadow_v_2_VAn);
      Ramettype* Rametpointer;
      Ramettype** Rametmeta;
      Virtual_Eelgrass_Meadow_v_2_VAnpointer->Rametcount = 0;
      Rametmeta = &(Virtual_Eelgrass_Meadow_v_2_VAnpointer->Ramet);
      Virtual_Eelgrass_Meadow_v_2_VAnpointer->Rametcount = init_pop(&(Rametmeta),
Virtual_Eelgrass_Meadow_v_2_VAnpointer->Adult__Initialization,
Virtual_Eelgrass_Meadow_v_2_VAnpointer->Rametcount, 349);
      Virtual_Eelgrass_Meadow_v_2_VAnpointer->Rametcount = init_pop(&(Rametmeta),
Virtual_Eelgrass_Meadow_v_2_VAnpointer->Initial_germinated_Seedlings,
Virtual_Eelgrass_Meadow_v_2_VAnpointer->Rametcount, 348);
      delete_list(*Rametmeta);
      *Rametmeta = 0;
   }; /* end(cond,0>=phase) */
   if (2>=phase) {
      Virtual_Eelgrass_Meadow_v_2_VAnpointer = &(this->Virtual_Eelgrass_Meadow_v_2_VAn);
      EveStatstype* EveStatspointer;
      EveStatspointer = &(Virtual_Eelgrass_Meadow_v_2_VAnpointer->EveStats);
      Virtual_Eelgrass_Meadow_v_2_VAnpointer->ptw_for_delay__seeds_last =
Virtual_Eelgrass_Meadow_v_2_VAnpointer->ptw_for_delay__seeds;
      Ramettype* Rametpointer;
      Rametpointer = Virtual_Eelgrass_Meadow_v_2_VAnpointer->Ramet;
      while ( Rametpointer != 0 ) {
         abort_check(this);
         Ramettype* Rametprogen;
         Rametprogen = (Ramettype*)Rametpointer->baseptrs[0];
         if (-2>=(2>=phase&&Rametpointer->new_instance? -2:phase)) {
            Specific_Growth_Ratetype* Specific_Growth_Ratepointer;
            Specific_Growth_Ratepointer = &(Rametpointer->Specific_Growth_Rate);
            Grass_Machinetype* Grass_Machinepointer;
            Grass_Machinepointer = &(Specific_Growth_Ratepointer->Grass_Machine);
            SGR_Seedlingtype* SGR_Seedlingpointer;
            SGR_Seedlingpointer = &(Grass_Machinepointer->SGR_Seedling);
            Temperaturetype* Temperaturepointer;
            Temperaturepointer = &(Specific_Growth_Ratepointer->Temperature);
            Physical_Settingtype* Physical_Settingpointer;
            Physical_Settingpointer = &(Specific_Growth_Ratepointer->Physical_Setting);
            SGR_Adulttype* SGR_Adultpointer;
            SGR_Adultpointer = &(Grass_Machinepointer->SGR_Adult);
            Physical_Settingpointer->water_depth_SA = 0.59999999999999998;
            Physical_Settingpointer->sediments_SA = 50;
```

```
            SGR_Adultpointer->sediment__limitation_A = (Physical_Settingpointer-
>sediments_SA<=55.4500000000000003?1:(Physical_Settingpointer-
>sediments_SA>=2000?0:13.6*pow(Physical_Settingpointer->sediments_SA,-
0.65000000000000002)));
            SGR_Seedlingpointer->sediment_limitation_S = (Physical_Settingpointer-
>sediments_SA<=1?1:(Physical_Settingpointer->sediments_SA>=2000?0:1.0239*exp(-
0.002*Physical_Settingpointer->sediments_SA)));
            Physical_Settingpointer->Nitrogen_SA = 0.0085400000000000007;
            Rametpointer->Nutrient__Canopy_Factor =
0.20599999999999999*log(Physical_Settingpointer->Nitrogen_SA)+1.7679;
            Temperaturepointer->Daysabove20 = 90;
            SGR_Seedlingpointer->umax_S = 0.029999999999999999;
            Rametpointer->Node_births_and_lengths = 0;
            Rametpointer->Me_0 = Rametpointer->instanceid[0];
            Rametpointer->Initial_Density = Virtual_Eelgrass_Meadow_v_2_VAnpointer-
>Set_Initial_Density;
            Rametpointer->one = 1;
            Rametpointer->Me = Rametpointer->instanceid[0];
         }; /* end(cond,-2>=(2>=phase&&Rametpointer->new_instance? -2:phase)) */
         if (0>=(2>=phase&&Rametpointer->new_instance? -2:phase)) {
            Specific_Growth_Ratetype* Specific_Growth_Ratepointer;
            Specific_Growth_Ratepointer = &(Rametpointer->Specific_Growth_Rate);
            Lighttype* Lightpointer;
            Lightpointer = &(Specific_Growth_Ratepointer->Light);
            Light_Adulttype* Light_Adultpointer;
            Light_Adultpointer = &(Lightpointer->Light_Adult);
            Temperaturetype* Temperaturepointer;
            Temperaturepointer = &(Specific_Growth_Ratepointer->Temperature);
            Temperaturepointer->daysabove_20box = 0;
            Temperaturepointer->FDegreeDays = 0;
            Temperaturepointer->GDegreeDays = 0;
            Light_Adultpointer->Izleaf_A_at_phase = ts[2];
            Rametpointer->Branching_Rate_Fix = (Rametprogen?Rametprogen-
>Branching_Rate_Fix_in_progenitor:0);
            Rametpointer->LeavesProduced = 0;
            Rametpointer->ptw_for_Pl_last = 0;
            Rametpointer->init_y_last = 0;
            Rametpointer->init_y_at_phase = ame_rand(0,  1.05);
            Rametpointer->init_XNW_last = 0;
            Rametpointer->init_XNW_at_phase = ame_rand(0,  1.6);
            Rametpointer->RHIZOME_DETRITUS = 0;
            Rametpointer->LEAFDETRITUS = 0;
            Rametpointer->My_birthday_at_phase = ts[2];
            Rametpointer->My_birthday = Rametpointer->My_birthday_at_phase;
            Rametpointer->ptw_for_In_last = 0;
            Rametpointer->Canopy_Height__for_Light_at_phase = ts[2];
            Rametpointer->OLD__NODES = 0;
            Rametpointer->ptw_for_Gi_last = 0;
            Rametpointer->BRANCH = ame_rand(0, 1);
            Rametpointer->ptw_for_BRANCH_0_0_last = 0;
            Rametpointer->ptw_for_BRANCH_0_last = 0;
            Rametpointer->Eve = ! (Rametpointer->channelId==261);
            Rametpointer->Eve_Status = (Rametpointer->Eve==1?1:0);
            Rametpointer->Branching_Rate_Fix_in_progenitor = Rametpointer->Eve_Status;
```

```
            Rametpointer->My_Mother_at_phase = (Rametprogen?Rametprogen-
>My_Mother_at_phase_in_progenitor:0);
            Rametpointer->My_Mother = Rametpointer->My_Mother_at_phase;
            Rametpointer->My_Mother_at_phase_in_progenitor = Rametpointer->instanceid[0];
            Rametpointer->ptw_for_Internode_Length_last = 0;
            Rametpointer->ptw_for_Plastochrone_Interval_0_last = 0;
            Rametpointer->ptw_for_Plastochrone_Interval_last = 0;
            Rametpointer->ptw_for_Square_Meter__Density_last = 0;
            Rametpointer->randomflower_at_phase = ame_rand(0, 1);
            Rametpointer->randomflower = Rametpointer->randomflower_at_phase;
            Rametpointer->ptw_for_delay_death_last = 0;
            Rametpointer->Initial___Nodes = (Rametpointer->channelId==349?4:0);
            Rametpointer->Add_a__node_ = ame_rand(0, 1);
            Rametpointer->ptw_for_Add_a__node__0_last = 0;
            int loop;
            for ( loop = 1; 3>=loop; ++loop ) {
               Rametpointer->MyCoords_last[loop-1] = 0;
            }; /* end(for,loop) */
            for ( arraybuild0 = 1; 1000>=arraybuild0; ++arraybuild0 ) {
               Rametpointer->_array__for_Pl_last[arraybuild0-1] = 0;
               Rametpointer->_array__for_In_last[arraybuild0-1] = 0;
               Rametpointer->_array__for_Gi_last[arraybuild0-1] = 0;
               Rametpointer->_array__for_BRANCH_0_0_last[arraybuild0-1] = 0;
               Rametpointer->_array__for_BRANCH_0_last[arraybuild0-1] = 0;
               Rametpointer->_array__for_Internode_Length_last[arraybuild0-1] = 0;
               Rametpointer->_array__for_Plastochrone_Interval_0_last[arraybuild0-1] = 0;
               Rametpointer->_array__for_Plastochrone_Interval_last[arraybuild0-1] = 0;
               Rametpointer->_array__for_Square_Meter__Density_last[arraybuild0-1] = 0;
               Rametpointer->_array__for_delay_death_last[arraybuild0-1] = 0;
               Rametpointer->_array__for_Add_a__node__0_last[arraybuild0-1] = 0;
            }; /* end(for,arraybuild0) */
         }; /* end(cond,0>=(2>=phase&&Rametpointer->new_instance? -2:phase)) */
         if (1>=(2>=phase&&Rametpointer->new_instance? -2:phase)) {
            Rametpointer->Timed_Square_Meter_Density_count = 0;
         }; /* end(cond,1>=(2>=phase&&Rametpointer->new_instance? -2:phase)) */
         Specific_Growth_Ratetype* Specific_Growth_Ratepointer;
         Specific_Growth_Ratepointer = &(Rametpointer->Specific_Growth_Rate);
         clocktype* clockpointer;
         clockpointer = &(Specific_Growth_Ratepointer->clock);
         Lighttype* Lightpointer;
         Lightpointer = &(Specific_Growth_Ratepointer->Light);
         Light_Adulttype* Light_Adultpointer;
         Light_Adultpointer = &(Lightpointer->Light_Adult);
         Light_Seedlingtype* Light_Seedlingpointer;
         Light_Seedlingpointer = &(Lightpointer->Light_Seedling);
         Temperaturetype* Temperaturepointer;
         Temperaturepointer = &(Specific_Growth_Ratepointer->Temperature);
         Rametpointer->ptw_for_delay_death = (Rametpointer-
>ptw_for_delay_death_last==1000?1:Rametpointer->ptw_for_delay_death_last+1);
         Rametpointer->ptr_for_delay_death = Rametpointer->ptw_for_delay_death-60-
1000*(int)(floor((double)(Rametpointer->ptw_for_delay_death-60-1)/1000));
         Rametpointer->ptw_for_Plastochrone_Interval = (Rametpointer-
>ptw_for_Plastochrone_Interval_last==1000?1:Rametpointer-
>ptw_for_Plastochrone_Interval_last+1);
```

```
        Rametpointer->ptr_for_Plastochrone_Interval = Rametpointer-
>ptw_for_Plastochrone_Interval-1-1000*(int)(floor((double)(Rametpointer-
>ptw_for_Plastochrone_Interval-1-1)/1000));
        Rametpointer->ptw_for_Plastochrone_Interval_0 = (Rametpointer-
>ptw_for_Plastochrone_Interval_0_last==1000?1:Rametpointer-
>ptw_for_Plastochrone_Interval_0_last+1);
        Rametpointer->ptr_for_Plastochrone_Interval_0 = Rametpointer-
>ptw_for_Plastochrone_Interval_0-1-1000*(int)(floor((double)(Rametpointer-
>ptw_for_Plastochrone_Interval_0-1-1)/1000));
        Node_births_and_lengthstype* Node_births_and_lengthspointer;
        Node_births_and_lengthstype** Node_births_and_lengthsmeta;
        Node_births_and_lengthsmeta = &(Rametpointer->Node_births_and_lengths);
        while ( *Node_births_and_lengthsmeta != 0 ) {
          abort_check(this);
          Node_births_and_lengthspointer = *Node_births_and_lengthsmeta;
          Node_births_and_lengthspointer->new_instance = 0;
          Node_births_and_lengthsmeta = &(Node_births_and_lengthspointer->next);
        }; /* end(while,*Node_births_and_lengthsmeta) */
        while ( Rametpointer->Add_a__node_>=1 ) {
          abort_check(this);
          Rametpointer->Add_a__node_ = Rametpointer->Add_a__node_-1;
          ++Rametpointer->Node_births_and_lengthscount;
          Node_births_and_lengthspointer = new Node_births_and_lengthstype;
          init_pop_member(Node_births_and_lengthspointer, Rametpointer-
>Node_births_and_lengthscount, 294);
          *Node_births_and_lengthsmeta = Node_births_and_lengthspointer;
          Node_births_and_lengthsmeta = &(Node_births_and_lengthspointer->next);
        }; /* end(while,New instances) */
        Virtual_Eelgrass_Meadow_v_2_VAnpointer->countME_count =
Virtual_Eelgrass_Meadow_v_2_VAnpointer->countME_count+1;
        Virtual_Eelgrass_Meadow_v_2_VAnpointer->Average_Iz_count =
Virtual_Eelgrass_Meadow_v_2_VAnpointer->Average_Iz_count+1;
        Virtual_Eelgrass_Meadow_v_2_VAnpointer->averageIleaf_count =
Virtual_Eelgrass_Meadow_v_2_VAnpointer->averageIleaf_count+1;
        Virtual_Eelgrass_Meadow_v_2_VAnpointer->avgkcanopy_count =
Virtual_Eelgrass_Meadow_v_2_VAnpointer->avgkcanopy_count+1;
        Virtual_Eelgrass_Meadow_v_2_VAnpointer->Total_Branches_of_Eves_sum = Rametpointer-
>Branching_Rate_Fix+Virtual_Eelgrass_Meadow_v_2_VAnpointer->Total_Branches_of_Eves_sum;
        Virtual_Eelgrass_Meadow_v_2_VAnpointer->sum_Eve_sum = Rametpointer-
>Eve_Status+Virtual_Eelgrass_Meadow_v_2_VAnpointer->sum_Eve_sum;
        Virtual_Eelgrass_Meadow_v_2_VAnpointer->Number_of__Shoots_count =
Virtual_Eelgrass_Meadow_v_2_VAnpointer->Number_of__Shoots_count+1;
        Virtual_Eelgrass_Meadow_v_2_VAnpointer->Average_Internode__Length_count =
Virtual_Eelgrass_Meadow_v_2_VAnpointer->Average_Internode__Length_count+1;
        Virtual_Eelgrass_Meadow_v_2_VAnpointer->averageLL_count =
Virtual_Eelgrass_Meadow_v_2_VAnpointer->averageLL_count+1;
        Virtual_Eelgrass_Meadow_v_2_VAnpointer->Average_Density_count =
Virtual_Eelgrass_Meadow_v_2_VAnpointer->Average_Density_count+1;
        Virtual_Eelgrass_Meadow_v_2_VAnpointer->rcmXNWY_count_0 =
Virtual_Eelgrass_Meadow_v_2_VAnpointer->rcmXNWY_count_0+1;
        Virtual_Eelgrass_Meadow_v_2_VAnpointer->rcmXNWY_count =
Virtual_Eelgrass_Meadow_v_2_VAnpointer->rcmXNWY_count+1;
        Virtual_Eelgrass_Meadow_v_2_VAnpointer->Radius_of_Gyration_count =
Virtual_Eelgrass_Meadow_v_2_VAnpointer->Radius_of_Gyration_count+1;
        clockpointer->days = (int)(ts[1]);
```

```
        clockpointer->year = (int)((double)(clockpointer->days-1)/365)+1;
        clockpointer->dayofyear = clockpointer->days-(clockpointer->year-1)*365;
        clockpointer->month = (clockpointer->dayofyear<32?1:(clockpointer-
>dayofyear<60?2:(clockpointer->dayofyear<91?3:(clockpointer->dayofyear<121?4:(clockpointer-
>dayofyear<152?5:(clockpointer->dayofyear<182?6:(clockpointer->dayofyear<213?7:(clockpointer-
>dayofyear<244?8:(clockpointer->dayofyear<274?9:(clockpointer->dayofyear<305?10:(clockpointer-
>dayofyear<335?11:12)))))))));
        clockpointer->dayreal = ts[1]-(clockpointer->year-1)*365;
        Rametpointer->Rhizome_Length_sum = 0;
        Rametpointer->MyCoords_at_phase_0 = ame_rand(0,  6.36);
        Rametpointer->MyCoords_at_phase = ame_rand(0,  6.36);
        Rametpointer->age_of__youngest_node_least = 1.0e+100;
        Rametpointer->my_number__of_nodes_count = 0;
        Rametpointer->youngest__birthday_greatest = -1.0e+100;
        for ( arraybuild0 = 1; 1000>=arraybuild0; ++arraybuild0 ) {
        }; /* end(for,arraybuild0) */
        int loop;
        for ( loop = 1; 3>=loop; ++loop ) {
        }; /* end(for,loop) */
        Rametpointer = Rametpointer->next;
      }; /* end(while,Rametpointer) */
      Virtual_Eelgrass_Meadow_v_2_VAnpointer->Number_of__Shoots =
Virtual_Eelgrass_Meadow_v_2_VAnpointer->Number_of__Shoots_count;
      Virtual_Eelgrass_Meadow_v_2_VAnpointer->shoot_density =
(ts[2]==0?0:(double)(Virtual_Eelgrass_Meadow_v_2_VAnpointer-
>Number_of__Shoots)/((Virtual_Eelgrass_Meadow_v_2_VAnpointer->Greatest_Y-
Virtual_Eelgrass_Meadow_v_2_VAnpointer-
>Least_Y)*(Virtual_Eelgrass_Meadow_v_2_VAnpointer->Greatest_XNW-
Virtual_Eelgrass_Meadow_v_2_VAnpointer->Least_XNW)));
      Virtual_Eelgrass_Meadow_v_2_VAnpointer->sum_Eve =
Virtual_Eelgrass_Meadow_v_2_VAnpointer->sum_Eve_sum;
      Virtual_Eelgrass_Meadow_v_2_VAnpointer->Total_Branches_of_Eves =
Virtual_Eelgrass_Meadow_v_2_VAnpointer->Total_Branches_of_Eves_sum;
      Virtual_Eelgrass_Meadow_v_2_VAnpointer->average_Eve_Branching =
Virtual_Eelgrass_Meadow_v_2_VAnpointer->Total_Branches_of_Eves;
      Virtual_Eelgrass_Meadow_v_2_VAnpointer->countME =
Virtual_Eelgrass_Meadow_v_2_VAnpointer->countME_count;
      int Eve_Plastochrone_Intervalscond;
      Eve_Plastochrone_Intervalstype** Eve_Plastochrone_Intervalsmeta;
      Eve_Plastochrone_Intervalsmeta = &(Virtual_Eelgrass_Meadow_v_2_VAnpointer-
>Eve_Plastochrone_Intervals);
      Ramettype* EvetaggingRametptr;
      EvetaggingRametptr = Virtual_Eelgrass_Meadow_v_2_VAnpointer->Ramet;
      while ( EvetaggingRametptr != 0 ) {
        abort_check(this);
        Ramettype* Rametprogen;
        Rametprogen = (Ramettype*)EvetaggingRametptr->baseptrs[0];
        Eve_Plastochrone_Intervalstype* Eve_Plastochrone_Intervalspointer;
        abort_check(this);
        int check_members;
        check_members = 0>=(2>=phase&&EvetaggingRametptr->new_instance? -2:phase);
        if (prune(Eve_Plastochrone_Intervalsmeta, 1, EvetaggingRametptr->instanceid[0])) {
          Eve_Plastochrone_Intervalspointer = *Eve_Plastochrone_Intervalsmeta;
          Eve_Plastochrone_Intervalspointer->new_instance = 0;
          if (check_members) {
```

```
            *Eve_Plastochrone_Intervalsmeta = Eve_Plastochrone_Intervalspointer->next;
          } else { /* check_members */
            Eve_Plastochrone_Intervalsmeta = &(Eve_Plastochrone_Intervalspointer->next);
          }; /* end(cond,check_members) */
        } else { /* Instance exists */
          if (check_members) {
            Eve_Plastochrone_Intervalspointer = new Eve_Plastochrone_Intervalstype;
            Eve_Plastochrone_Intervalspointer->instanceid[0] = EvetaggingRametptr->instanceid[0];
            Eve_Plastochrone_Intervalspointer->baseptrs[0] = EvetaggingRametptr;
            Eve_Plastochrone_Intervalspointer->new_instance = 1;
          }; /* end(cond,check_members) */
        }; /* end(cond,Instance exists) */
        if (check_members) {
          Eve_Plastochrone_Intervalspointer->I_am_Eve = EvetaggingRametptr->Eve_Status==1;
          if (Eve_Plastochrone_Intervalspointer->I_am_Eve) {
            Eve_Plastochrone_Intervalspointer->next = *Eve_Plastochrone_Intervalsmeta;
            *Eve_Plastochrone_Intervalsmeta = Eve_Plastochrone_Intervalspointer;
            Eve_Plastochrone_Intervalsmeta = &(Eve_Plastochrone_Intervalspointer->next);
          } else { /* Eve_Plastochrone_Intervalspointer->I_am_Eve */
            delete Eve_Plastochrone_Intervalspointer;
          }; /* end(cond,Eve_Plastochrone_Intervalspointer->I_am_Eve) */
          }; /* end(cond,check_members) */
        EvetaggingRametptr = EvetaggingRametptr->next;
      }; /* end(while,EvetaggingRametptr) */
      delete_list(*Eve_Plastochrone_Intervalsmeta);
      *Eve_Plastochrone_Intervalsmeta = 0;
      Eve_Plastochrone_Intervalstype* Eve_Plastochrone_Intervalspointer;
      Eve_Plastochrone_Intervalspointer = Virtual_Eelgrass_Meadow_v_2_VAnpointer-
>Eve_Plastochrone_Intervals;
      while ( Eve_Plastochrone_Intervalspointer != 0 ) {
        abort_check(this);
        Ramettype* EvetaggingRametptr;
        EvetaggingRametptr = (Ramettype*)Eve_Plastochrone_Intervalspointer->baseptrs[0];
        if (-2>=(2>=phase&&Eve_Plastochrone_Intervalspointer->new_instance? -2:phase)) {
          Eve_Plastochrone_Intervalspointer->count = EvetaggingRametptr->one;
        }; /* end(cond,-2>=(2>=phase&&Eve_Plastochrone_Intervalspointer->new_instance? -
2:phase)) */
        EveStatspointer->Tagged_PI_sum_0 = Eve_Plastochrone_Intervalspointer-
>count+EveStatspointer->Tagged_PI_sum_0;
        EveStatspointer->mean_number__of_new_nodes_sum_0 = Eve_Plastochrone_Intervalspointer-
>count+EveStatspointer->mean_number__of_new_nodes_sum_0;
        EveStatspointer->mean_new_leaves_sum_0 = Eve_Plastochrone_Intervalspointer-
>count+EveStatspointer->mean_new_leaves_sum_0;
        EveStatspointer->stdevnodes_sum = Eve_Plastochrone_Intervalspointer-
>count+EveStatspointer->stdevnodes_sum;
        EveStatspointer->aboveavg_count = EveStatspointer->aboveavg_count+1;
        EveStatspointer->abovestdev_sum = Eve_Plastochrone_Intervalspointer-
>count+EveStatspointer->abovestdev_sum;
        EveStatspointer->avgbelow_count = EveStatspointer->avgbelow_count+1;
        EveStatspointer->belowstdev_sum = Eve_Plastochrone_Intervalspointer-
>count+EveStatspointer->belowstdev_sum;
        Eve_Plastochrone_Intervalspointer->Leaves_Shed = EvetaggingRametptr->LeavesProduced;
        EveStatspointer->mean_new_leaves_sum = Eve_Plastochrone_Intervalspointer-
>Leaves_Shed+EveStatspointer->mean_new_leaves_sum;
        Eve_Plastochrone_Intervalspointer = Eve_Plastochrone_Intervalspointer->next;
```

```
      }; /* end(while,Eve_Plastochrone_Intervalspointer) */
      EveStatspointer->mean_new_leaves = (double)(EveStatspointer-
>mean_new_leaves_sum)/EveStatspointer->mean_new_leaves_sum_0;
      Rametpointer = Virtual_Eelgrass_Meadow_v_2_VAnpointer->Ramet;
      while ( Rametpointer != 0 ) {
         abort_check(this);
         Ramettype* Rametprogen;
         Rametprogen = (Ramettype*)Rametpointer->baseptrs[0];
         if (0>=(2>=phase&&Rametpointer->new_instance? -2:phase)) {
            Node_births_and_lengthstype* Node_births_and_lengthspointer;
            Node_births_and_lengthstype** Node_births_and_lengthsmeta;
            Rametpointer->Node_births_and_lengthscount = 0;
            Node_births_and_lengthsmeta = &(Rametpointer->Node_births_and_lengths);
            Rametpointer->Node_births_and_lengthscount =
init_pop(&(Node_births_and_lengthsmeta), Rametpointer->Initial___Nodes, Rametpointer-
>Node_births_and_lengthscount, 293);
            delete_list(*Node_births_and_lengthsmeta);
            *Node_births_and_lengthsmeta = 0;
         }; /* end(cond,0>=(2>=phase&&Rametpointer->new_instance? -2:phase)) */
         Adulthoodtype* Adulthoodpointer;
         Adulthoodpointer = &(Rametpointer->Adulthood);
         Rametpointer->ptw_for_delay_death_last = Rametpointer->ptw_for_delay_death;
         Rametpointer->ptw_for_Plastochrone_Interval_last = Rametpointer-
>ptw_for_Plastochrone_Interval;
         Rametpointer->ptw_for_Plastochrone_Interval_0_last = Rametpointer-
>ptw_for_Plastochrone_Interval_0;
         Node_births_and_lengthstype* Node_births_and_lengthspointer;
         Node_births_and_lengthspointer = Rametpointer->Node_births_and_lengths;
         while ( Node_births_and_lengthspointer != 0 ) {
            abort_check(this);
            Node_births_and_lengthstype* Node_births_and_lengthsprogen;
            Node_births_and_lengthsprogen =
(Node_births_and_lengthstype*)Node_births_and_lengthspointer->baseptrs[0];
            if (-2>=(2>=phase&&Node_births_and_lengthspointer->new_instance? -2:phase)) {
               Node_births_and_lengthspointer->index = Node_births_and_lengthspointer-
>instanceid[0];
            }; /* end(cond,-2>=(2>=phase&&Node_births_and_lengthspointer->new_instance? -
2:phase)) */
            if (0>=(2>=phase&&Node_births_and_lengthspointer->new_instance? -2:phase)) {
               Node_births_and_lengthspointer->node_age_at_phase = ts[1];
               Node_births_and_lengthspointer->node_birthday_at_phase = ts[1];
               Node_births_and_lengthspointer->node_birthday = (Node_births_and_lengthspointer-
>channelId==293?0:Node_births_and_lengthspointer->node_birthday_at_phase);
            }; /* end(cond,0>=(2>=phase&&Node_births_and_lengthspointer->new_instance? -
2:phase)) */
            Node_births_and_lengthspointer->node_age = ts[1]-Node_births_and_lengthspointer-
>node_age_at_phase;
            Rametpointer->age_of__youngest_node_least = min(Node_births_and_lengthspointer-
>node_age,Rametpointer->age_of__youngest_node_least);
            Rametpointer->my_number__of_nodes_count = Rametpointer-
>my_number__of_nodes_count+1;
            Rametpointer->youngest__birthday_greatest = max(Node_births_and_lengthspointer-
>node_birthday,Rametpointer->youngest__birthday_greatest);
            Node_births_and_lengthspointer = Node_births_and_lengthspointer->next;
         }; /* end(while,Node_births_and_lengthspointer) */
```

```
        Rametpointer->youngest__birthday = Rametpointer->youngest__birthday_greatest;
        Rametpointer->my_number__of_nodes = Rametpointer->my_number__of_nodes_count;
        Adulthoodpointer->is_adult = (Rametpointer->channelId==349 || Rametpointer-
>channelId==261 || Rametpointer->my_number__of_nodes>=4?1:0);
        Rametpointer->adult_status = (Adulthoodpointer->is_adult==1?1:0);
        Virtual_Eelgrass_Meadow_v_2_VAnpointer->sum_adults_sum = Rametpointer-
>adult_status+Virtual_Eelgrass_Meadow_v_2_VAnpointer->sum_adults_sum;
        Rametpointer->initial_ch = (Adulthoodpointer->is_adult?12:5.5);
        Virtual_Eelgrass_Meadow_v_2_VAnpointer->how_many_adults_true_sum =
(Adulthoodpointer->is_adult?1:0)+Virtual_Eelgrass_Meadow_v_2_VAnpointer-
>how_many_adults_true_sum;
        Rametpointer->age_of__youngest_node = Rametpointer->age_of__youngest_node_least;
        for ( arraybuild0 = 1; 1000>=arraybuild0; ++arraybuild0 ) {
           Rametpointer->_array__for_Plastochrone_Interval_0[arraybuild0-1] =
(arraybuild0==Rametpointer->ptw_for_Plastochrone_Interval_0?Rametpointer-
>youngest__birthday:Rametpointer->_array__for_Plastochrone_Interval_0_last[arraybuild0-1]);
           Rametpointer->_array__for_Plastochrone_Interval[arraybuild0-1] =
(arraybuild0==Rametpointer->ptw_for_Plastochrone_Interval?Rametpointer-
>youngest__birthday:Rametpointer->_array__for_Plastochrone_Interval_last[arraybuild0-1]);
        }; /* end(for,arraybuild0) */
        Rametpointer->Plastochrone_Interval = (Rametpointer-
>_array__for_Plastochrone_Interval[Rametpointer->ptr_for_Plastochrone_Interval-1]==Rametpointer-
>youngest__birthday?0:Rametpointer->youngest__birthday-Rametpointer-
>_array__for_Plastochrone_Interval_0[Rametpointer->ptr_for_Plastochrone_Interval_0-1]);
        for ( arraybuild0 = 1; 1000>=arraybuild0; ++arraybuild0 ) {
           Rametpointer->_array__for_Plastochrone_Interval_0_last[arraybuild0-1] = Rametpointer-
>_array__for_Plastochrone_Interval_0[arraybuild0-1];
           Rametpointer->_array__for_Plastochrone_Interval_last[arraybuild0-1] = Rametpointer-
>_array__for_Plastochrone_Interval[arraybuild0-1];
        }; /* end(for,arraybuild0) */
        Rametpointer = Rametpointer->next;
     }; /* end(while,Rametpointer) */
     Virtual_Eelgrass_Meadow_v_2_VAnpointer->how_many_adults_true =
Virtual_Eelgrass_Meadow_v_2_VAnpointer->how_many_adults_true_sum;
     Virtual_Eelgrass_Meadow_v_2_VAnpointer->sum_adults =
Virtual_Eelgrass_Meadow_v_2_VAnpointer->sum_adults_sum;
     Eve_Plastochrone_Intervalstype* Eve_Plastochrone_Intervalspointer_0;
     Eve_Plastochrone_Intervalspointer_0 = Virtual_Eelgrass_Meadow_v_2_VAnpointer-
>Eve_Plastochrone_Intervals;
     while ( Eve_Plastochrone_Intervalspointer_0 != 0 ) {
        abort_check(this);
        Ramettype* EvetaggingRametptr;
        EvetaggingRametptr = (Ramettype*)Eve_Plastochrone_Intervalspointer_0->baseptrs[0];
        Eve_Plastochrone_Intervalspointer_0->Eve_PI = EvetaggingRametptr->Plastochrone_Interval;
        EveStatspointer->Tagged_PI_sum = Eve_Plastochrone_Intervalspointer_0-
>Eve_PI+EveStatspointer->Tagged_PI_sum;
        EveStatspointer->GreatestPI_greatest = max(Eve_Plastochrone_Intervalspointer_0-
>Eve_PI,EveStatspointer->GreatestPI_greatest);
        Eve_Plastochrone_Intervalspointer_0 = Eve_Plastochrone_Intervalspointer_0->next;
     }; /* end(while,Eve_Plastochrone_Intervalspointer_0) */
     EveStatspointer->GreatestPI = EveStatspointer->GreatestPI_greatest;
     EveStatspointer->Tagged_PI = (EveStatspointer-
>Tagged_PI_sum==0?0:(double)(EveStatspointer->Tagged_PI_sum)/EveStatspointer-
>Tagged_PI_sum_0);
     Ramettype* Rametpointer_0;
```

```
      Rametpointer_0 = Virtual_Eelgrass_Meadow_v_2_VAnpointer->Ramet;
      while ( Rametpointer_0 != 0 ) {
         abort_check(this);
         Ramettype* Rametprogen;
         Rametprogen = (Ramettype*)Rametpointer_0->baseptrs[0];
         if (0>=(2>=phase&&Rametpointer_0->new_instance? -2:phase)) {
            Adulthoodtype* Adulthoodpointer;
            Adulthoodpointer = &(Rametpointer_0->Adulthood);
            Rametpointer_0->GROWING_INTERNODE = (double)((double)(Rametpointer_0-
>initial_ch*0.0011999999999999999)/1.2*4.7000000000000002)/4;
            Rametpointer_0->LEAFBUNDLE = (Adulthoodpointer-
>is_adult?(double)(Rametpointer_0-
>initial_ch*0.0011999999999999999)/1.2*3.7000000000000002:0.0015);
            Rametpointer_0->ROOTS___RHIZOMES = (double)((double)(Rametpointer_0-
>initial_ch*0.0011999999999999999)/1.2*4.7000000000000002)/4*2.5;
            Rametpointer_0->OLDESTLEAF = (Adulthoodpointer->is_adult?(double)(Rametpointer_0-
>initial_ch*0.0011999999999999999)/1.2:0.014999999999999999);
            Rametpointer_0->NODES = (Adulthoodpointer->is_adult?4:0);
         }; /* end(cond,0>=(2>=phase&&Rametpointer_0->new_instance? -2:phase)) */
         Rametpointer_0 = Rametpointer_0->next;
      }; /* end(while,Rametpointer_0) */
   }; /* end(cond,2>=phase) */
   Virtual_Eelgrass_Meadow_v_2_VAnpointer = &(this->Virtual_Eelgrass_Meadow_v_2_VAn);
   EveStatstype* EveStatspointer;
   EveStatspointer = &(Virtual_Eelgrass_Meadow_v_2_VAnpointer->EveStats);
   Ramettype* Rametpointer;
   Rametpointer = Virtual_Eelgrass_Meadow_v_2_VAnpointer->Ramet;
   while ( Rametpointer != 0 ) {
      abort_check(this);
      Ramettype* Rametprogen;
      Rametprogen = (Ramettype*)Rametpointer->baseptrs[0];
      Adulthoodtype* Adulthoodpointer;
      Adulthoodpointer = &(Rametpointer->Adulthood);
      Specific_Growth_Ratetype* Specific_Growth_Ratepointer;
      Specific_Growth_Ratepointer = &(Rametpointer->Specific_Growth_Rate);
      Lighttype* Lightpointer;
      Lightpointer = &(Specific_Growth_Ratepointer->Light);
      Light_Seedlingtype* Light_Seedlingpointer;
      Light_Seedlingpointer = &(Lightpointer->Light_Seedling);
      Grass_Machinetype* Grass_Machinepointer;
      Grass_Machinepointer = &(Specific_Growth_Ratepointer->Grass_Machine);
      SGR_Seedlingtype* SGR_Seedlingpointer;
      SGR_Seedlingpointer = &(Grass_Machinepointer->SGR_Seedling);
      Flowertype* Flowerpointer;
      Flowerpointer = &(Rametpointer->Flower);
      Temperaturetype* Temperaturepointer;
      Temperaturepointer = &(Specific_Growth_Ratepointer->Temperature);
      Physical_Settingtype* Physical_Settingpointer;
      Physical_Settingpointer = &(Specific_Growth_Ratepointer->Physical_Setting);
      SGR_Adulttype* SGR_Adultpointer;
      SGR_Adultpointer = &(Grass_Machinepointer->SGR_Adult);
      Rametpointer->Io = (Rametpointer->NODES==4?4:0);
      Rametpointer->ptw_for_Add_a__node__0 = (Rametpointer-
>ptw_for_Add_a__node__0_last==1000?1:Rametpointer->ptw_for_Add_a__node__0_last+1);
```

```
	Rametpointer->ptr_for_Add_a__node__0 = Rametpointer->ptw_for_Add_a__node__0-1-
1000*(int)(floor((double)(Rametpointer->ptw_for_Add_a__node__0-1-1)/1000));
	Rametpointer->ptw_for_Square_Meter__Density = (Rametpointer-
>ptw_for_Square_Meter__Density_last==1000?1:Rametpointer-
>ptw_for_Square_Meter__Density_last+1);
	Rametpointer->ptr_for_Square_Meter__Density = Rametpointer-
>ptw_for_Square_Meter__Density-fmod(ts[2],10)-1000*(int)(floor((double)(Rametpointer-
>ptw_for_Square_Meter__Density-fmod(ts[2],10)-1)/1000));
	Rametpointer->ptw_for_Internode_Length = (Rametpointer-
>ptw_for_Internode_Length_last==1000?1:Rametpointer->ptw_for_Internode_Length_last+1);
	Rametpointer->ptr_for_Internode_Length = Rametpointer->ptw_for_Internode_Length-1-
1000*(int)(floor((double)(Rametpointer->ptw_for_Internode_Length-1-1)/1000));
	Rametpointer->Di__branch = (Rametpointer->Eve_Status==1?5:((Rametprogen?Rametprogen-
>Di__branch_in_progenitor:0)==5?0.25:0.75));
	Rametpointer->Di__branch_in_progenitor = Rametpointer->NODES;
	Rametpointer->ptw_for_BRANCH_0 = (Rametpointer-
>ptw_for_BRANCH_0_last==1000?1:Rametpointer->ptw_for_BRANCH_0_last+1);
	Rametpointer->ptr_for_BRANCH_0 = Rametpointer->ptw_for_BRANCH_0-1-
1000*(int)(floor((double)(Rametpointer->ptw_for_BRANCH_0-1-1)/1000));
	Rametpointer->ptw_for_BRANCH_0_0 = (Rametpointer-
>ptw_for_BRANCH_0_0_last==1000?1:Rametpointer->ptw_for_BRANCH_0_0_last+1);
	Rametpointer->ptr_for_BRANCH_0_0 = Rametpointer->ptw_for_BRANCH_0_0-1-
1000*(int)(floor((double)(Rametpointer->ptw_for_BRANCH_0_0-1-1)/1000));
	Rametpointer->ptw_for_Gi = (Rametpointer->ptw_for_Gi_last==1000?1:Rametpointer-
>ptw_for_Gi_last+1);
	Rametpointer->ptr_for_Gi = Rametpointer->ptw_for_Gi-1-
1000*(int)(floor((double)(Rametpointer->ptw_for_Gi-1-1)/1000));
	Rametpointer->ptw_for_In = (Rametpointer->ptw_for_In_last==1000?1:Rametpointer-
>ptw_for_In_last+1);
	Rametpointer->ptr_for_In = Rametpointer->ptw_for_In-1-
1000*(int)(floor((double)(Rametpointer->ptw_for_In-1-1)/1000));
	Rametpointer->init_XNW = (ts[2]<=1?Rametpointer->init_XNW_at_phase:Rametpointer-
>init_XNW_last);
	Rametpointer->init_y = (ts[2]<=1?Rametpointer->init_y_at_phase:Rametpointer->init_y_last);
	Rametpointer->ptw_for_Pl = (Rametpointer->ptw_for_Pl_last==1000?1:Rametpointer-
>ptw_for_Pl_last+1);
	Rametpointer->ptr_for_Pl = Rametpointer->ptw_for_Pl-1-
1000*(int)(floor((double)(Rametpointer->ptw_for_Pl-1-1)/1000));
	Physical_Settingpointer->k_SA = array_9[(int)(ts[2])-1];
	Temperaturepointer->Temperature_SA = array_10[(int)(ts[2])-1];
	Temperaturepointer->flow1 = (Temperaturepointer->Temperature_SA>20?1:0);
	Temperaturepointer->FDays = (Temperaturepointer->Temperature_SA>15?1:0);
	Temperaturepointer->GDays = (Temperaturepointer->Temperature_SA<20&&ts[2]>244?1:0);
	SGR_Adultpointer->umax_A = (Temperaturepointer->Temperature_SA<=0?-
0.001:0.0177*log(Temperaturepointer->Temperature_SA)+0.0011000000000000001);
	SGR_Adultpointer->alpha_A = (Temperaturepointer-
>Temperature_SA>30?0.0050000000000000001:(Temperaturepointer-
>Temperature_SA<5?0.0050000000000000001:-0.00029999999999999997*pow(Temperaturepointer-
>Temperature_SA,2)+0.0070000000000000001*Temperaturepointer->Temperature_SA-
0.029600000000000001));
	SGR_Adultpointer->ro_A = -0.000727*Temperaturepointer->Temperature_SA-
0.0068250000000000003;
	Flowerpointer->flower = (Temperaturepointer->Temperature_SA>=15&&(ts[2]==103 ||
ts[2]==468 || ts[2]==833 || ts[2]==1198)&&Rametpointer-
>my_number__of_nodes>=3&&Rametpointer->randomflower<=(Temperaturepointer-
```

```
>Daysabove20<=100?0.29999999999999999:(Temperaturepointer-
>Daysabove20>=150?0.90000000000000002:0.5))?1:0);
      Rametpointer->seeds_perflowerShoot = (Flowerpointer->flower==1?10:0);
      Virtual_Eelgrass_Meadow_v_2_VAnpointer->sum_seeds_sum = Rametpointer-
>seeds_perflowerShoot+Virtual_Eelgrass_Meadow_v_2_VAnpointer->sum_seeds_sum;
      Rametpointer->b_A = (Temperaturepointer-
>Temperature_SA<8.0640000000000001?0.65000000000000002:-
0.034599999999999999*Temperaturepointer->Temperature_SA+1.0289999999999999);
      Light_Seedlingpointer->Surface_PAR_S = array_11[(int)(ts[2])-1];
      Light_Seedlingpointer->Iz_SA = Light_Seedlingpointer->Surface_PAR_S*exp(-
Physical_Settingpointer->k_SA*Physical_Settingpointer->water_depth_SA);
      SGR_Seedlingpointer->F_I_T__S = (Temperaturepointer-
>Temperature_SA<=25?(double)((0.96999999999999997*Temperaturepointer->Temperature_SA-
0.75)*tanh((double)(Light_Seedlingpointer->Iz_SA)/(4.5978000000000003*Temperaturepointer-
>Temperature_SA+3.3473000000000002))+(-0.094899999999999998*Temperaturepointer-
>Temperature_SA-1.0503))/20.02:(double)((-2.7999999999999998*Temperaturepointer-
>Temperature_SA+94.299999999999997)*tanh((double)(Light_Seedlingpointer-
>Iz_SA)/(4.5978000000000003*Temperaturepointer->Temperature_SA+3.3473000000000002))+(-
0.094899999999999998*Temperaturepointer->Temperature_SA-1.0503))/20.02);
      SGR_Seedlingpointer->SGR_S = SGR_Seedlingpointer->umax_S*SGR_Seedlingpointer-
>F_I_T__S*SGR_Seedlingpointer->sediment_limitation_S;
      Rametpointer->GI_S = (Flowerpointer->flower==1?0:SGR_Seedlingpointer->SGR_S);
      Light_Seedlingpointer->Percent_Irradiance_S = (double)(Light_Seedlingpointer-
>Iz_SA)/Light_Seedlingpointer->Surface_PAR_S;
      Rametpointer->b_S = (Light_Seedlingpointer-
>Iz_SA<=90?0.29999999999999999:(Light_Seedlingpointer-
>Iz_SA>=499?0.51000000000000001:0.42999999999999999));
      Rametpointer->b = (Adulthoodpointer->is_adult==1?Rametpointer->b_A:Rametpointer->b_S);
      Rametpointer->average_internode_length_sum = 0;
      for ( arraybuild0 = 1; 1000>=arraybuild0; ++arraybuild0 ) {
         Rametpointer->_array__for_Add_a__node__0[arraybuild0-1] = (arraybuild0==Rametpointer-
>ptw_for_Add_a__node__0?Rametpointer->OLDESTLEAF:Rametpointer-
>_array__for_Add_a__node__0_last[arraybuild0-1]);
         Rametpointer->_array__for_Internode_Length[arraybuild0-1] = (arraybuild0==Rametpointer-
>ptw_for_Internode_Length?Rametpointer->OLDESTLEAF:Rametpointer-
>_array__for_Internode_Length_last[arraybuild0-1]);
         Rametpointer->_array__for_BRANCH_0[arraybuild0-1] = (arraybuild0==Rametpointer-
>ptw_for_BRANCH_0?Rametpointer->NODES:Rametpointer-
>_array__for_BRANCH_0_last[arraybuild0-1]);
         Rametpointer->_array__for_BRANCH_0_0[arraybuild0-1] = (arraybuild0==Rametpointer-
>ptw_for_BRANCH_0_0?Rametpointer->NODES:Rametpointer-
>_array__for_BRANCH_0_0_last[arraybuild0-1]);
         Rametpointer->_array__for_Gi[arraybuild0-1] = (arraybuild0==Rametpointer-
>ptw_for_Gi?Rametpointer->OLDESTLEAF:Rametpointer->_array__for_Gi_last[arraybuild0-1]);
         Rametpointer->_array__for_In[arraybuild0-1] = (arraybuild0==Rametpointer-
>ptw_for_In?Rametpointer->OLDESTLEAF:Rametpointer->_array__for_In_last[arraybuild0-1]);
         Rametpointer->_array__for_Pl[arraybuild0-1] = (arraybuild0==Rametpointer-
>ptw_for_Pl?Rametpointer->OLDESTLEAF:Rametpointer->_array__for_Pl_last[arraybuild0-1]);
      }; /* end(for,arraybuild0) */
      Rametpointer->Pl = (Rametpointer->_array__for_Pl[Rametpointer->ptr_for_Pl-1]>Rametpointer-
>OLDESTLEAF?1:0);
      Rametpointer->In = (Rametpointer->_array__for_In[Rametpointer->ptr_for_In-1]>Rametpointer-
>OLDESTLEAF?1:0);
      Rametpointer->Gi = (Rametpointer->_array__for_Gi[Rametpointer->ptr_for_Gi-
1]>Rametpointer->OLDESTLEAF?Rametpointer->GROWING_INTERNODE:0);
```

```
      Rametpointer->BRANCH_0 = (Rametpointer->NODES==4&&Rametpointer-
>_array__for_BRANCH_0[Rametpointer->ptr_for_BRANCH_0-1]==3?1:(Rametpointer-
>NODES==3&&Rametpointer->_array__for_BRANCH_0_0[Rametpointer->ptr_for_BRANCH_0_0-
1]==2?1:0));
      Rametpointer->Internode_Length = (ts[1]==1?(double)(Rametpointer-
>ROOTS___RHIZOMES*317.27999999999997)/1000:(Rametpointer-
>_array__for_Internode_Length[Rametpointer->ptr_for_Internode_Length-1]>Rametpointer-
>OLDESTLEAF?(double)(317.27999999999997*Rametpointer-
>GROWING_INTERNODE)/1000:0));
      Rametpointer->Add_a__node__0 = (Rametpointer-
>_array__for_Add_a__node__0[Rametpointer->ptr_for_Add_a__node__0-1]>Rametpointer-
>OLDESTLEAF?1:0);
      int loop;
      for ( loop = 1; 3>=loop; ++loop ) {
         Rametpointer->MyCoords[loop-1] = (ts[2]<=1?(loop==1?Rametpointer-
>init_XNW:(loop==2?Rametpointer->init_y:Rametpointer->MyCoords_at_phase)):(Rametpointer-
>channelId==345?(loop==1?Rametpointer->init_XNW:(loop==2?Rametpointer-
>init_y:Rametpointer->MyCoords_at_phase_0)):Rametpointer->MyCoords_last[loop-1]));
         Rametpointer->My_Birthplace[loop-1] = (Rametpointer->Eve_Status==1?Rametpointer-
>MyCoords[loop-1]:(Rametprogen?Rametprogen->My_Birthplace_in_progenitor[loop-1]:0));
         Rametpointer->My_Birthplace_in_progenitor[loop-1] = Rametpointer->MyCoords[loop-1];
      }; /* end(for,loop) */
      Rametpointer->Move_YAPEXNW = (Rametpointer->channelId==349?Rametpointer-
>Internode_Length*sin(Rametpointer->My_Birthplace[3-1]):(Rametpointer-
>Di__branch==0.25?Rametpointer->Internode_Length*sin(Rametpointer->My_Birthplace[3-
1]+1.3260000000000001):Rametpointer->Internode_Length*sin(Rametpointer->My_Birthplace[3-1]-
1.3260000000000001)));
      Rametpointer->Move_XNWAPEXNW = (Rametpointer->channelId==349?Rametpointer-
>Internode_Length*cos(Rametpointer->My_Birthplace[3-1]):(Rametpointer-
>Di__branch==0.25?Rametpointer->Internode_Length*cos(Rametpointer->My_Birthplace[3-
1]+1.3260000000000001):Rametpointer->Internode_Length*cos(Rametpointer->My_Birthplace[3-1]-
1.3260000000000001)));
      Rametpointer = Rametpointer->next;
   }; /* end(while,Rametpointer) */
   Virtual_Eelgrass_Meadow_v_2_VAnpointer->sum_seeds =
Virtual_Eelgrass_Meadow_v_2_VAnpointer->sum_seeds_sum;
   Virtual_Eelgrass_Meadow_v_2_VAnpointer->Seeds_that_will_Germinate__Seed_Bank_ =
(int)(round(Virtual_Eelgrass_Meadow_v_2_VAnpointer->sum_seeds*0.40000000000000002));
   for ( arraybuild0 = 1; 1000>=arraybuild0; ++arraybuild0 ) {
      Virtual_Eelgrass_Meadow_v_2_VAnpointer->_array__for_delay__seeds[arraybuild0-1] =
(arraybuild0==Virtual_Eelgrass_Meadow_v_2_VAnpointer-
>ptw_for_delay__seeds?Virtual_Eelgrass_Meadow_v_2_VAnpointer-
>Seeds_that_will_Germinate__Seed_Bank_:Virtual_Eelgrass_Meadow_v_2_VAnpointer-
>_array__for_delay__seeds_last[arraybuild0-1]);
   }; /* end(for,arraybuild0) */
   Virtual_Eelgrass_Meadow_v_2_VAnpointer->delay__seeds =
Virtual_Eelgrass_Meadow_v_2_VAnpointer-
>_array__for_delay__seeds[Virtual_Eelgrass_Meadow_v_2_VAnpointer->ptr_for_delay__seeds-1];
   Virtual_Eelgrass_Meadow_v_2_VAnpointer->Germinated_Seeds_0 =
(ts[2]==274?40:(int)(round(Virtual_Eelgrass_Meadow_v_2_VAnpointer->delay__seeds)));
   Rametpointer = Virtual_Eelgrass_Meadow_v_2_VAnpointer->Ramet;
   while ( Rametpointer != 0 ) {
      abort_check(this);
      Ramettype* Rametprogen;
      Rametprogen = (Ramettype*)Rametpointer->baseptrs[0];
```

```
    if (0>=(2>=phase&&Rametpointer->new_instance? -2:phase)) {
      Rametpointer->Y = (ts[2]<=1?Rametpointer->init_y:Rametpointer->My_Birthplace[2-1]);
      Rametpointer->XNW = (ts[2]<=1?Rametpointer->init_XNW:Rametpointer-
>My_Birthplace[1-1]);
      Rametpointer->DIRECTION = (Rametpointer->Di__branch==0.25?Rametpointer-
>My_Birthplace[3-1]+1.3260000000000001:(Rametpointer->Di__branch==0.75?Rametpointer-
>My_Birthplace[3-1]-1.3260000000000001:Rametpointer->My_Birthplace[3-1]));
    }; /* end(cond,0>=(2>=phase&&Rametpointer->new_instance? -2:phase)) */
    if (2>=phase) {
      Flowertype* Flowerpointer;
      Flowerpointer = &(Rametpointer->Flower);
      Specific_Growth_Ratetype* Specific_Growth_Ratepointer;
      Specific_Growth_Ratepointer = &(Rametpointer->Specific_Growth_Rate);
      Lighttype* Lightpointer;
      Lightpointer = &(Specific_Growth_Ratepointer->Light);
      Light_Seedlingtype* Light_Seedlingpointer;
      Light_Seedlingpointer = &(Lightpointer->Light_Seedling);
      Virtual_Eelgrass_Meadow_v_2_VAnpointer->Greatest_XNW_last_greatest =
max(Rametpointer->XNW,Virtual_Eelgrass_Meadow_v_2_VAnpointer-
>Greatest_XNW_last_greatest);
      Virtual_Eelgrass_Meadow_v_2_VAnpointer->Least_XNW_last_least = min(Rametpointer-
>XNW,Virtual_Eelgrass_Meadow_v_2_VAnpointer->Least_XNW_last_least);
      Virtual_Eelgrass_Meadow_v_2_VAnpointer->rcmXNWY_sum = Rametpointer-
>XNW+Virtual_Eelgrass_Meadow_v_2_VAnpointer->rcmXNWY_sum;
      Virtual_Eelgrass_Meadow_v_2_VAnpointer->Greatest_Y_last_greatest = max(Rametpointer-
>Y,Virtual_Eelgrass_Meadow_v_2_VAnpointer->Greatest_Y_last_greatest);
      Virtual_Eelgrass_Meadow_v_2_VAnpointer->Least_Y_last_least = min(Rametpointer-
>Y,Virtual_Eelgrass_Meadow_v_2_VAnpointer->Least_Y_last_least);
      Virtual_Eelgrass_Meadow_v_2_VAnpointer->rcmXNWY_sum_0 = Rametpointer-
>Y+Virtual_Eelgrass_Meadow_v_2_VAnpointer->rcmXNWY_sum_0;
      Virtual_Eelgrass_Meadow_v_2_VAnpointer->sum_BRANCH_sum = Rametpointer-
>BRANCH_0+Virtual_Eelgrass_Meadow_v_2_VAnpointer->sum_BRANCH_sum;
      Virtual_Eelgrass_Meadow_v_2_VAnpointer->Average_Iz_sum = Light_Seedlingpointer-
>Iz_SA+Virtual_Eelgrass_Meadow_v_2_VAnpointer->Average_Iz_sum;
      Rametpointer->countflowers = (Flowerpointer->flower==1?1:0);
      Virtual_Eelgrass_Meadow_v_2_VAnpointer->sumflowers_sum = Rametpointer-
>countflowers+Virtual_Eelgrass_Meadow_v_2_VAnpointer->sumflowers_sum;
      Rametpointer->flower_time = (Flowerpointer->flower==1?1:0);
      Virtual_Eelgrass_Meadow_v_2_VAnpointer->sum_LEAFbundle_sum = Rametpointer-
>LEAFBUNDLE+Virtual_Eelgrass_Meadow_v_2_VAnpointer->sum_LEAFbundle_sum;
      Rametpointer->Negative_Root_biomass_0 = (Rametpointer->ROOTS___RHIZOMES< -
0.10000000000000001?1:0);
      Rametpointer->Negative_Root_biomass = loses(Rametpointer->Negative_Root_biomass_0,2);
      Virtual_Eelgrass_Meadow_v_2_VAnpointer->sum_Roots_sum = Rametpointer-
>ROOTS___RHIZOMES+Virtual_Eelgrass_Meadow_v_2_VAnpointer->sum_Roots_sum;
      for ( arraybuild0 = 1; 1000>=arraybuild0; ++arraybuild0 ) {
        Rametpointer->_array__for_delay_death[arraybuild0-1] = (arraybuild0==Rametpointer-
>ptw_for_delay_death?Rametpointer->flower_time:Rametpointer-
>_array__for_delay_death_last[arraybuild0-1]);
        Rametpointer->_array__for_Add_a__node__0_last[arraybuild0-1] = Rametpointer-
>_array__for_Add_a__node__0[arraybuild0-1];
        Rametpointer->_array__for_Internode_Length_last[arraybuild0-1] = Rametpointer-
>_array__for_Internode_Length[arraybuild0-1];
        Rametpointer->_array__for_BRANCH_0_last[arraybuild0-1] = Rametpointer-
>_array__for_BRANCH_0[arraybuild0-1];
```

```
            Rametpointer->_array__for_BRANCH_0_0_last[arraybuild0-1] = Rametpointer-
>_array__for_BRANCH_0_0[arraybuild0-1];
            Rametpointer->_array__for_Gi_last[arraybuild0-1] = Rametpointer-
>_array__for_Gi[arraybuild0-1];
            Rametpointer->_array__for_In_last[arraybuild0-1] = Rametpointer-
>_array__for_In[arraybuild0-1];
            Rametpointer->_array__for_Pl_last[arraybuild0-1] = Rametpointer-
>_array__for_Pl[arraybuild0-1];
         }; /* end(for,arraybuild0) */
         Rametpointer->delay_death = Rametpointer->_array__for_delay_death[Rametpointer-
>ptr_for_delay_death-1];
         Rametpointer->Death_0 = Rametpointer->delay_death;
         Rametpointer->Death = loses(Rametpointer->Death_0,2);
         for ( arraybuild0 = 1; 1000>=arraybuild0; ++arraybuild0 ) {
            Rametpointer->_array__for_delay_death_last[arraybuild0-1] = Rametpointer-
>_array__for_delay_death[arraybuild0-1];
         }; /* end(for,arraybuild0) */
         Node_births_and_lengthstype* Node_births_and_lengthspointer;
         Node_births_and_lengthspointer = Rametpointer->Node_births_and_lengths;
         while ( Node_births_and_lengthspointer != 0 ) {
            abort_check(this);
            Node_births_and_lengthstype* Node_births_and_lengthsprogen;
            Node_births_and_lengthsprogen =
(Node_births_and_lengthstype*)Node_births_and_lengthspointer->baseptrs[0];
            if (0>=(2>=phase&&Node_births_and_lengthspointer->new_instance? -2:phase)) {
               Node_births_and_lengthspointer->my_internode_length =
(Node_births_and_lengthspointer->instanceid[0]==1?0:317.27999999999997*Rametpointer-
>GROWING_INTERNODE);
            }; /* end(cond,0>=(2>=phase&&Node_births_and_lengthspointer->new_instance? -
2:phase)) */
            Node_births_and_lengthspointer = Node_births_and_lengthspointer->next;
         }; /* end(while,Node_births_and_lengthspointer) */
      }; /* end(cond,2>=phase) */
      Node_births_and_lengthstype* Node_births_and_lengthspointer;
      Node_births_and_lengthspointer = Rametpointer->Node_births_and_lengths;
      while ( Node_births_and_lengthspointer != 0 ) {
         abort_check(this);
         Node_births_and_lengthstype* Node_births_and_lengthsprogen;
         Node_births_and_lengthsprogen =
(Node_births_and_lengthstype*)Node_births_and_lengthspointer->baseptrs[0];
         Rametpointer->average_internode_length_sum = Node_births_and_lengthspointer-
>my_internode_length+Rametpointer->average_internode_length_sum;
         Node_births_and_lengthspointer = Node_births_and_lengthspointer->next;
      }; /* end(while,Node_births_and_lengthspointer) */
      Rametpointer->average_internode_length = (Rametpointer-
>my_number__of_nodes>0?(double)(Rametpointer->average_internode_length_sum)/Rametpointer-
>my_number__of_nodes:0);
      Rametpointer->Decomp = (ts[2]<2 || Rametpointer-
>my_number__of_nodes<1?0:(fmod(Rametpointer-
>my_number__of_nodes,12)==0?(double)(Rametpointer-
>average_internode_length*12)/317.27999999999997:0));
      Node_births_and_lengthstype* Node_births_and_lengthspointer_0;
      Node_births_and_lengthspointer_0 = Rametpointer->Node_births_and_lengths;
      while ( Node_births_and_lengthspointer_0 != 0 ) {
         abort_check(this);
```

```
         Node_births_and_lengthstype* Node_births_and_lengthsprogen;
         Node_births_and_lengthsprogen =
(Node_births_and_lengthstype*)Node_births_and_lengthspointer_0->baseptrs[0];
         if (2>=phase) {
            Rametpointer->Rhizome_Length_sum = Node_births_and_lengthspointer_0-
>my_internode_length+Rametpointer->Rhizome_Length_sum;
         }; /* end(cond,2>=phase) */
         Node_births_and_lengthspointer_0 = Node_births_and_lengthspointer_0->next;
      }; /* end(while,Node_births_and_lengthspointer_0) */
      Rametpointer = Rametpointer->next;
   }; /* end(while,Rametpointer) */
   Rametpointer = Virtual_Eelgrass_Meadow_v_2_VAnpointer->Ramet;
   while ( Rametpointer != 0 ) {
      abort_check(this);
      Ramettype* Rametprogen;
      Rametprogen = (Ramettype*)Rametpointer->baseptrs[0];
      if (0>=(2>=phase&&Rametpointer->new_instance? -2:phase)) {
      }; /* end(cond,0>=(2>=phase&&Rametpointer->new_instance? -2:phase)) */
      if (2>=phase) {
         Rametpointer->Rhizome_Length = Rametpointer->Rhizome_Length_sum;
         Virtual_Eelgrass_Meadow_v_2_VAnpointer->Total_Rhizome__Length_sum = Rametpointer-
>Rhizome_Length+Virtual_Eelgrass_Meadow_v_2_VAnpointer->Total_Rhizome__Length_sum;
         Virtual_Eelgrass_Meadow_v_2_VAnpointer->Average_Internode__Length_sum =
Rametpointer->average_internode_length+Virtual_Eelgrass_Meadow_v_2_VAnpointer-
>Average_Internode__Length_sum;
         Node_births_and_lengthstype* Node_births_and_lengthspointer;
         Node_births_and_lengthspointer = Rametpointer->Node_births_and_lengths;
         while ( Node_births_and_lengthspointer != 0 ) {
            abort_check(this);
            Node_births_and_lengthstype* Node_births_and_lengthsprogen;
            Node_births_and_lengthsprogen =
(Node_births_and_lengthstype*)Node_births_and_lengthspointer->baseptrs[0];
            if (0>=(2>=phase&&Node_births_and_lengthspointer->new_instance? -2:phase)) {
               Node_births_and_lengthspointer->NodeXNW = (Node_births_and_lengthspointer-
>instanceid[0]==2?17:(Node_births_and_lengthspointer-
>instanceid[0]==3?34:(Node_births_and_lengthspointer->instanceid[0]==4?51:Rametpointer-
>XNW)));
               Node_births_and_lengthspointer->NodeY = (Node_births_and_lengthspointer-
>instanceid[0]<5?0:Rametpointer->Y);
            }; /* end(cond,0>=(2>=phase&&Node_births_and_lengthspointer->new_instance? -
2:phase)) */
            Node_births_and_lengthspointer = Node_births_and_lengthspointer->next;
         }; /* end(while,Node_births_and_lengthspointer) */
      }; /* end(cond,2>=phase) */
      Rametpointer = Rametpointer->next;
   }; /* end(while,Rametpointer) */
   Eve_Plastochrone_Intervalstype* Eve_Plastochrone_Intervalspointer;
   Eve_Plastochrone_Intervalspointer = Virtual_Eelgrass_Meadow_v_2_VAnpointer-
>Eve_Plastochrone_Intervals;
   while ( Eve_Plastochrone_Intervalspointer != 0 ) {
      abort_check(this);
      Ramettype* EvetaggingRametptr;
      EvetaggingRametptr = (Ramettype*)Eve_Plastochrone_Intervalspointer->baseptrs[0];
      if (2>=phase) {
```

```
        Eve_Plastochrone_Intervalspointer->below = EvetaggingRametptr-
>GROWING_INTERNODE+EvetaggingRametptr->ROOTS___RHIZOMES;
        EveStatspointer->avgbelow_sum = Eve_Plastochrone_Intervalspointer-
>below+EveStatspointer->avgbelow_sum;
        EveStatspointer->belowstdev_sum_0 = Eve_Plastochrone_Intervalspointer-
>below+EveStatspointer->belowstdev_sum_0;
        Eve_Plastochrone_Intervalspointer->above = EvetaggingRametptr-
>OLDESTLEAF+EvetaggingRametptr->LEAFBUNDLE;
        EveStatspointer->aboveavg_sum = Eve_Plastochrone_Intervalspointer-
>above+EveStatspointer->aboveavg_sum;
        EveStatspointer->abovestdev_sum_0 = Eve_Plastochrone_Intervalspointer-
>above+EveStatspointer->abovestdev_sum_0;
        Eve_Plastochrone_Intervalspointer->NewNodes = EvetaggingRametptr-
>OLD__NODES+EvetaggingRametptr->NODES;
        Eve_Plastochrone_Intervalspointer->Nodessquared = Eve_Plastochrone_Intervalspointer-
>NewNodes*Eve_Plastochrone_Intervalspointer->NewNodes;
        EveStatspointer->stdevnodes_sum_0 = Eve_Plastochrone_Intervalspointer-
>Nodessquared+EveStatspointer->stdevnodes_sum_0;
        EveStatspointer->mean_number__of_new_nodes_sum = Eve_Plastochrone_Intervalspointer-
>NewNodes+EveStatspointer->mean_number__of_new_nodes_sum;
        EveStatspointer->Greatest_New_Nodes_greatest = max(Eve_Plastochrone_Intervalspointer-
>NewNodes,EveStatspointer->Greatest_New_Nodes_greatest);
        EveStatspointer->stdevnodes_sum_1 = Eve_Plastochrone_Intervalspointer-
>NewNodes+EveStatspointer->stdevnodes_sum_1;
        EveStatspointer->leastnodes_least = min(Eve_Plastochrone_Intervalspointer-
>NewNodes,EveStatspointer->leastnodes_least);
      }; /* end(cond,2>=phase) */
      Eve_Plastochrone_Intervalspointer = Eve_Plastochrone_Intervalspointer->next;
    }; /* end(while,Eve_Plastochrone_Intervalspointer) */
    if (1>=phase) {
      Virtual_Eelgrass_Meadow_v_2_VAnpointer = &(this->Virtual_Eelgrass_Meadow_v_2_VAn);
      int Neighbourscond;
      Neighbourstype** Neighboursmeta;
      Neighboursmeta = &(Virtual_Eelgrass_Meadow_v_2_VAnpointer->Neighbours);
      Ramettype* My_NeighbourRametptr;
      My_NeighbourRametptr = Virtual_Eelgrass_Meadow_v_2_VAnpointer->Ramet;
      while ( My_NeighbourRametptr != 0 ) {
        abort_check(this);
        Ramettype* Rametprogen;
        Rametprogen = (Ramettype*)My_NeighbourRametptr->baseptrs[0];
        Ramettype* MERametptr;
        MERametptr = Virtual_Eelgrass_Meadow_v_2_VAnpointer->Ramet;
        while ( MERametptr != 0 ) {
          abort_check(this);
          Ramettype* Rametprogen;
          Rametprogen = (Ramettype*)MERametptr->baseptrs[0];
          Neighbourstype* Neighbourspointer;
          abort_check(this);
          if (prune(Neighboursmeta, 2, My_NeighbourRametptr->instanceid[0], MERametptr-
>instanceid[0])) {
            Neighbourspointer = *Neighboursmeta;
            Neighbourspointer->new_instance = 0;
            *Neighboursmeta = Neighbourspointer->next;
          } else { /* Instance exists */
            Neighbourspointer = new Neighbourstype;
```

```
            Neighbourspointer->instanceid[0] = My_NeighbourRametptr->instanceid[0];
            Neighbourspointer->instanceid[1] = MERametptr->instanceid[0];
            Neighbourspointer->baseptrs[1] = MERametptr;
            Neighbourspointer->baseptrs[0] = My_NeighbourRametptr;
            Neighbourspointer->new_instance = 1;
          }; /* end(cond,Instance exists) */
          Neighbourspointer->cond1 = fmod(ts[1],10)==0&&My_NeighbourRametptr-
>XNW<=MERametptr->XNW+0.10000000000000001&&My_NeighbourRametptr-
>XNW>MERametptr->XNW-0.10000000000000001&&My_NeighbourRametptr->Y<=MERametptr-
>Y+0.10000000000000001&&My_NeighbourRametptr->Y>MERametptr->Y-
0.10000000000000001;
          if (Neighbourspointer->cond1) {
            Neighbourspointer->next = *Neighboursmeta;
            *Neighboursmeta = Neighbourspointer;
            Neighboursmeta = &(Neighbourspointer->next);
            My_NeighbourRametptr->Timed_Square_Meter_Density_count =
My_NeighbourRametptr->Timed_Square_Meter_Density_count+1;
          } else { /* Neighbourspointer->cond1 */
            delete Neighbourspointer;
          }; /* end(cond,Neighbourspointer->cond1) */
          MERametptr = MERametptr->next;
        }; /* end(while,MERametptr) */
        My_NeighbourRametptr = My_NeighbourRametptr->next;
      }; /* end(while,My_NeighbourRametptr) */
      delete_list(*Neighboursmeta);
      *Neighboursmeta = 0;
      Neighbourstype* Neighbourspointer;
      Neighbourspointer = Virtual_Eelgrass_Meadow_v_2_VAnpointer->Neighbours;
      while ( Neighbourspointer != 0 ) {
        abort_check(this);
        Ramettype* MERametptr;
        Ramettype* My_NeighbourRametptr;
        MERametptr = (Ramettype*)Neighbourspointer->baseptrs[1];
        My_NeighbourRametptr = (Ramettype*)Neighbourspointer->baseptrs[0];
        if (-2>=(1>=phase&&Neighbourspointer->new_instance? -2:phase)) {
          Neighbourspointer->index_0 = Neighbourspointer->instanceid[1];
        }; /* end(cond,-2>=(1>=phase&&Neighbourspointer->new_instance? -2:phase)) */
        Neighbourspointer = Neighbourspointer->next;
      }; /* end(while,Neighbourspointer) */
    }; /* end(cond,1>=phase) */
    if (2>=phase) {
      Virtual_Eelgrass_Meadow_v_2_VAnpointer = &(this->Virtual_Eelgrass_Meadow_v_2_VAn);
      EveStatstype* EveStatspointer_0;
      EveStatspointer_0 = &(Virtual_Eelgrass_Meadow_v_2_VAnpointer->EveStats);
      EveStatspointer_0->leastnodes = EveStatspointer_0->leastnodes_least;
      EveStatspointer_0->Greatest_New_Nodes = EveStatspointer_0->Greatest_New_Nodes_greatest;
      EveStatspointer_0->mean_number__of_new_nodes = (double)(EveStatspointer_0-
>mean_number__of_new_nodes_sum)/EveStatspointer_0->mean_number__of_new_nodes_sum_0;
      EveStatspointer_0->stdevnodes = sqrt((double)(EveStatspointer_0-
>stdevnodes_sum*EveStatspointer_0->stdevnodes_sum_0-EveStatspointer_0-
>stdevnodes_sum_1*EveStatspointer_0->stdevnodes_sum_1)/(EveStatspointer_0-
>stdevnodes_sum*(EveStatspointer_0->stdevnodes_sum-1)));
      EveStatspointer_0->abovestdev = sqrt((double)(EveStatspointer_0-
>abovestdev_sum*EveStatspointer_0->abovestdev_sum_0-EveStatspointer_0-
```

```
>abovestdev_sum_0*EveStatspointer_0->abovestdev_sum_0)/(EveStatspointer_0-
>abovestdev_sum*(EveStatspointer_0->abovestdev_sum-1)));
      EveStatspointer_0->aboveavg = (double)(EveStatspointer_0->aboveavg_sum)/EveStatspointer_0-
>aboveavg_count;
      EveStatspointer_0->belowstdev = sqrt((double)(EveStatspointer_0-
>belowstdev_sum*EveStatspointer_0->belowstdev_sum_0-EveStatspointer_0-
>belowstdev_sum_0*EveStatspointer_0->belowstdev_sum_0)/(EveStatspointer_0-
>belowstdev_sum*(EveStatspointer_0->belowstdev_sum-1)));
      EveStatspointer_0->avgbelow = (double)(EveStatspointer_0-
>avgbelow_sum)/EveStatspointer_0->avgbelow_count;
      Virtual_Eelgrass_Meadow_v_2_VAnpointer->Average_Internode__Length =
(double)(Virtual_Eelgrass_Meadow_v_2_VAnpointer-
>Average_Internode__Length_sum)/Virtual_Eelgrass_Meadow_v_2_VAnpointer-
>Average_Internode__Length_count;
      Virtual_Eelgrass_Meadow_v_2_VAnpointer->Total_Rhizome__Length =
(double)(Virtual_Eelgrass_Meadow_v_2_VAnpointer->Total_Rhizome__Length_sum)/1000;
      Virtual_Eelgrass_Meadow_v_2_VAnpointer->sum_Roots =
Virtual_Eelgrass_Meadow_v_2_VAnpointer->sum_Roots_sum;
      Virtual_Eelgrass_Meadow_v_2_VAnpointer->sum_LEAFbundle =
Virtual_Eelgrass_Meadow_v_2_VAnpointer->sum_LEAFbundle_sum;
      Virtual_Eelgrass_Meadow_v_2_VAnpointer->abovebelow_ratio =
(double)(Virtual_Eelgrass_Meadow_v_2_VAnpointer-
>sum_LEAFbundle)/Virtual_Eelgrass_Meadow_v_2_VAnpointer->sum_Roots;
      Virtual_Eelgrass_Meadow_v_2_VAnpointer->total_biomass =
Virtual_Eelgrass_Meadow_v_2_VAnpointer-
>sum_Roots+Virtual_Eelgrass_Meadow_v_2_VAnpointer->sum_LEAFbundle;
      Virtual_Eelgrass_Meadow_v_2_VAnpointer->sumflowers =
Virtual_Eelgrass_Meadow_v_2_VAnpointer->sumflowers_sum;
      Virtual_Eelgrass_Meadow_v_2_VAnpointer->Average_Iz =
(double)(Virtual_Eelgrass_Meadow_v_2_VAnpointer-
>Average_Iz_sum)/Virtual_Eelgrass_Meadow_v_2_VAnpointer->Average_Iz_count;
      Virtual_Eelgrass_Meadow_v_2_VAnpointer->sum_BRANCH =
Virtual_Eelgrass_Meadow_v_2_VAnpointer->sum_BRANCH_sum;
      Virtual_Eelgrass_Meadow_v_2_VAnpointer->Least_Y_last =
Virtual_Eelgrass_Meadow_v_2_VAnpointer->Least_Y_last_least;
      Virtual_Eelgrass_Meadow_v_2_VAnpointer->Greatest_Y_last =
Virtual_Eelgrass_Meadow_v_2_VAnpointer->Greatest_Y_last_greatest;
      Virtual_Eelgrass_Meadow_v_2_VAnpointer->Least_XNW_last =
Virtual_Eelgrass_Meadow_v_2_VAnpointer->Least_XNW_last_least;
      Virtual_Eelgrass_Meadow_v_2_VAnpointer->Greatest_XNW_last =
Virtual_Eelgrass_Meadow_v_2_VAnpointer->Greatest_XNW_last_greatest;
      for ( arraybuild0 = 1; 1000>=arraybuild0; ++arraybuild0 ) {
         Virtual_Eelgrass_Meadow_v_2_VAnpointer->_array__for_delay__seeds_last[arraybuild0-1] =
Virtual_Eelgrass_Meadow_v_2_VAnpointer->_array__for_delay__seeds[arraybuild0-1];
      }; /* end(for,arraybuild0) */
      int loop;
      for ( loop = 1; 2>=loop; ++loop ) {
         Virtual_Eelgrass_Meadow_v_2_VAnpointer->rcmXNWY[loop-1] =
(loop==1?(double)(Virtual_Eelgrass_Meadow_v_2_VAnpointer-
>rcmXNWY_sum)/Virtual_Eelgrass_Meadow_v_2_VAnpointer-
>rcmXNWY_count:(double)(Virtual_Eelgrass_Meadow_v_2_VAnpointer-
>rcmXNWY_sum_0)/Virtual_Eelgrass_Meadow_v_2_VAnpointer->rcmXNWY_count_0);
      }; /* end(for,loop) */
      Rametpointer = Virtual_Eelgrass_Meadow_v_2_VAnpointer->Ramet;
      while ( Rametpointer != 0 ) {
```

```
        abort_check(this);
        Ramettype* Rametprogen;
        Rametprogen = (Ramettype*)Rametpointer->baseptrs[0];
        if (1>=(2>=phase&&Rametpointer->new_instance? -2:phase)) {
        }; /* end(cond,1>=(2>=phase&&Rametpointer->new_instance? -2:phase)) */
        Rametpointer->Squared_Distance_to_rcm = pow(Rametpointer->XNW-
Virtual_Eelgrass_Meadow_v_2_VAnpointer->rcmXNWY[1-1],2)+pow(Rametpointer->Y-
Virtual_Eelgrass_Meadow_v_2_VAnpointer->rcmXNWY[2-1],2);
        Virtual_Eelgrass_Meadow_v_2_VAnpointer->Radius_of_Gyration_sum = Rametpointer-
>Squared_Distance_to_rcm+Virtual_Eelgrass_Meadow_v_2_VAnpointer->Radius_of_Gyration_sum;
        Rametpointer->ptw_for_Add_a__node__0_last = Rametpointer->ptw_for_Add_a__node__0;
        Rametpointer->ptw_for_Square_Meter__Density_last = Rametpointer-
>ptw_for_Square_Meter__Density;
        Rametpointer->ptw_for_Internode_Length_last = Rametpointer->ptw_for_Internode_Length;
        Rametpointer->ptw_for_BRANCH_0_last = Rametpointer->ptw_for_BRANCH_0;
        Rametpointer->ptw_for_BRANCH_0_0_last = Rametpointer->ptw_for_BRANCH_0_0;
        Rametpointer->ptw_for_Gi_last = Rametpointer->ptw_for_Gi;
        Rametpointer->ptw_for_In_last = Rametpointer->ptw_for_In;
        Rametpointer->init_XNW_last = Rametpointer->XNW;
        Rametpointer->init_y_last = Rametpointer->Y;
        Rametpointer->ptw_for_Pl_last = Rametpointer->ptw_for_Pl;
        int loop_0;
        for ( loop_0 = 1; 3>=loop_0; ++loop_0 ) {
           Rametpointer->MyCoords_last[loop_0-1] = (loop_0==1?Rametpointer-
>XNW:(loop_0==2?Rametpointer->Y:Rametpointer->DIRECTION));
        }; /* end(for,loop_0) */
        Rametpointer = Rametpointer->next;
      }; /* end(while,Rametpointer) */
      Virtual_Eelgrass_Meadow_v_2_VAnpointer->Radius_of_Gyration =
sqrt((double)(Virtual_Eelgrass_Meadow_v_2_VAnpointer-
>Radius_of_Gyration_sum)/Virtual_Eelgrass_Meadow_v_2_VAnpointer-
>Radius_of_Gyration_count);
   }; /* end(cond,2>=phase) */
   Virtual_Eelgrass_Meadow_v_2_VAnpointer = &(this->Virtual_Eelgrass_Meadow_v_2_VAn);
   Rametpointer = Virtual_Eelgrass_Meadow_v_2_VAnpointer->Ramet;
   while ( Rametpointer != 0 ) {
      abort_check(this);
      Ramettype* Rametprogen;
      Rametprogen = (Ramettype*)Rametpointer->baseptrs[0];
      Specific_Growth_Ratetype* Specific_Growth_Ratepointer;
      Specific_Growth_Ratepointer = &(Rametpointer->Specific_Growth_Rate);
      Lighttype* Lightpointer;
      Lightpointer = &(Specific_Growth_Ratepointer->Light);
      Light_Adulttype* Light_Adultpointer;
      Light_Adultpointer = &(Lightpointer->Light_Adult);
      Light_Seedlingtype* Light_Seedlingpointer;
      Light_Seedlingpointer = &(Lightpointer->Light_Seedling);
      Flowertype* Flowerpointer;
      Flowerpointer = &(Rametpointer->Flower);
      Adulthoodtype* Adulthoodpointer;
      Adulthoodpointer = &(Rametpointer->Adulthood);
      Grass_Machinetype* Grass_Machinepointer;
      Grass_Machinepointer = &(Specific_Growth_Ratepointer->Grass_Machine);
      SGR_Adulttype* SGR_Adultpointer;
      SGR_Adultpointer = &(Grass_Machinepointer->SGR_Adult);
```

```
      Rametpointer->Timed_Square_Meter_Density = (ts[2]<=1?Rametpointer-
>Initial_Density:(fmod(ts[2],10)==0?Rametpointer->Timed_Square_Meter_Density_count*25:0));
      Rametpointer->Canopy_Height__for_Light = (ts[2]<=1 || ts[2]==Rametpointer-
>Canopy_Height__for_Light_at_phase?30:28.550000000000001*pow((Adulthoodpointer-
>is_adult?Light_Adultpointer->Percent_Irradiance_A:Light_Seedlingpointer->Percent_Irradiance_S),-
0.59999999999999998));
      Rametpointer->maol = (Adulthoodpointer->is_adult?Rametpointer-
>Nutrient__Canopy_Factor*((double)(Rametpointer-
>Canopy_Height__for_Light*0.001199999999999999)/1.2):0.014999999999999999*1.2);
      Rametpointer->malb = (Adulthoodpointer->is_adult?Rametpointer-
>maol*3.7000000000000002:Rametpointer->maol);
      Rametpointer->Ol = (Rametpointer->LEAFBUNDLE>Rametpointer->malb?Rametpointer-
>LEAFBUNDLE-Rametpointer->malb:0);
      Rametpointer->Sl = (Rametpointer->OLDESTLEAF>Rametpointer->maol?Rametpointer-
>OLDESTLEAF-Rametpointer->maol:0);
      for ( arraybuild0 = 1; 1000>=arraybuild0; ++arraybuild0 ) {
         Rametpointer->_array__for_Square_Meter__Density[arraybuild0-1] =
(arraybuild0==Rametpointer->ptw_for_Square_Meter__Density?Rametpointer-
>Timed_Square_Meter_Density:Rametpointer-
>_array__for_Square_Meter__Density_last[arraybuild0-1]);
      }; /* end(for,arraybuild0) */
      Rametpointer->Square_Meter__Density = (ts[2]<=1?75:(fmod(ts[2],10)==0?Rametpointer-
>Timed_Square_Meter_Density:Rametpointer-
>_array__for_Square_Meter__Density[(int)(Rametpointer->ptr_for_Square_Meter__Density)-1]));
      Light_Adultpointer->kcanopy_A =
(ts[2]<=1?0:2.0899999999999999+0.00018000000000000001*Rametpointer-
>Square_Meter__Density*0.01*Rametpointer->Canopy_Height__for_Light);
      Light_Adultpointer->Izleaf_A = (ts[2]<=1 || ts[2]==Light_Adultpointer-
>Izleaf_A_at_phase?Light_Seedlingpointer->Iz_SA:Light_Seedlingpointer->Iz_SA*exp(-
1*Light_Adultpointer->kcanopy_A*0.0050000000000000001*Rametpointer-
>Canopy_Height__for_Light));
      SGR_Adultpointer->Set_GMAXNW_A = (double)(SGR_Adultpointer-
>umax_A*SGR_Adultpointer->alpha_A*Light_Adultpointer->Izleaf_A)/(SGR_Adultpointer-
>umax_A+SGR_Adultpointer->alpha_A*Light_Adultpointer->Izleaf_A)+SGR_Adultpointer->ro_A;
      SGR_Adultpointer->SGR_A = (SGR_Adultpointer-
>Set_GMAXNW_A>=0.059999999999999998?0.059999999999999998:(SGR_Adultpointer-
>Set_GMAXNW_A<= -0.059999999999999998?-0.059999999999999998:SGR_Adultpointer-
>Set_GMAXNW_A*SGR_Adultpointer->sediment__limitation_A));
      Rametpointer->GI_A = (Flowerpointer->flower==1?0:SGR_Adultpointer->SGR_A);
      Rametpointer->Gl = Rametpointer->LEAFBUNDLE*(Adulthoodpointer-
>is_adult?Rametpointer->GI_A:Rametpointer->GI_S);
      Rametpointer->Gr = (Flowerpointer->flower==1?0:(double)(Rametpointer->b*Rametpointer-
>Gl)/(1-Rametpointer->b));
      Light_Adultpointer->Percent_Irradiance_A = (ts[2]<=1 || Rametpointer-
>Square_Meter__Density<=220?(double)(Light_Seedlingpointer->Iz_SA)/Light_Seedlingpointer-
>Surface_PAR_S:(double)(Light_Adultpointer->Izleaf_A)/Light_Seedlingpointer-
>Surface_PAR_S+0.050000000000000003);
      Rametpointer = Rametpointer->next;
   }; /* end(while,Rametpointer) */
   Ramettype* Rametpointer_0;
   Rametpointer_0 = Virtual_Eelgrass_Meadow_v_2_VAnpointer->Ramet;
   while ( Rametpointer_0 != 0 ) {
      abort_check(this);
      Ramettype* Rametprogen;
      Rametprogen = (Ramettype*)Rametpointer_0->baseptrs[0];
```

```
      if (2>=phase) {
         Specific_Growth_Ratetype* Specific_Growth_Ratepointer;
         Specific_Growth_Ratepointer = &(Rametpointer_0->Specific_Growth_Rate);
         Lighttype* Lightpointer;
         Lightpointer = &(Specific_Growth_Ratepointer->Light);
         Light_Adulttype* Light_Adultpointer;
         Light_Adultpointer = &(Lightpointer->Light_Adult);
         Temperaturetype* Temperaturepointer;
         Temperaturepointer = &(Specific_Growth_Ratepointer->Temperature);
         Rametpointer_0->check_b_adult___fix = (Temperaturepointer-
>Temperature_SA<8.064000000000001?0.75:(double)(Rametpointer_0->Gr)/(Rametpointer_0-
>Gr+Rametpointer_0->Gl));
         Virtual_Eelgrass_Meadow_v_2_VAnpointer->averageIleaf_sum = Light_Adultpointer-
>Izleaf_A+Virtual_Eelgrass_Meadow_v_2_VAnpointer->averageIleaf_sum;
         Virtual_Eelgrass_Meadow_v_2_VAnpointer->avgkcanopy_sum = Light_Adultpointer-
>kcanopy_A+Virtual_Eelgrass_Meadow_v_2_VAnpointer->avgkcanopy_sum;
         Rametpointer_0->Canopy_Height = Rametpointer_0-
>Nutrient__Canopy_Factor*Rametpointer_0->Canopy_Height__for_Light;
         Virtual_Eelgrass_Meadow_v_2_VAnpointer->averageLL_sum = Rametpointer_0-
>Canopy_Height__for_Light+Virtual_Eelgrass_Meadow_v_2_VAnpointer->averageLL_sum;
         Virtual_Eelgrass_Meadow_v_2_VAnpointer->Average_Density_sum = Rametpointer_0-
>Timed_Square_Meter_Density+Virtual_Eelgrass_Meadow_v_2_VAnpointer-
>Average_Density_sum;
         for ( arraybuild0 = 1; 1000>=arraybuild0; ++arraybuild0 ) {
            Rametpointer_0->_array__for_Square_Meter__Density_last[arraybuild0-1] =
Rametpointer_0->_array__for_Square_Meter__Density[arraybuild0-1];
         }; /* end(for,arraybuild0) */
      }; /* end(cond,2>=phase) */
      Rametpointer_0 = Rametpointer_0->next;
   }; /* end(while,Rametpointer_0) */
   if (2>=phase) {
      Virtual_Eelgrass_Meadow_v_2_VAnpointer = &(this->Virtual_Eelgrass_Meadow_v_2_VAn);
      Virtual_Eelgrass_Meadow_v_2_VAnpointer->Average_Density =
(double)(Virtual_Eelgrass_Meadow_v_2_VAnpointer-
>Average_Density_sum)/Virtual_Eelgrass_Meadow_v_2_VAnpointer->Average_Density_count;
      Virtual_Eelgrass_Meadow_v_2_VAnpointer->averageLL =
(double)(Virtual_Eelgrass_Meadow_v_2_VAnpointer-
>averageLL_sum)/Virtual_Eelgrass_Meadow_v_2_VAnpointer->averageLL_count;
      Virtual_Eelgrass_Meadow_v_2_VAnpointer->avgkcanopy =
(double)(Virtual_Eelgrass_Meadow_v_2_VAnpointer-
>avgkcanopy_sum)/Virtual_Eelgrass_Meadow_v_2_VAnpointer->avgkcanopy_count;
      Virtual_Eelgrass_Meadow_v_2_VAnpointer->averageIleaf =
(double)(Virtual_Eelgrass_Meadow_v_2_VAnpointer-
>averageIleaf_sum)/Virtual_Eelgrass_Meadow_v_2_VAnpointer->averageIleaf_count;
   }; /* end(cond,2>=phase) */

}; /* end(procedure,evalmodel) */

   /* FREE ALL DATA STRUCTURES */

void do_exitmodel () {

/*    namespace delete AME_model
 */
```

```
}; /* end(procedure,dummy) */
int do_evalmodel(int);
}; /* end(class,AME_model) */
char I_am_Eve_name[] = "I am Eve";
char I_am_Eve_spec[] = "Eve_Status_Evetagging==1";
char Eve_PI_name[] = "Eve PI";
char Eve_PI_spec[] = "Plastochrone_Interval_Evetagging";
char count_name[] = "count";
char count_spec[] = "one_Evetagging";
char NewNodes_name[] = "NewNodes";
char NewNodes_spec[] = "(OLD__NODES_Evetagging+NODES_Evetagging)";
char Leaves_Shed_name[] = "Leaves Shed";
char Leaves_Shed_spec[] = "LeavesProduced_Evetagging";
char Nodessquared_name[] = "Nodessquared";
char Nodessquared_spec[] = "NewNodes*NewNodes";
char above_name[] = "above";
char above_spec[] = "OLDESTLEAF_Evetagging+LEAFBUNDLE_Evetagging";
char below_name[] = "below";
char below_spec[] =
"GROWING_INTERNODE___Evetagging+ROOTS___RHIZOMES_Evetagging";
char Eve_Plastochrone_Intervals_name[] = "Eve Plastochrone Intervals";
char Tagged_PI_name[] = "Tagged PI";
char GreatestPI_name[] = "GreatestPI";
char mean_number__of_new_nodes_name[] = "mean number \n\
of new nodes";
char mean_number__of_new_nodes_spec[] = "(sum({NewNodes}))/(sum({count}))";
char Greatest_New_Nodes_name[] = "Greatest New Nodes";
char Greatest_New_Nodes_spec[] = "greatest({NewNodes})";
char mean_new_leaves_name[] = "mean new leaves";
char mean_new_leaves_spec[] = "(sum({Leaves_Shed}))/(sum({count}))";
char stdevnodes_name[] = "stdevnodes";
char stdevnodes_spec[] = "sqrt(((sum({count})*sum({Nodessquared}))-
(sum({NewNodes})*sum({NewNodes})))/(sum({count})*(sum({count})-1)))";
char leastnodes_name[] = "leastnodes";
char leastnodes_spec[] = "least({NewNodes})";
char aboveavg_name[] = "aboveavg";
char aboveavg_spec[] = "(sum({above}))/(count({above}))";
char abovestdev_name[] = "abovestdev";
char abovestdev_spec[] = "sqrt(((sum({count})*sum({above}))-
(sum({above})*sum({above})))/(sum({count})*(sum({count})-1)))";
char avgbelow_name[] = "avgbelow";
char avgbelow_spec[] = "(sum({below}))/(count({below}))";
char belowstdev_name[] = "belowstdev";
char belowstdev_spec[] = "sqrt(((sum({count})*sum({below}))-
(sum({below})*sum({below})))/(sum({count})*(sum({count})-1)))";
char EveStats_name[] = "EveStats";
char water_depth_SA_name[] = "water depth_SA";
char water_depth_SA_spec[] = "0.59999999999999998";
char water_depth_SA_comment[] = "units meters.";
char sediments_SA_name[] = "sediments_SA";
char sediments_SA_spec[] = "50";
char sediments_SA_description[] = "Sediment sulfide concentration";
char sediments_SA_comment[] = "Units are uMol sulfide.  Will eventually take this and use regression
to relate to sediment grain size characteristics.";
char Nitrogen_SA_name[] = "Nitrogen_SA";
```

```
char Nitrogen_SA_spec[] = "0.0085400000000000007";
char Nitrogen_SA_description[] = "g N/m2 as in Roberts et al. paper";
char Nitrogen_SA_comment[] = "Units are g N/m2/day";
char k_SA_name[] = "k_SA";
char k_SA_spec[] = "table(int(time()))";
char k_SA_comment[] = "light attenuation coefficient units m-1. Taken from M.Brush averaged k-
value for HIB.";
char Physical_Setting_name[] = "Physical Setting";
char month_name[] = "month";
char month_spec[] = "if dayofyear<32 then 1 elseif dayofyear<60 then 2 elseif dayofyear<91 then 3
elseif dayofyear<121 then 4 elseif dayofyear<152 then 5 elseif dayofyear<182 then 6 elseif
dayofyear<213 then 7 elseif dayofyear<244 then 8 elseif dayofyear<274 then 9 elseif dayofyear<305
then 10 elseif dayofyear<335 then \
11 else 12";
char month_description[] = "month #";
char month_comment[] = "For plotting purposes";
char days_name[] = "days";
char days_spec[] = "int(time(1))";
char year_name[] = "year";
char year_spec[] = "int((days-1)/365)+1";
char dayofyear_name[] = "dayofyear";
char dayofyear_spec[] = "days-(year-1)*365";
char dayreal_name[] = "dayreal";
char clock_name[] = "clock";
char clock_comment[] = "Submodel to track time";
char flow1_name[] = "flow1";
char flow1_spec[] = "if Temperature_SA>20 then 1 else 0";
char daysabove_20box_name[] = "daysabove_20box";
char daysabove_20box_spec[] = "0";
char Temperature_SA_name[] = "Temperature_SA";
char Temperature_SA_spec[] = "table(int(time()))";
char Temperature_SA_description[] = "Temperature degrees celsius";
char Temperature_SA_comment[] = "Taken from Short 1980 model. for Ninigret Pond.  Units are
degrees Celsius.\n\
\n\
I should change this to use table data? JF";
char Daysabove20_name[] = "Daysabove20";
char Daysabove20_spec[] = "90";
char Daysabove20_comment[] = "Counts how many days are above 20degrees C in the year (or an
average year). This correlates to temperature stress, when respiration increases more than
photosynthesis. \n\
\n\
Need to figure out how to have this parameter connect directly to the Temperature table to actually
count how many days are above 20 to make the model more seamless with less necessary inputs. Right
now you have to count.\n\
\n\
Around the world, average optimal growth temperatures are ~15-20°C, above which productivity
begins to decrease due to the dramatic effect of temperature on respiration (Marsh et al. 1986). The
optimum temperature for photosynthesis and growth is commonly based on measurements taken in
saturating light \
conditions. \n\
\n\
\"So, on the whole I would suggest floral induction of  Z. marina is more likely in individuals at a
higher metabolic state/most likely size (growing well).\"\n\
\n\
```

The model output matches literature valueswhich indicate that as water temperatures increase above 20◦C Z.marina respiration increases at a greater rate than photosynthesiscausing stress and eventually mortality when water temperaturesare greater than 25◦C (Marsh et al., 1986; Nejrup et al., 2008). \
Jarvis(2014)\n\
\n\
\n\
GB: ~50 days above 20 per year\n\
VA South Bay: ~126 days above 20 per year\n\
NC ~180 days above 20 per year";
char FDegreeDays_name[] = "FDegreeDays";
char FDegreeDays_spec[] = "0";
char FDegreeDays_comment[] = "Daily temperature readings can be used to calculate growing degree-days, which is a measure of accumulated heat. Since plant development is temperature-dependent, phenological events of plants can also be used to track degree-days…\n\
Full bloom: date 95% of flowers have opened (e.g. 1 out of 20 buds remains closed).\n\
First bloom: date first flower on the plant opens to reveal pistils and / or stamens.\n\
\n\
I simply just need more data on eelgrass flowering first blooms and full bloom to properly parameterize this variable.\n\
\n\
I wish I could make the coding here more elegant, however, whatever date it is on the 3rd day above 15 degrees C that is the flowering date.\n\
\n\
VA South Bay: Flowers on day 103";
char FDays_name[] = "FDays";
char FDays_spec[] = "if Temperature_SA>15 then 1 else 0";
char GDegreeDays_name[] = "GDegreeDays";
char GDegreeDays_spec[] = "0";
char GDays_name[] = "GDays";
char GDays_spec[] = "if Temperature_SA<20 and time(0)>244 then 1 else 0";
char Temperature_name[] = "Temperature";
char sediment__limitation_A_name[] = "sediment\n\
 limitation_A";
char sediment__limitation_A_spec[] = "if sediments_SA<=55.45 then 1 elseif sediments_SA>=2000 then 0 else 13.6*(sediments_SA^-.65)";
char sediment__limitation_A_description[] = "Sediment Sulfide limitation";
char sediment__limitation_A_comment[] = "Taken From Goodman thesis. Converted from units of oxygen/min/dm2. Scaled from 0 to 1 as limitation factor.";
char SGR_A_name[] = "SGR_A";
char SGR_A_spec[] = "if Set_GMAX_A>=.06 then .06 elseif Set_GMAX_A<=-.06 then -.06 else Set_GMAX_A*sediment__limitation_A";
char SGR_A_description[] = "specific growth rate";
char SGR_A_comment[] = "Takes GMAX set by temperature and light and reduced by sediment sulfide limitation factor. Units are gdw/gdw/day. I put boundary limits on the adult GMAX since this formulation was causing at times abnormally large spikes and dips in the SGR due to an extreme GMAX.\n\
\n\
See Table 1.2 of Harris dissertation for why 0.06 was chosen as a boundary limit.";
char umax_A_name[] = "umax_A";
char umax_A_spec[] = "if Temperature_SA<=0 then -0.001 else .0177*log(Temperature_SA)+.0011";
char umax_A_description[] = "maxgrowth from Growth versus irradiance curves.";
char umax_A_comment[] = "predicted from Olesen and sand jensen\n\
Based on equations found in PICURVES_3.xls\n\
units are grams dw growth/gramsdryweight/d";
char alpha_A_name[] = "alpha_A";

char alpha_A_spec[] = "if Temperature_SA>30 then .005 elseif Temperature_SA<5 then 0.005 else (-0.0003*Temperature_SA^2)+0.007*Temperature_SA-0.0296";
char alpha_A_description[] = "slope of Growth versus Irradiance curves";
char alpha_A_comment[] = "Based on data from Olesen and sand-jensen\n\
equations found in picurves_3.xls\n\
Units are g / gdw/d over mol/m2/d";
char ro_A_name[] = "ro_A";
char ro_A_spec[] = " -0.000727*Temperature_SA-.006825";
char ro_A_description[] = "repsiratory cost from Growth versus Irradiance curves.";
char ro_A_comment[] = "Taken from Olesen_sandjensen\n\
predicted using equations found in PICURVE_3.xls\n\
Units are grams dw growth/gdw/d";
char Set_GMAXNW_A_name[] = "Set GMAX_A";
char Set_GMAXNW_A_spec[] = "((umax*alpha*Izleaf/(umax+alpha*Izleaf))+ro)";
char Set_GMAXNW_A_description[] = "Setting t and Light determined Gmax";
char Set_GMAXNW_A_comment[] = "PS equation from Baly and other sources. Uses Relationship reported by Olesen and Sand-Jensen for Growth versus Irradiance. All units for calculating parameters and Gmax converted from ash-free dry weight to dry weight.\n\
units are g/gdw/d";
char SGR_Adult_name[] = "SGR_Adult";
char SGR_S_name[] = "SGR_S";
char SGR_S_spec[] = "umax_S*F_I_T__S*sediment_limitation_S";
char SGR_S_description[] = "Simulated Specific Growth Rate";
char SGR_S_comment[] = "Units are gdw/gdw/day\n\
\n\
if channel_is(Germinated_Seeds) and year<= 1 then 0 -- the seeds which enter the model as x, y coordinates at the moment of flowering remain dormant in the model as they do not technically germinate until year 2 due to hot summer temperatures(dormany) and ease of model programming this phenomena.\n\
\n\
previous code for when I did not have the degree day compartments: if channel_is(Germinated_Seeds) and year<= 1 then 0 else umax_S*F_I_T__S*sediment_limitation_S";
char umax_S_name[] = "umax_S";
char umax_S_spec[] = ".03";
char umax_S_description[] = "maximum specific growth rate";
char umax_S_comment[] = "units: gdw/gdw/day\n\
Taken as approximate average from values reported in literatare and explained on AERS poster. .03";
char sediment_limitation_S_name[] = "sediment\n\
limitation_S";
char sediment_limitation_S_spec[] = "if sediments_SA<=1 then 1 elseif sediments_SA>=2000 then 0 else 1.0239*exp(-0.002*sediments_SA)";
char sediment_limitation_S_comment[] = "Taken from Dooley et al 2012. Scaled from 0 to 1 as limitation factor.";
char F_I_T__S_name[] = "F(I,T)_S";
char F_I_T__S_spec[] = "if Temperature_SA<= 25 then ((((0.97*Temperature_SA-0.75)*tanh((Iz_SA/(((4.5978*Temperature_SA)+3.3473))))+((-0.0949*Temperature_SA)-1.0503))/20.02) else ((((-2.8*Temperature_SA)+94.3)*tanh((Iz_SA/(((4.5978*Temperature_SA)+3.3473))))+((-0.0949*Temperature_SA)-1.0503))/20.02))";
char F_I_T__S_comment[] = "Abe et al. 2008 formulation, uses a Jassby & Platt parameterization";
char SGR_Seedling_name[] = "SGR_Seedling";
char Grass_Machine_name[] = "Grass Machine";
char kcanopy_A_name[] = "kcanopy_A";
char kcanopy_A_spec[] = "if time()<=1 then 0 else 2.09+0.00018*Square_Meter__Density*0.01*Canopy_Height__for_Light";

```
char kcanopy_A_description[] = "Light attenuation through leaf canopy";
char kcanopy_A_comment[] = "From Short 1980 taking effect of local shoot density and leaf length
into account.  Units are m-1.";
char Percent_Irradiance_A_name[] = "Percent\n\
Irradiance_A";
char Percent_Irradiance_A_spec[] = "if time()<=1 or Square_Meter__Density<=220 then
Iz_SA/Surface_PAR_S else Izleaf_A/Surface_PAR_S+.05";
char Percent_Irradiance_A_comment[] = "Percent of surface irradiance reaching leaf of eelgrass.";
char Izleaf_A_name[] = "Izleaf_A";
char Izleaf_A_spec[] = "if time()<=1 or time()==init_time() then Iz_SA else Iz_SA*exp(-1*kcanopy*
0.005 *Canopy_Height__for_Light)";
char Izleaf_A_comment[] = " Light reaching eelgrass leaf after attenuation of downwelling irradiance
by shading of leaves.  .005 conversion from cm to m and using only half the length of the leaf.  Units
are uMol/m2/day.";
char Light_Adult_name[] = "Light_Adult";
char Surface_PAR_S_name[] = "Surface PAR_S";
char Surface_PAR_S_spec[] = "table(int(time()))";
char Surface_PAR_S_description[] = "Irradiance values";
char Surface_PAR_S_comment[] = "Taken from Taskinas Creek Dataset from M.Brush. (Not for all)";
char Iz_SA_name[] = "Iz_SA";
char Iz_SA_spec[] = "Surface_PAR_S*exp((-k_SA*water_depth_SA))";
char Iz_SA_comment[] = "Light at depth after attenuation through water column\n\
\n\
Surface_PAR_S*exp((-k_SA*water_depth_SA))";
char Percent_Irradiance_S_name[] = "Percent\n\
Irradiance_S";
char Percent_Irradiance_S_spec[] = "Iz_SA/(Surface_PAR_S)";
char Percent_Irradiance_S_comment[] = "Percent of surface irradiance reaching leaf of eelgrass. No
leafIZ like adult model.";
char Light_Seedling_name[] = "Light_Seedling";
char Light_name[] = "Light";
char Specific_Growth_Rate_name[] = "Specific Growth Rate";
char my_internode_length_name[] = "my internode\n\
length";
char my_internode_length_spec[] = "if index(1)==1 then 0 else  317.28*Growing_R___R";
char my_internode_length_comment[] = "This sets the size of the internode associated with this node.
The first node  (i.e. oldest) of the ramet is given a value of 0, so that an internode length between nodes
0 and 1 is associated with node 1, between nodes 1 and 2 is associated with node 2, and so on.  Units
are mm.";
char index_name[] = "index";
char index_spec[] = "index(1)";
char index_comment[] = "instance of a particular node in the node population submodel";
char node_age_name[] = "node age";
char node_age_spec[] = "time(1)-init_time(1)";
char node_birthday_name[] = "node\n\
birthday";
char node_birthday_spec[] = "if channel_is(Initial___Nodes) then 0 else init_time(1)";
char node_birthday_comment[] = "Time step when a node was created.";
char NodeXNW_name[] = "NodeX";
char NodeXNW_spec[] = "if index(1)==2 then 17 elseif index(1)==3 then 34 elseif index(1)==4 then
51 else X";
char NodeXNW_comment[] = "Takes location of X when node is born to initialize state variable of a
node instance in this submodel.  These coordinates then mark the location of the node in 2-D space.
Units are meters.";
char NodeY_name[] = "NodeY";
```

char NodeY_spec[] = "if index(1)<5 then 0 else Y";
char NodeY_comment[] = "Takes location of X when node is born to initialize state variable of a node instance in this submodel. These coordinates then mark the location of the node in 2-D space. Units are meters.";
char Node_births_and_lengths_name[] = "Node births and lengths";
char Node_births_and_lengths_comment[] = "This population submodel was created so that I could calculate a length to be associated with each node. It accomplishes this, but processing the values of the population list becomes tricky, so there are many variables outside of the model used to transfer information back to the other variables in \
the shoot model.";
char is_adult_name[] = "is_adult";
char is_adult_spec[] = "if channel_is(Adult__Initialization) or channel_is(BRANCH) or my_number__of_nodes>=4 then \"true\" else \"false\"";
char is_adult_comment[] = "original: my_number__of_nodes>=4, however, this was allowing for new branches off of adult eves to be considered seedlings, which is incorrect.\n\
\n\
You are an adult if you have >=4 nodes, are a branch/have branched, or were initialized as an adult. One except is seedlings with 3 nodes that sexually reproduce, while it is possible they are not technically considered adults in the model yet (just shy of adulthood).";
char Adulthood_name[] = "Adulthood";
char flower_name[] = "flower";
char flower_spec[] = "if Temperature_SA>=15 and (time(0)==103 or time(0)==468 or time(0)==833 or time(0)==1198) and my_number__of_nodes>=3 and randomflower<=(if Daysabove20<=100 then 0.3 elseif Daysabove20>=150 then 0.9 else 0.5) then \"true\" else \"false\"";
char flower_comment[] = "JF current code: This code causes an individual to flower once the temperature degree day compartment reaches 3 days above an average of 15 degrees C and the individual has more than 3 nodes (adulthood). The Daysabove20 is a parameter that is linked to the temperature and dictates the set probability \
of flowering. The flowering is not a percentage of the total shoots but of the adult shoots that are capable of flowering. At the moment, you need to manually input the index aka \"day\" of year the 3rd 15 C degree day is met.\n\
\n\
\n\
L.Harris' previos code: Hybrid mechanistic-stochastic parameter to decide if an individual should flower. Linked to temperature, time of year, and overall probability. Boolean.\n\
\n\
Flowering percentages were chosen after performing a review of eelgrass reproductive shoot percentages in the literature.\n\
\n\
GB: ~50 days above 20 per year\n\
VA South Bay: ~126 days above 20 per year\n\
NC ~180 days above 20 per year\n\
\n\
How do I return just the first \"true\" value? Cannot unless it is an array.\n\
\n\
previous code before the degree days compartment: if Temperature_SA>=16 and delay(Temperature_SA,1)<15 and my_number__of_nodes>3 and randomflower<=(if Daysabove20<=100 then 0.3 elseif Daysabove20>=150 then 0.9 else 0.5) then \"true\" else \"false\"";
char Flower_name[] = "Flower";
char Branching_Rate_Fix_name[] = "Branching Rate\n\
Fix";
char Branching_Rate_Fix_spec[] = "in_progenitor(Eve_Status)";
char Branching_Rate_Fix_comment[] = "Currently you have two associations in the model which are set up as\n\
parent/child associations, i.e., the Branching Rate and Inheritance\n\

```
submodels. I can't see why you need an association for branching rate at\n\
all, since all it is doing is counting the number of ramets that are\n\
branches of Eves. To get this count, you could just add a variable in\n\
the ramet submodel with the equation 'in_progenitor(Eve_Status)' which\n\
would have the value 1 if its parent was Eve and 0 otherwise. You would\n\
then sum this outside the submodel to get the total branches of Eves, to\n\
divide by the initial number to get the average Eve branching. You could\n\
then delete the Branching Rate submodel.";
char Negative_Root_biomass_0_name[] = "Negative\n\
Root biomass";
char Negative_Root_biomass_0_spec[] = "if ROOTS___RHIZOMES<-0.1 then 1 else 0";
char Negative_Root_biomass_0_comment[] = "This code signifies death of an individual that has lost
all of its root biomass (i.e. below 0). Previously, root biomass was just going very negative but the
individual was not dying.";
char countflowers_name[] = "countflowers";
char countflowers_spec[] = "if flower==\"true\" then 1 else 0";
char adult_status_name[] = "adult status";
char adult_status_spec[] = "if is_adult==\"true\" then 1 else 0";
char Me_0_name[] = "Me_0";
char Me_0_spec[] = "index(1)";
char seeds_perflowerShoot_name[] = "seeds\n\
perflowerShoot";
char seeds_perflowerShoot_spec[] = "if flower==\"true\" then 10 else 0";
char seeds_perflowerShoot_comment[] = "After april, the flowering shoots will have set out 10 seeds
per reproductive shoot. J.Jarvis 2014\n\
\n\
Silberhorn 1983 (~23 seeds per flowering shoot). Another parameter to consider.";
char Initial_Density_name[] = "Initial\n\
Density";
char Initial_Density_spec[] = "Set_Initial_Density";
char Initial_Density_comment[] = "Density of shoots specified for start of simulation.  Units should be
shoots/m2";
char LeavesProduced_name[] = "LeavesProduced";
char LeavesProduced_spec[] = "0";
char Pl_name[] = "Pl";
char Pl_spec[] = "if delay(OLDESTLEAF,1)>OLDESTLEAF then 1 else 0";
char init_y_name[] = "init_y";
char init_y_spec[] = "if time()<=1 then rand_const(0,1.05) else last(Y)";
char init_y_comment[] = "a table?";
char init_XNW_name[] = "init_X";
char init_XNW_spec[] = "if time()<=1 then rand_const(0,1.6) else last(X)";
char init_XNW_comment[] = "previous if time()<=1 then rand_const(0,1.6) elseif
channel_is(Germinated_Seeds) then rand_const(0,1.6) else last(X)";
char check_b_adult___fix_name[] = "check b\n\
adult ? fix";
char check_b_adult___fix_spec[] = "if Temperature_SA< 8.064 then 0.75 else Gr/(Gr+Gl)";
char RHIZOME_DETRITUS_name[] = "RHIZOME\n\
DETRITUS";
char RHIZOME_DETRITUS_spec[] = "0";
char RHIZOME_DETRITUS_comment[] = "Biomass of decayed and decaying internode material.
Units are grams.";
char LEAFDETRITUS_name[] = "LEAFDETRITUS";
char LEAFDETRITUS_spec[] = "0";
char LEAFDETRITUS_comment[] = "State variable keeping track of leaf detritus.  Units are grams.";
char Gr_name[] = "Gr";
```

```
char Gr_spec[] = "if flower==\"true\" then 0 else b*Gl/(1-b)";
char Gl_name[] = "Gl";
char Gl_spec[] = "Leaves*(if is_adult then GI_A else GI_S)";
char Decomp_name[] = "Decomp";
char Decomp_spec[] = "if time()<2 or my_number__of_nodes<1 then 0 elseif
fmod(my_number__of_nodes,12)==0 then (average_internode_length*12)/317.28 else 0";
char Rhizome_Length_name[] = "Rhizome Length";
char Rhizome_Length_spec[] = "sum({my_internode_length})";
char Rhizome_Length_comment[] = "Sums the entire node submodel for a ramet to calculate its total
rhizome length.  This will include ALL rhizome ever produced, even if some of the older material
eventually dies off.  Units are mm.";
char My_birthday_name[] = "My birthday";
char My_birthday_spec[] = "init_time()";
char My_birthday_comment[] = "Self-explanatory.  Units are day of simulation.";
char Y_name[] = "Y";
char Y_spec[] = "if time()<=1 then init_y else element([My_Birthplace],2)";
char Y_comment[] = "Y coordinate for current location in 2-D space.  Units are meters. nitial Eve
population starting location determined by planted grid location.";
char XNW_name[] = "X";
char XNW_spec[] = "if time()<=1 then init_X else element([My_Birthplace],1)";
char XNW_comment[] = "X coordinate for current position in 2-D space.  Units are meters.  Initial
Eve population starting location determined by planted grid location.";
char Move_YAPEXNW_name[] = "Move YAPEX";
char Move_YAPEXNW_spec[] = "if channel_is(Adult__Initialization) then
Internode_Length*sin((element([My_Birthplace],3))) elseif Di__branch==0.25 then
Internode_Length*sin((element([My_Birthplace],3)+1.326)) else
Internode_Length*sin((element([My_Birthplace],3)-1.326))";
char Move_XNWAPEXNW_name[] = "Move XAPEX";
char Move_XNWAPEXNW_spec[] = "if channel_is(Adult__Initialization) then
Internode_Length*cos((element([My_Birthplace],3))) elseif Di__branch==0.25 then
Internode_Length*cos((element([My_Birthplace],3)+1.326)) else
Internode_Length*cos((element([My_Birthplace],3)-1.326))";
char My_Birthplace_name[] = "My Birthplace";
char My_Birthplace_spec[] = "if Eve_Status==1 then [MyCoords] else in_progenitor([MyCoords])";
char My_Birthplace_comment[] = "Coordinates and direction of mother ramet at time of branching
passed from Inheritence submodel.\n\
\n\
\n\
.....Jasper The Inheritance submodel is used to pass 'My Coords' of the parent to\n\
'My Birthplace' of the offspring. This could be done more simply by\n\
adding a direct influence, and setting the equation for 'My Birthplace'\n\
to 'if Eve_Status==1 then [My_Coords] else in_progenitor([My_Coords])'.\n\
The condition is needed because the condition in the association\n\
includes a clause that makes it exist between Eve individuals and\n\
themselves, so their birthplaces are their own coords.\n\
\n\
The same submodel also passes values from NODES in the parent to\n\
Di-branch in the offspring. This connection could also be made directly\n\
with the in_progenitor() function, after which the Inheritance submodel\n\
could be deleted.";
char MyCoords_name[] = "MyCoords";
char MyCoords_spec[] = "if time()<=1 then[init_X,init_y,rand_var(0,6.36)] elseif
channel_is(Germinated_Seeds) then[init_X,init_y,rand_var(0,6.36)] else last([X,Y,DIRECTION])";
char MyCoords_comment[] = "Array used to pass an individual's coordinates and direction to herself
or her daughter ramet.\n\
```

```
\n\
if time()<=1 then[rand_var(0,1.5),rand_var(0,.95),rand_var(0,6.36)]else last([X,Y,DIRECTION])\n\
\n\
previous: if time()<=1 then[init_X,init_y,rand_var(0,6.36)] else last([X,Y,DIRECTION]) when init_x and y were an input table number";
char age_of__youngest_node_name[] = "age of \n\
youngest node";
char age_of__youngest_node_spec[] = "least({node_age})";
char age_of__youngest_node_comment[] = "Units are days.";
char GROWING_INTERNODE_name[] = "GROWING\n\
INTERNODE \n\
";
char GROWING_INTERNODE_spec[] = "(((initial_ch*0.0012)/1.2)*4.7)/4";
char GROWING_INTERNODE_comment[] = "A growing node-internode not yet \"born\".  In reality, this reserve might be more closely associated with the meristem.  Units are grams";
char NODES_name[] = "NODES";
char NODES_spec[] = "if is_adult then 4 else 0";
char NODES_comment[] = "This keeps track of the number of nodes up to a value of 4 and is only included here because this number represents the magic time when a shoot should create new branches!  This stock and flow equation keeps track of new sets of 4* and less nodes to accomplish the node driven branching rate.  Units \
are number of nodes.";
char In_name[] = "In";
char In_spec[] = "if delay(OLDESTLEAF,1)>OLDESTLEAF then 1 else 0";
char Io_name[] = "Io";
char Io_spec[] = "if NODES==4 then 4 else 0";
char b_A_name[] = "b_A";
char b_A_spec[] = "if Temperature_SA< 8.064 then 0.65 else (-.0346*Temperature_SA)+1.029";
char b_A_description[] = "% Belowground Allocation";
char b_A_comment[] = "Function of temperature.  Based on data from 1999 Mesocosm.  Excel file assocated with parameterization named \"k1_temperatureallocation\".  Because data does not include situations with VERY cold temperatures or very high belowground partitioning, the if-then statement caps the max percentage at 0.75 \
at a temperature of 7.385 degrees (the temperature associated with this value using the temperature-partitioning regression equation).  units are a percentage of growth (gdw/gdw)";
char Canopy_Height__for_Light_name[] = "Canopy Height \n\
for Light";
char Canopy_Height__for_Light_spec[] = "if time()<=1 or time()==init_time() then 30 else 28.55*(((if is_adult then Percent_Irradiance_A else Percent_Irradiance_S))^-.60)";
char Canopy_Height__for_Light_description[] = "User defined maximum canopy height for depth";
char Canopy_Height__for_Light_comment[] = "At the moment this is a user defined varible.  It would be nice to have a canopy height versus depth relationship! Units are centimeters.";
char LEAFBUNDLE_name[] = "LEAFBUNDLE";
char LEAFBUNDLE_spec[] = "if is_adult then ((initial_ch*0.0012)/1.2)*3.7 else .0015";
char LEAFBUNDLE_comment[] = "Leaves, excluding the eldest.  Units are grams.\n\
\n\
Seedling data taken from Brush/Orth data (estimate). .001 is representative of a seedling with one leaf when it is initialized (Jan 1) in the model or once it germinates. The maol for a seedling is .0012";
char Ol_name[] = "Ol";
char Ol_spec[] = "if LEAFBUNDLE>malb then LEAFBUNDLE-malb else 0";
char OLD__NODES_name[] = "OLD \n\
NODES";
char OLD__NODES_spec[] = "0";
```

```
char OLD__NODES_comment[] = "We might like the mortality of a shoot ot be related to how many
nodes it has so we store the toal number here.  This also provides a check on the population size value
taken from the node population submodel.  Units are # of nodes.";
char Nutrient__Canopy_Factor_name[] = "Nutrient \n\
Canopy Factor";
char Nutrient__Canopy_Factor_spec[] = "(.206*Log(Nitrogen_SA))+1.7679";
char Nutrient__Canopy_Factor_comment[] = "An observation in many enrichment experiments has
been that leaves elongate in response to higher nitrogen concentrations.  This multiplicative factor
takes the predicted weight of the longest leaf (which limits for the leaf compartment are based upon)
and increases or decreases these limits under high \
or low nitrogen conditions, respectively.  The regression equation was taken from Roberts et al.";
char malb_name[] = "malb";
char malb_spec[] = "if is_adult then Oldest_Leaf_Mass* 3.7 else Oldest_Leaf_Mass";
char malb_comment[] = "This value is an estimate of how much four leaves should weigh based on the
canopy height of the shoot.  It is also assumed that the third leaf is the longest leaf and determines the
canopy height.  Distribution of weight amongst the leaves was determined from empirical data taken
from mesocosms and \
field sites in southern Rhode Island.  Kept in file \"Leaf sizes\".  Units are gdw.";
char Gi_name[] = "Gi";
char Gi_spec[] = "if delay(OLDESTLEAF,1)>OLDESTLEAF then GROWING_INTERNODE else
0";
char ROOTS___RHIZOMES_name[] = "ROOTS &\n\
RHIZOMES";
char ROOTS___RHIZOMES_spec[] = "((((initial_ch*0.0012)/1.2)*4.7)/4)*2.5";
char ROOTS___RHIZOMES_comment[] = "Established roots and rhizomes.  This would be what you
would actually measure as rhizome growth if you were taking field measurements.  Units are g.\n\
\n\
original: ((((initial_ch*0.0012)/1.2)*4.7)/4)*3";
char OLDESTLEAF_name[] = "OLDESTLEAF";
char OLDESTLEAF_spec[] = "if is_adult then (initial_ch*0.0012)/1.2 else 0.015";
char OLDESTLEAF_comment[] = "The \"oldest leaf\", or material that exceeds the maximum biomass
threshold for the \"leaves\" compartment.  Units are grams.\n\
\n\
***If is_adult then_____ else ___\n\
\n\
0.001 is taken from calibration data from M. Brush young seedlings";
char BRANCH_0_name[] = "BRANCH";
char BRANCH_0_spec[] = "if NODES==4 and delay(NODES,1)==3 then 1 elseif NODES==3 and
delay(NODES,1)==2 then 1 else 0";
char BRANCH_0_description[] = "Asexual Reproduction/Branching";
char BRANCH_0_comment[] = " This population process relates the number of nodes produced since
the last lateral shoot to help time the creation of a new lateral shoot.  Data used to choose a value of
4/3(?) nodes between shoots was taken from ponds/mesocosm data. Units are number of shoots per
day.\n\
\n\
original *if NODES==6 and delay(NODES,1)==5 then 1 elseif NODES==5 and delay(NODES,1)==4
then 1 else 0*\n\
\n\
Once you produce a BRANCH you are an adult.\n\
\n\
if NODES==4 and delay(NODES,1)==3 then 1 elseif NODES==3 and delay(NODES,1)==2 then 1
else 0";
char Death_0_name[] = "Death";
char Death_0_spec[] = "delay_death";
char Death_0_description[] = "Mortality";
```

```
char Death_0_comment[] = "Mortality due to flowering.  Units are ramets/day\n\
\n\
if FLOWER__CLOCK==41 then 1 else 0 - original\n\
\n\
Do I meed tp add a mortality of biomass is 0? Is there any biomass below 0?\n\
\n\
Individuals are removed at the start of the time step following the one in which their number came up therefore I wanted to delay the time from flower to death.";
char Sl_name[] = " Sl";
char Sl_spec[] = "if OLDESTLEAF>maol then OLDESTLEAF-maol else 0";
char maol_name[] = "maol";
char maol_spec[] = "if is_adult then Nutrient__Canopy_Factor*(Canopy_Height__for_Light* 0.0012 / 1.2) else 0.015*1.2";
char maol_description[] = "Oldest Leaf Mass determined by Nutrient conditions";
char maol_comment[] = "Uses the canopy height as representative of the third leaf.  Empirically, the third leaf represents 1.2xoldest leaf length, so the oldest leaf weight is calculated based on this trick as well as a limitation factor based on nutrient conditions.  The .0012 factor is the conversion factor to get from \
length (cm) to biomass (g).  Taken from Nixon Lab data.  File = Leaf SIzes\n\
\n\
***If is_adult then ____ else 0.0012 \n\
Units of biomass, seedling set at .0012 g DW (biomass) do not confuse with the conversion factor (coincidence that they are the same value). .0012 taken from Brush seedling data. Add a bit more perhaps .002 so that seedlings are not creating new nodes in one week or less from model calibration.";
char my_number__of_nodes_name[] = "my number \n\
of nodes";
char my_number__of_nodes_spec[] = "count({index})";
char my_number__of_nodes_comment[] = "Counts number of instances in node submodel to determine how many nodes exist for an individual ramet.";
char one_name[] = "one";
char one_spec[] = "1";
char one_comment[] = "Parameter used to count number of individuals outside RAMET submodel";
char average_internode_length_name[] = "average\n\
internode\n\
length";
char average_internode_length_spec[] = "if my_number__of_nodes>0 then sum({my_internode_length})/my_number__of_nodes else 0";
char average_internode_length_comment[] = "Self explanatory.  Units are mm.";
char youngest__birthday_name[] = "youngest \n\
birthday";
char youngest__birthday_spec[] = "greatest({node_birthday})";
char youngest__birthday_comment[] = "Units are day of simulation";
char Canopy_Height_name[] = "Canopy Height";
char Canopy_Height_spec[] = "Nutrient__Canopy_Factor*Canopy_Height__for_Light";
char Eve_name[] = "Eve";
char Eve_spec[] = "not channel_is(BRANCH)";
char Eve_comment[] = "Used to determine if an individual is part of the \"eve\" population. \n\
\n\
-I do not think this works. It just tells you the same what index(1) or Me tells you since even newly added individuals during a run were returning a \"true\" value.\n\
\n\
if channel_is(BRANCH) or channel_is(germinated seeds) then 0 else 1";
char Eve_Status_name[] = "Eve Status";
char Eve_Status_spec[] = "if Eve==\"true\" then 1 else 0";
char Eve_Status_comment[] = "Boolean function to signal Eve Status";
```

```
char My_Mother_name[] = "My Mother";
char My_Mother_spec[] = "parent(1)";
char My_Mother_comment[] = "Identity of mother within population";
char Me_name[] = "Me";
char Me_spec[] = "index(1)";
char Me_comment[] = "My identity";
char Internode_Length_name[] = "Internode\n\
Length";
char Internode_Length_spec[] = "if time(1)==1 then (ROOTS___RHIZOMES*317.28)/1000 elseif
delay(OLDESTLEAF,1)>OLDESTLEAF then 317.28*GROWING_INTERNODE/1000 else 0";
char Internode_Length_comment[] = "This translates the biomass associated with the \"birth\" of a
node into an internode length that can be used to drive the spatial coordinates of the simulations.  Units
are m.";
char Plastochrone_Interval_name[] = "Plastochrone\n\
Interval";
char Plastochrone_Interval_spec[] = "if delay(youngest__birthday, 1)==youngest__birthday then 0 else
youngest__birthday-delay(youngest__birthday,1)";
char Plastochrone_Interval_comment[] = "Calculates node plastochrone interval (which is, by default,
also the leaf plastochrone interval).  Units are days.";
char Di__branch_name[] = "Di\n\
-branch";
char Di__branch_spec[] = "if Eve_Status==1 then 5 elseif in_progenitor(NODES)==5 then 0.25 else
0.75";
char Timed_Square_Meter_Density_name[] = "Timed Square\n\
Meter Density";
char Timed_Square_Meter_Density_spec[] = "if time()<=1 then Initial_Density elseif
fmod(time(),10)==0 then (count({index_My_Neighbour})*25) else 0";
char Timed_Square_Meter_Density_comment[] = "To increase simulation speed, the conditional
neighbour model only checks for neighbours every 10th time step.  This variable stores a value from
the submodel on each of these 10th timesteps.";
char DIRECTION_name[] = "DIRECTION";
char DIRECTION_spec[] = "if Di__branch== 0.25 then element([My_Birthplace],3)+ 1.326 elseif
Di__branch== 0.75 then element([My_Birthplace],3)- 1.326 else element([My_Birthplace],3)";
char DIRECTION_comment[] = "This stores the direction that a ramet is heading in so that it may be
passed to daughter ramets.  Units are radians.";
char Squared_Distance_to_rcm_name[] = "Squared\n\
Distance\n\
to rcm";
char Squared_Distance_to_rcm_spec[] = "(X-(element([rcmXY],1)))^2 + (Y-
(element([rcmXY],2)))^2";
char Squared_Distance_to_rcm_comment[] = "Needed to calculate Radius of Gyration parameter.
From Sintes et al. 2005";
char Square_Meter__Density_name[] = "Square Meter \n\
Density";
char Square_Meter__Density_spec[] = "if time()<=1 then 75 elseif fmod(time(),10)==0 then
Timed_Square_Meter_Density else delay(Timed_Square_Meter_Density,fmod(time(),10))";
char Square_Meter__Density_comment[] = "To increase simulation time, a local shoot density is only
calculated every 10th time step.  This variable take that value from the \"Timed Square Density\"
variable and associates it with the timesteps in between so that a constant shoot density is maintained
for the intervening simulation steps.  \
It will look something like this:\n\
\n\
Time    Shoot Density\n\
10      300\n\
11      300\n\
```

```
12        300\n\
....19    300\n\
20        update to new density - 320\n\
21        320\n\
and so on.\n\
\n\
Units are ramets/m2\n\
\n\
if time()<=1 then 75 elseif fmod(time(),10)==0 then Timed_Square_Meter_Density else
delay(Timed_Square_Meter_Density,fmod(time(),10)) --former code, playing around to see if this is
the reason the the adult SGR changes among individuals.\n\
\n\
if time()<=1 then 75 elseif fmod(time(),10)==0 then Timed_Square_Meter_Density else
delay(Timed_Square_Meter_Density,fmod(time(),10))";
char initial_ch_name[] = "initial ch";
char initial_ch_spec[] = "if is_adult then 12 else 5.5";
char initial_ch_comment[] = "cm -JF\n\
Adults is rand_var(12.17,25.92)\n\
Seedlings is randvar(2,9) \n\
This variable is initial length of seedlings in cm";
char randomflower_name[] = "randomflower";
char randomflower_spec[] = "rand_const(0,1)";
char randomflower_comment[] = "Random constant to determine probablistically if a plant should
flower or not.";
char b_S_name[] = "b_S";
char b_S_spec[] = "if Iz_SA<=90 then 0.30 elseif Iz_SA >=499 then 0.51 else 0.43";
char b_S_description[] = "% below ground allocation";
char b_S_comment[] = "Iz below ground biomass seedling relationship taken from Joanne Bintz's
dissertation work. Units are a percentage of growth (gdw/gdw)\n\
\n\
Less below ground allocation when light is limited.";
char b_name[] = "b";
char b_spec[] = "if is_adult ==\"true\" then b_A else b_S";
char GI_A_name[] = "GI_A";
char GI_A_spec[] = "if flower==\"true\" then 0 else SGR_A";
char GI_S_name[] = "GI_S";
char GI_S_spec[] = "if flower==\"true\" then 0 else SGR_S";
char delay_death_name[] = "delay death";
char delay_death_spec[] = "delay(flower_time,60)";
char delay_death_comment[] = "However, recent observations\n\
of an annual form of Zostera marina L. indicate that flowering\n\
plants in some populations are produced during the first year of\n\
growth. All flowering shoots, regardless of their age, will die by\n\
the end of the growing season, and usually by the end of the flowering period. Therefore, I chose 60
days.\n\
\n\
Delay death until 21 degrees around when fruit maturation is complete.";
char flower_time_name[] = "flower time";
char flower_time_spec[] = "if flower==\"true\" then 1 else 0";
char Initial___Nodes_name[] = "Node births and lengths/Initial\n\
# Nodes";
char Initial___Nodes_spec[] = "if channel_is(Adult__Initialization) then 4 else 0";
char Initial___Nodes_description[] = "Initialization";
char Initial___Nodes_comment[] = "How many nodes to start the model off with.";
char Add_a__node__0_name[] = "Node births and lengths/Add a \n\
```

node!";
char Add_a__node__0_spec[] = "if delay(OLDESTLEAF,1)>OLDESTLEAF then 1 else 0";
char Add_a__node__0_description[] = "Immigration";
char Add_a__node__0_comment[] = "A new node is \"born\" each time that an old leaf is sloughed off. In actuality, this is an immigration process in population terms, rather than a reproductive term. To accomplish this, the \"birth\" immigration process checks to see if the tank for \"Oldest Leaf Mass\" was larger on the last time \
step than on the current time step. If it's lower, then that tank has been emptied and a new node is born! Units are nodes/day.\n\
\n\
Couldn't the mass of the leaf decrease during hot temperatures and thus signify a node birth incorrectly?*";
char Ramet_name[] = "Ramet";
char Ramet_comment[] = "The RAMET submodel. Contains all rules and mechanisms for growth and colonization of an eelgrass ramet.";
char cond1_name[] = "cond1";
char cond1_spec[] = "fmod(time(), 10)==0 and (X_My_Neighbour<=(X_ME+0.1)) and (X_My_Neighbour>(X_ME-0.1)) and (Y_My_Neighbour<=(Y_ME+0.1)) and (Y_My_Neighbour>(Y_ME-0.1))";
char cond1_comment[] = "Identify neighbours within 1 meter radius of individual";
char index_0_name[] = "index";
char index_0_spec[] = "index(1)";
char Neighbours_name[] = "Neighbours";
char Neighbours_comment[] = "Relational/Conditional submodel used to identify a ramet's neighbours for the purposes of calculating a local shoot density.";
char countME_name[] = "countME";
char countME_spec[] = "count({Me_0})";
char sum_adults_name[] = "sum_adults";
char sum_adults_spec[] = "sum({adult_status})";
char sum_Roots_name[] = "sum_Roots";
char sum_Roots_spec[] = "sum({ROOTS___RHIZOMES})";
char NH_GB_TEMP_name[] = "NH_GB_TEMP";
char NH_GB_TEMP_spec[] = "table(int(time()))";
char NH_GB_PAR_name[] = "NH_GB_PAR";
char NH_GB_PAR_spec[] = "table(int(time()))";
char VA_HIB_TEMP_name[] = "VA_HIB_TEMP";
char VA_HIB_TEMP_spec[] = "table(int(time()))";
char abovebelow_ratio_name[] = "abovebelow ratio";
char abovebelow_ratio_spec[] = "sum_LEAFbundle/sum_Roots";
char NC_NERRS_TEMP_name[] = "NC_NERRS_TEMP";
char NC_NERRS_TEMP_spec[] = "table(int(time()))";
char South_Bay_2013_TEMP_name[] = "South Bay_2013_TEMP";
char South_Bay_2013_TEMP_spec[] = "table(int(time()))";
char VA_HIB_PAR_name[] = "VA_HIB_PAR";
char VA_HIB_PAR_spec[] = "table(int(time()))";
char NC_NERRS_PAR_name[] = "NC_NERRS_PAR";
char NC_NERRS_PAR_spec[] = "table(int(time()))";
char Average_Iz_name[] = "Average Iz";
char Average_Iz_spec[] = "(sum({Iz_SA}))/(count({Iz_SA}))";
char averageIleaf_name[] = "averageIleaf";
char averageIleaf_spec[] = "(sum({Izleaf_A}))/(count({Izleaf_A}))";
char avgkcanopy_name[] = "avgkcanopy";
char avgkcanopy_spec[] = "(sum({kcanopy_A}))/(count({kcanopy_A}))";
char Total_Branches_of_Eves_name[] = "Total Branches of\n\
Eves";

```
char Total_Branches_of_Eves_spec[] = "sum({Branching_Rate_Fix})";
char NC1_NC2Comb_TEMP_name[] = "NC1_NC2Comb_TEMP";
char NC1_NC2Comb_TEMP_spec[] = "table(int(time()))";
char sum_Eve_name[] = "sum Eve";
char sum_Eve_spec[] = "sum({Eve_Status})";
char average_Eve_Branching_name[] = "average Eve\n\
Branching";
char average_Eve_Branching_spec[] = "Total_Branches_of_Eves";
char average_Eve_Branching_comment[] = "Total_Branches_of_Eves/Adult__Initialization\n\
\n\
need to divide by initialization";
char shoot_density_name[] = "shoot\n\
density";
char shoot_density_spec[] = "if time()==0 then 0 else Number_of__Shoots/((Greatest_Y-
Least_Y)*(Greatest_X-Least_X))";
char shoot_density_comment[] = "Shoot density based on furthest extent of X-Y coordinates and
number of existing shoots.  Units are shoots/m2";
char Number_of__Shoots_name[] = "Number of \n\
Shoots";
char Number_of__Shoots_spec[] = "count({one})";
char Number_of__Shoots_comment[] = "Total number of shoots in RAMET population model.";
char sum_LEAFbundle_name[] = "sum_LEAFbundle";
char sum_LEAFbundle_spec[] = "sum({LEAFBUNDLE})";
char total_biomass_name[] = "total biomass";
char total_biomass_spec[] = "sum_Roots+sum_LEAFbundle";
char Total_Rhizome__Length_name[] = "Total Rhizome \n\
Length";
char Total_Rhizome__Length_spec[] = "(sum({Rhizome_Length}))/1000";
char Total_Rhizome__Length_comment[] = "Takes sum of entire population's internode lengths to
calculate the total rhizome length of the meadow.  Converts from mm to meters.";
char Average_Internode__Length_name[] = "Average Internode \n\
Length";
char Average_Internode__Length_spec[] =
"(sum({average_internode_length}))/(count({average_internode_length}))";
char Average_Internode__Length_comment[] = "This calculates the average internode length for the
entire population of ramets.  Units are mm.";
char Greatest_Y_name[] = "Greatest Y";
char Greatest_Y_spec[] = "last(greatest({Y}))";
char Greatest_XNW_name[] = "Greatest X";
char Greatest_XNW_spec[] = "last(greatest({X}))";
char Least_XNW_name[] = "Least X";
char Least_XNW_spec[] = "last(least({X}))";
char Least_Y_name[] = "Least Y";
char Least_Y_spec[] = "last(least({Y}))";
char averageLL_name[] = "averageLL";
char averageLL_spec[] = "sum({Canopy_Height__for_Light})/count({Canopy_Height__for_Light})";
char averageLL_comment[] = "Parameter used to calculate average canopy height of entire population.
Units are cm.";
char Average_Density_name[] = "Average\n\
Density";
char Average_Density_comment[] = "Takes average of local shoot densities calculated for each
individual to compare with total shoot density calculated from entire extent of X-Y coordinates.  Units
are ramets/m2.";
char rcmXNWY_name[] = "rcmXY";
char rcmXNWY_spec[] = "[(sum({X}))/(count({X})), (sum({Y}))/(count({Y}))]";
```

char rcmXNWY_comment[] = "Needed to calculate Radius of Gyration parameter. Methods from Sintes et al. (2005).";
char Radius_of_Gyration_name[] = "Radius\n\
of Gyration";
char Radius_of_Gyration_spec[] =
"sqrt((sum({Squared_Distance_to_rcm}))/(count({Squared_Distance_to_rcm})))";
char Radius_of_Gyration_comment[] = "Radius of Gyration. Calculated based on Sintes et al. 2005. Describes shape of patch formation";
char sumflowers_name[] = "sumflowers";
char sumflowers_spec[] = "sum({countflowers})";
char sumflowers_comment[] = "Does this need to be count flowers=true as opposed to just counting all the individuals?";
char sum_seeds_name[] = "sum seeds";
char sum_seeds_spec[] = "sum({seeds_perflowerShoot})";
char sum_seeds_comment[] = "Suming all seeds from all reproduction shoots. Therefore total # of seeds produced from adult flowering population per day\n\
\n\
***Need to fix this. I need the seeds to only be counted on the one day of flowering and then remain dormant until the temperature cools to 15C, which could initiate germination.";
char Seeds_that_will_Germinate__Seed_Bank__name[] = "Seeds that will\n\
Germinate [Seed Bank]";
char Seeds_that_will_Germinate__Seed_Bank__spec[] = "round(sum_seeds*0.4)";
char Seeds_that_will_Germinate__Seed_Bank__comment[] = "40% of seeds produced are viable J.Jarvis 2014\n\
\n\
Should I cut down this percentage further to account for low seedling establishment rates?";
char Set_Initial_Density_name[] = "Set Initial\n\
Density";
char Set_Initial_Density_spec[] = "62.5";
char delay__seeds_name[] = "delay \n\
seeds";
char delay__seeds_spec[] = "delay(Seeds_that_will_Germinate__Seed_Bank_, 171)";
char delay__seeds_comment[] = "delays migration by x time units so that they only enter the model as growing individuals once they germinate. Germination occurs once the temperature decreases from 20 degrees C (accumulates 3 Germination degree days under 20 degrees C).\n\
\n\
delay(Seeds_that_will_Germinate__Seed_Bank_, 30)\n\
\n\
The seeds remained in the seed-bank until water tem-peratures decreased below 20◦C as this is when germination isinitiated in Chesapeake Bay populations (Moore et al., 1993).\n\
\n\
In this case since they flower on day 12 and they would germinate on day 304 then the germinated seeds are delayed by 304-12 = 292 days for North Carolina\n\
\n\
VA South Bay: They flower on day 103 and they germinate on day 273 therefore seeds are delayed by 273-103 = 170 days\n\
\n\
if time(0)<=365 then delay(Seeds_that_will_Germinate__Seed_Bank_, 274) else delay(Seeds_that_will_Germinate__Seed_Bank_, 171)";
char BINTZ_MESO_PAR_name[] = "BINTZ_MESO_PAR";
char BINTZ_MESO_PAR_spec[] = "table(int(time()))";
char BINTZ_MESO_TEMP_name[] = "BINTZ_MESO_TEMP";
char BINTZ_MESO_TEMP_spec[] = "table(int(time()))";
char sum_BRANCH_name[] = "sum BRANCH";
char sum_BRANCH_spec[] = "sum({BRANCH})";

```
char how_many_adults_true_name[] = "how many adults\n\
true";
char how_many_adults_true_spec[] = "howmanytrue({is_adult})";
char Germinated_Seeds_0_name[] = "Ramet/Germinated\n\
Seeds";
char Germinated_Seeds_0_spec[] = "if time(0)==274 then 40 else round(delay__seeds)";
char Germinated_Seeds_0_comment[] = "The number of seeds that will germinate from the seed bank enter the model as inidividuals 1 time step after the reproductive shoots flower. For this reason, their SGR is 0 for the first year since technically the seeds have not yet germinated. Germination takes place once temperatures cool after the \
hot summer. Therefore the seeds are in a period of dormany until year 2.\n\
\n\
Waiting until January for the SGR to kick in is also representative of germination in late fall as there is often a delay between germination and when the first photosynthetic leaves emerge from the ground's surface.\n\
\n\
You don't need to count or sum this value you can just use the bos itself to do so.";
char Initial_germinated_Seedlings_name[] = "Ramet/Initial germinated\n\
Seedlings";
char Initial_germinated_Seedlings_spec[] = "0";
char Initial_germinated_Seedlings_comment[] = "Number of seeds that will germinate and establish into seedlings. This number is taken from the adult model flowering components and feed into the seedling model (Not entirely correct). This is the initial number of seedlings at the start of a model run.";
char Adult__Initialization_name[] = "Ramet/Adult \n\
Initialization";
char Adult__Initialization_spec[] = "0";
char Adult__Initialization_comment[] = "Initial number of ADULT ramets.";
char Virtual_Eelgrass_Meadow_v_2_VAn_name[] = "Virtual Eelgrass Meadow v.2_VAn";
int nodecount = 194;
node_data_line nodedata[] = {{"node00000", VALUELESS, 0, NULL, 0, NULL, SPLIT, {0},
 {1, 0}, 95, 0, 0, SUBMODEL, {Virtual_Eelgrass_Meadow_v_2_VAn_name, NULL, NULL,
NULL}},
 {"node01022", VALUELESS, 0, NULL, 0, NULL, SPLIT, {START_VM, MEMBERS, END_VM,
0},
 {1, 1, -1, 0}, 98, 0, 0, SUBMODEL, {Eve_Plastochrone_Intervals_name, NULL, NULL, NULL}},
 {"node01581", FLAG, 0, NULL, 0, NULL, DERIVED, {0},
 {1, 1, -1, 4, 0}, 102, 0, 1, CONDITION, {I_am_Eve_name, I_am_Eve_spec, NULL, NULL}},
 {"node01584", REAL, 0, NULL, 0, NULL, DERIVED, {0},
 {1, 1, -1, 5, 0}, 103,  -1.0e+100,  1.0e+100, VARIABLE, {Eve_PI_name, Eve_PI_spec, NULL,
NULL}},
 {"node01586", INTEGER, 0, NULL, 0, NULL, DERIVED, {0},
 {1, 1, -1, 6, 0}, 104, -268435455, 268435455, VARIABLE, {count_name, count_spec, NULL,
NULL}},
 {"node01591", REAL, 0, NULL, 0, NULL, DERIVED, {0},
 {1, 1, -1, 7, 0}, 105,  -1.0e+100,  1.0e+100, VARIABLE, {NewNodes_name, NewNodes_spec,
NULL, NULL}},
 {"node01595", REAL, 0, NULL, 0, NULL, DERIVED, {0},
 {1, 1, -1, 8, 0}, 106,  -1.0e+100,  1.0e+100, VARIABLE, {Leaves_Shed_name, Leaves_Shed_spec,
NULL, NULL}},
 {"node01599", REAL, 0, NULL, 0, NULL, DERIVED, {0},
 {1, 1, -1, 9, 0}, 107,  -1.0e+100,  1.0e+100, VARIABLE, {Nodessquared_name, Nodessquared_spec,
NULL, NULL}},
 {"node01602", REAL, 0, NULL, 0, NULL, DERIVED, {0},
```

{1, 1, -1, 10, 0}, 108, -1.0e+100, 1.0e+100, VARIABLE, {above_name, above_spec, NULL, NULL}},
{"node01606", REAL, 0, NULL, 0, NULL, DERIVED, {0},
{1, 1, -1, 11, 0}, 109, -1.0e+100, 1.0e+100, VARIABLE, {below_name, below_spec, NULL, NULL}},
{"node01023", VALUELESS, 0, NULL, 0, NULL, SPLIT, {0},
{1, 2, 0}, 110, 0, 0, SUBMODEL, {EveStats_name, NULL, NULL, NULL}},
{"node01614", REAL, 0, NULL, 0, NULL, DERIVED, {0},
{1, 2, 1, 0}, 113, -1.0e+100, 1.0e+100, VARIABLE, {Tagged_PI_name, NULL, NULL, NULL}},
{"node01616", REAL, 0, NULL, 0, NULL, DERIVED, {0},
{1, 2, 2, 0}, 114, -1.0e+100, 1.0e+100, VARIABLE, {GreatestPI_name, NULL, NULL, NULL}},
{"node01618", REAL, 0, NULL, 0, NULL, DERIVED, {0},
{1, 2, 3, 0}, 115, -1.0e+100, 1.0e+100, VARIABLE, {mean_number__of_new_nodes_name, mean_number__of_new_nodes_spec, NULL, NULL}},
{"node01620", REAL, 0, NULL, 0, NULL, DERIVED, {0},
{1, 2, 4, 0}, 116, -1.0e+100, 1.0e+100, VARIABLE, {Greatest_New_Nodes_name, Greatest_New_Nodes_spec, NULL, NULL}},
{"node01622", REAL, 0, NULL, 0, NULL, DERIVED, {0},
{1, 2, 5, 0}, 117, -1.0e+100, 1.0e+100, VARIABLE, {mean_new_leaves_name, mean_new_leaves_spec, NULL, NULL}},
{"node01624", REAL, 0, NULL, 0, NULL, DERIVED, {0},
{1, 2, 6, 0}, 118, -1.0e+100, 1.0e+100, VARIABLE, {stdevnodes_name, stdevnodes_spec, NULL, NULL}},
{"node01626", REAL, 0, NULL, 0, NULL, DERIVED, {0},
{1, 2, 7, 0}, 119, -1.0e+100, 1.0e+100, VARIABLE, {leastnodes_name, leastnodes_spec, NULL, NULL}},
{"node01633", REAL, 0, NULL, 0, NULL, DERIVED, {0},
{1, 2, 8, 0}, 120, -1.0e+100, 1.0e+100, VARIABLE, {aboveavg_name, aboveavg_spec, NULL, NULL}},
{"node01635", REAL, 0, NULL, 0, NULL, DERIVED, {0},
{1, 2, 9, 0}, 121, -1.0e+100, 1.0e+100, VARIABLE, {abovestdev_name, abovestdev_spec, NULL, NULL}},
{"node01637", REAL, 0, NULL, 0, NULL, DERIVED, {0},
{1, 2, 10, 0}, 122, -1.0e+100, 1.0e+100, VARIABLE, {avgbelow_name, avgbelow_spec, NULL, NULL}},
{"node01639", REAL, 0, NULL, 0, NULL, DERIVED, {0},
{1, 2, 11, 0}, 123, -1.0e+100, 1.0e+100, VARIABLE, {belowstdev_name, belowstdev_spec, NULL, NULL}},
{"node01039", VALUELESS, 0, NULL, 0, NULL, SPLIT, {MEMBERS, 0},
{1, 3, -1, 0}, 124, 0, 0, SUBMODEL, {Ramet_name, NULL, NULL, Ramet_comment}},
{"node01695", VALUELESS, 0, NULL, 0, NULL, SPLIT, {0},
{1, 3, -1, 4, 0}, 127, 0, 0, SUBMODEL, {Specific_Growth_Rate_name, NULL, NULL, NULL}},
{"node01918", VALUELESS, 0, NULL, 0, NULL, SPLIT, {0},
{1, 3, -1, 4, 1, 0}, 130, 0, 0, SUBMODEL, {Physical_Setting_name, NULL, NULL, NULL}},
{"node01935", REAL, 0, NULL, 0, NULL, DERIVED, {0},
{1, 3, -1, 4, 1, 1, 0}, 133, 0.1, 5.0, VARIABLE, {water_depth_SA_name, water_depth_SA_spec, NULL, water_depth_SA_comment}},
{"node01936", REAL, 0, NULL, 0, NULL, DERIVED, {0},
{1, 3, -1, 4, 1, 2, 0}, 134, 25, 2000, VARIABLE, {sediments_SA_name, sediments_SA_spec, sediments_SA_description, sediments_SA_comment}},
{"node01939", REAL, 0, NULL, 0, NULL, DERIVED, {0},
{1, 3, -1, 4, 1, 3, 0}, 135, 0.1, 100, VARIABLE, {Nitrogen_SA_name, Nitrogen_SA_spec, Nitrogen_SA_description, Nitrogen_SA_comment}},
{"node01943", REAL, 0, NULL, 0, NULL, DERIVED, {0},

{1, 3, -1, 4, 1, 4, 0}, 136, -1.0e+100, 1.0e+100, VARIABLE, {k_SA_name, k_SA_spec, NULL, k_SA_comment}},
{"node01919", VALUELESS, 0, NULL, 0, NULL, SPLIT, {0},
{1, 3, -1, 4, 2, 0}, 137, 0, 0, SUBMODEL, {clock_name, NULL, NULL, clock_comment}},
{"node01944", INTEGER, 0, NULL, 0, NULL, DERIVED, {0},
{1, 3, -1, 4, 2, 1, 0}, 140, -268435455, 268435455, VARIABLE, {month_name, month_spec, month_description, month_comment}},
{"node01946", INTEGER, 0, NULL, 0, NULL, DERIVED, {0},
{1, 3, -1, 4, 2, 2, 0}, 141, -268435455, 268435455, VARIABLE, {days_name, days_spec, NULL, NULL}},
{"node01948", INTEGER, 0, NULL, 0, NULL, DERIVED, {0},
{1, 3, -1, 4, 2, 3, 0}, 142, -268435455, 268435455, VARIABLE, {year_name, year_spec, NULL, NULL}},
{"node01950", INTEGER, 0, NULL, 0, NULL, DERIVED, {0},
{1, 3, -1, 4, 2, 4, 0}, 143, -268435455, 268435455, VARIABLE, {dayofyear_name, dayofyear_spec, NULL, NULL}},
{"node01952", REAL, 0, NULL, 0, NULL, DERIVED, {0},
{1, 3, -1, 4, 2, 5, 0}, 144, -1.0e+100, 1.0e+100, VARIABLE, {dayreal_name, NULL, NULL, NULL}},
{"node01920", VALUELESS, 0, NULL, 0, NULL, SPLIT, {0},
{1, 3, -1, 4, 3, 0}, 145, 0, 0, SUBMODEL, {Temperature_name, NULL, NULL, NULL}},
{"arc00703", REAL, 0, NULL, 0, NULL, DERIVED, {0},
{1, 3, -1, 4, 3, 2, 0}, 149, -1.0e+100, 1.0e+100, FLOW, {flow1_name, flow1_spec, NULL, NULL}},
{"node00905", REAL, 0, NULL, 0, NULL, DERIVED, {0},
{1, 3, -1, 4, 3, 3, 0}, 150, -1.0e+100, 1.0e+100, COMPARTMENT, {daysabove_20box_name, daysabove_20box_spec, NULL, NULL}},
{"node01955", REAL, 0, NULL, 0, NULL, DERIVED, {0},
{1, 3, -1, 4, 3, 4, 0}, 151, -1.0e+100, 1.0e+100, VARIABLE, {Temperature_SA_name, Temperature_SA_spec, Temperature_SA_description, Temperature_SA_comment}},
{"node01957", REAL, 0, NULL, 0, NULL, DERIVED, {0},
{1, 3, -1, 4, 3, 5, 0}, 152, -1.0e+100, 1.0e+100, VARIABLE, {Daysabove20_name, Daysabove20_spec, NULL, Daysabove20_comment}},
{"node01960", REAL, 0, NULL, 0, NULL, DERIVED, {0},
{1, 3, -1, 4, 3, 7, 0}, 154, -1.0e+100, 1.0e+100, COMPARTMENT, {FDegreeDays_name, FDegreeDays_spec, NULL, FDegreeDays_comment}},
{"arc01152", REAL, 0, NULL, 0, NULL, DERIVED, {0},
{1, 3, -1, 4, 3, 8, 0}, 155, -1.0e+100, 1.0e+100, FLOW, {FDays_name, FDays_spec, NULL, NULL}},
{"node01964", REAL, 0, NULL, 0, NULL, DERIVED, {0},
{1, 3, -1, 4, 3, 10, 0}, 157, -1.0e+100, 1.0e+100, COMPARTMENT, {GDegreeDays_name, GDegreeDays_spec, NULL, NULL}},
{"arc01156", REAL, 0, NULL, 0, NULL, DERIVED, {0},
{1, 3, -1, 4, 3, 11, 0}, 158, -1.0e+100, 1.0e+100, FLOW, {GDays_name, GDays_spec, NULL, NULL}},
{"node01921", VALUELESS, 0, NULL, 0, NULL, SPLIT, {0},
{1, 3, -1, 4, 4, 0}, 159, 0, 0, SUBMODEL, {Grass_Machine_name, NULL, NULL, NULL}},
{"node01973", VALUELESS, 0, NULL, 0, NULL, SPLIT, {0},
{1, 3, -1, 4, 4, 1, 0}, 162, 0, 0, SUBMODEL, {SGR_Adult_name, NULL, NULL, NULL}},
{"node01978", REAL, 0, NULL, 0, NULL, DERIVED, {0},
{1, 3, -1, 4, 4, 1, 1, 0}, 165, -1.0e+100, 1.0e+100, VARIABLE, {sediment__limitation_A_name, sediment__limitation_A_spec, sediment__limitation_A_description, sediment__limitation_A_comment}},
{"node01980", REAL, 0, NULL, 0, NULL, DERIVED, {0},
{1, 3, -1, 4, 4, 1, 2, 0}, 166, -1.0e+100, 1.0e+100, VARIABLE, {SGR_A_name, SGR_A_spec, SGR_A_description, SGR_A_comment}},

{"node01982", REAL, 0, NULL, 0, NULL, DERIVED, {0},
{1, 3, -1, 4, 4, 1, 3, 0}, 167, -1.0e+100, 1.0e+100, VARIABLE, {umax_A_name, umax_A_spec, umax_A_description, umax_A_comment}},
{"node01984", REAL, 0, NULL, 0, NULL, DERIVED, {0},
{1, 3, -1, 4, 4, 1, 4, 0}, 168, -1.0e+100, 1.0e+100, VARIABLE, {alpha_A_name, alpha_A_spec, alpha_A_description, alpha_A_comment}},
{"node01986", REAL, 0, NULL, 0, NULL, DERIVED, {0},
{1, 3, -1, 4, 4, 1, 5, 0}, 169, -1.0e+100, 1.0e+100, VARIABLE, {ro_A_name, ro_A_spec, ro_A_description, ro_A_comment}},
{"node01988", REAL, 0, NULL, 0, NULL, DERIVED, {0},
{1, 3, -1, 4, 4, 1, 6, 0}, 170, -1.0e+100, 1.0e+100, VARIABLE, {Set_GMAXNW_A_name, Set_GMAXNW_A_spec, Set_GMAXNW_A_description, Set_GMAXNW_A_comment}},
{"node01974", VALUELESS, 0, NULL, 0, NULL, SPLIT, {0},
{1, 3, -1, 4, 4, 2, 0}, 171, 0, 0, SUBMODEL, {SGR_Seedling_name, NULL, NULL, NULL}},
{"node02096", REAL, 0, NULL, 0, NULL, DERIVED, {0},
{1, 3, -1, 4, 4, 2, 1, 0}, 174, -1.0e+100, 1.0e+100, VARIABLE, {SGR_S_name, SGR_S_spec, SGR_S_description, SGR_S_comment}},
{"node02098", REAL, 0, NULL, 0, NULL, DERIVED, {0},
{1, 3, -1, 4, 4, 2, 2, 0}, 175, -1.0e+100, 1.0e+100, VARIABLE, {umax_S_name, umax_S_spec, umax_S_description, umax_S_comment}},
{"node02500", REAL, 0, NULL, 0, NULL, DERIVED, {0},
{1, 3, -1, 4, 4, 2, 3, 0}, 176, -1.0e+100, 1.0e+100, VARIABLE, {sediment_limitation_S_name, sediment_limitation_S_spec, NULL, sediment_limitation_S_comment}},
{"node02502", REAL, 0, NULL, 0, NULL, DERIVED, {0},
{1, 3, -1, 4, 4, 2, 4, 0}, 177, -1.0e+100, 1.0e+100, VARIABLE, {F_I_T__S_name, F_I_T__S_spec, NULL, F_I_T__S_comment}},
{"node01922", VALUELESS, 0, NULL, 0, NULL, SPLIT, {0},
{1, 3, -1, 4, 5, 0}, 178, 0, 0, SUBMODEL, {Light_name, NULL, NULL, NULL}},
{"node02516", VALUELESS, 0, NULL, 0, NULL, SPLIT, {0},
{1, 3, -1, 4, 5, 1, 0}, 181, 0, 0, SUBMODEL, {Light_Adult_name, NULL, NULL, NULL}},
{"node02522", REAL, 0, NULL, 0, NULL, DERIVED, {0},
{1, 3, -1, 4, 5, 1, 1, 0}, 184, -1.0e+100, 1.0e+100, VARIABLE, {kcanopy_A_name, kcanopy_A_spec, kcanopy_A_description, kcanopy_A_comment}},
{"node02524", REAL, 0, NULL, 0, NULL, DERIVED, {0},
{1, 3, -1, 4, 5, 1, 2, 0}, 185, -1.0e+100, 1.0e+100, VARIABLE, {Percent_Irradiance_A_name, Percent_Irradiance_A_spec, NULL, Percent_Irradiance_A_comment}},
{"node02526", REAL, 0, NULL, 0, NULL, DERIVED, {0},
{1, 3, -1, 4, 5, 1, 3, 0}, 186, -1.0e+100, 1.0e+100, VARIABLE, {Izleaf_A_name, Izleaf_A_spec, NULL, Izleaf_A_comment}},
{"node02517", VALUELESS, 0, NULL, 0, NULL, SPLIT, {0},
{1, 3, -1, 4, 5, 2, 0}, 187, 0, 0, SUBMODEL, {Light_Seedling_name, NULL, NULL, NULL}},
{"node02538", REAL, 0, NULL, 0, NULL, DERIVED, {0},
{1, 3, -1, 4, 5, 2, 1, 0}, 190, 1, 64, VARIABLE, {Surface_PAR_S_name, Surface_PAR_S_spec, Surface_PAR_S_description, Surface_PAR_S_comment}},
{"node02539", REAL, 0, NULL, 0, NULL, DERIVED, {0},
{1, 3, -1, 4, 5, 2, 2, 0}, 191, -1.0e+100, 1.0e+100, VARIABLE, {Iz_SA_name, Iz_SA_spec, NULL, Iz_SA_comment}},
{"node02541", REAL, 0, NULL, 0, NULL, DERIVED, {0},
{1, 3, -1, 4, 5, 2, 3, 0}, 192, -1.0e+100, 1.0e+100, VARIABLE, {Percent_Irradiance_S_name, Percent_Irradiance_S_spec, NULL, Percent_Irradiance_S_comment}},
{"node01760", VALUELESS, 0, NULL, 0, NULL, SPLIT, {MEMBERS, 0},
{1, 3, -1, 5, -1, 0}, 193, 0, 0, SUBMODEL, {Node_births_and_lengths_name, NULL, NULL, Node_births_and_lengths_comment}},
{"node02545", REAL, 0, NULL, 0, NULL, DERIVED, {0},

{1, 3, -1, 5, -1, 4, 0}, 196, -1.0e+100, 1.0e+100, COMPARTMENT, {my_internode_length_name, my_internode_length_spec, NULL, my_internode_length_comment}},
{"node02549", INTEGER, 0, NULL, 0, NULL, DERIVED, {0},
{1, 3, -1, 5, -1, 5, 0}, 197, -268435455, 268435455, VARIABLE, {index_name, index_spec, NULL, index_comment}},
{"node02555", REAL, 0, NULL, 0, NULL, DERIVED, {0},
{1, 3, -1, 5, -1, 6, 0}, 198, -1.0e+100, 1.0e+100, VARIABLE, {node_age_name, node_age_spec, NULL, NULL}},
{"node02557", REAL, 0, NULL, 0, NULL, DERIVED, {0},
{1, 3, -1, 5, -1, 7, 0}, 199, -1.0e+100, 1.0e+100, VARIABLE, {node_birthday_name, node_birthday_spec, NULL, node_birthday_comment}},
{"node02548", REAL, 0, NULL, 0, NULL, DERIVED, {0},
{1, 3, -1, 5, -1, 8, 0}, 200, -1.0e+100, 1.0e+100, COMPARTMENT, {NodeXNW_name, NodeXNW_spec, NULL, NodeXNW_comment}},
{"node02564", REAL, 0, NULL, 0, NULL, DERIVED, {0},
{1, 3, -1, 5, -1, 9, 0}, 201, -1.0e+100, 1.0e+100, COMPARTMENT, {NodeY_name, NodeY_spec, NULL, NodeY_comment}},
{"node01903", VALUELESS, 0, NULL, 0, NULL, SPLIT, {0},
{1, 3, -1, 6, 0}, 202, 0, 0, SUBMODEL, {Adulthood_name, NULL, NULL, NULL}},
{"node02570", FLAG, 0, NULL, 0, NULL, DERIVED, {0},
{1, 3, -1, 6, 1, 0}, 205, 0, 1, VARIABLE, {is_adult_name, is_adult_spec, NULL, is_adult_comment}},
{"node01908", VALUELESS, 0, NULL, 0, NULL, SPLIT, {0},
{1, 3, -1, 7, 0}, 206, 0, 0, SUBMODEL, {Flower_name, NULL, NULL, NULL}},
{"node02576", FLAG, 0, NULL, 0, NULL, DERIVED, {0},
{1, 3, -1, 7, 1, 0}, 209, 0, 1, VARIABLE, {flower_name, flower_spec, NULL, flower_comment}},
{"node01645", INTEGER, 0, NULL, 0, NULL, DERIVED, {0},
{1, 3, -1, 8, 0}, 210, -268435455, 268435455, VARIABLE, {Branching_Rate_Fix_name, Branching_Rate_Fix_spec, NULL, Branching_Rate_Fix_comment}},
{"node01648", INTEGER, 0, NULL, 0, NULL, DERIVED, {0},
{1, 3, -1, 10, 0}, 212, -268435455, 268435455, LOSS, {Negative_Root_biomass_0_name, Negative_Root_biomass_0_spec, NULL, Negative_Root_biomass_0_comment}},
{"node01650", INTEGER, 0, NULL, 0, NULL, DERIVED, {0},
{1, 3, -1, 11, 0}, 213, -268435455, 268435455, VARIABLE, {countflowers_name, countflowers_spec, NULL, NULL}},
{"node01653", INTEGER, 0, NULL, 0, NULL, DERIVED, {0},
{1, 3, -1, 12, 0}, 214, -268435455, 268435455, VARIABLE, {adult_status_name, adult_status_spec, NULL, NULL}},
{"node01669", INTEGER, 0, NULL, 0, NULL, DERIVED, {0},
{1, 3, -1, 14, 0}, 216, -268435455, 268435455, VARIABLE, {Me_0_name, Me_0_spec, NULL, NULL}},
{"node01671", INTEGER, 0, NULL, 0, NULL, DERIVED, {0},
{1, 3, -1, 15, 0}, 217, -268435455, 268435455, VARIABLE, {seeds_perflowerShoot_name, seeds_perflowerShoot_spec, NULL, seeds_perflowerShoot_comment}},
{"node01661", REAL, 0, NULL, 0, NULL, DERIVED, {0},
{1, 3, -1, 16, 0}, 218, -1.0e+100, 1.0e+100, VARIABLE, {Initial_Density_name, Initial_Density_spec, NULL, Initial_Density_comment}},
{"node01665", REAL, 0, NULL, 0, NULL, DERIVED, {0},
{1, 3, -1, 17, 0}, 219, -1.0e+100, 1.0e+100, COMPARTMENT, {LeavesProduced_name, LeavesProduced_spec, NULL, NULL}},
{"arc00918", REAL, 0, NULL, 0, NULL, DERIVED, {0},
{1, 3, -1, 18, 0}, 220, -1.0e+100, 1.0e+100, FLOW, {Pl_name, Pl_spec, NULL, NULL}},
{"node01686", REAL, 0, NULL, 0, NULL, DERIVED, {0},
{1, 3, -1, 19, 0}, 221, -1.0e+100, 1.0e+100, VARIABLE, {init_y_name, init_y_spec, NULL, init_y_comment}},

{"node01685", REAL, 0, NULL, 0, NULL, DERIVED, {0},
{1, 3, -1, 20, 0}, 222, -1.0e+100, 1.0e+100, VARIABLE, {init_XNW_name, init_XNW_spec,
NULL, init_XNW_comment}},
{"node01690", REAL, 0, NULL, 0, NULL, DERIVED, {0},
{1, 3, -1, 21, 0}, 223, -1.0e+100, 1.0e+100, VARIABLE, {check_b_adult___fix_name,
check_b_adult___fix_spec, NULL, NULL}},
{"node01700", REAL, 0, NULL, 0, NULL, DERIVED, {0},
{1, 3, -1, 22, 0}, 224, -1.0e+100, 1.0e+100, COMPARTMENT, {RHIZOME_DETRITUS_name,
RHIZOME_DETRITUS_spec, NULL, RHIZOME_DETRITUS_comment}},
{"node01754", REAL, 0, NULL, 0, NULL, DERIVED, {0},
{1, 3, -1, 23, 0}, 225, -1.0e+100, 1.0e+100, COMPARTMENT, {LEAFDETRITUS_name,
LEAFDETRITUS_spec, NULL, LEAFDETRITUS_comment}},
{"arc01682", REAL, 0, NULL, 0, NULL, DERIVED, {0},
{1, 3, -1, 24, 0}, 226, -1.0e+100, 1.0e+100, FLOW, {Gr_name, Gr_spec, NULL, NULL}},
{"arc01698", REAL, 0, NULL, 0, NULL, DERIVED, {0},
{1, 3, -1, 25, 0}, 227, -1.0e+100, 1.0e+100, FLOW, {Gl_name, Gl_spec, NULL, NULL}},
{"arc01714", REAL, 0, NULL, 0, NULL, DERIVED, {0},
{1, 3, -1, 27, 0}, 229, -1.0e+100, 1.0e+100, FLOW, {Decomp_name, Decomp_spec, NULL,
NULL}},
{"node01702", REAL, 0, NULL, 0, NULL, DERIVED, {0},
{1, 3, -1, 28, 0}, 230, -1.0e+100, 1.0e+100, VARIABLE, {Rhizome_Length_name,
Rhizome_Length_spec, NULL, Rhizome_Length_comment}},
{"node01707", REAL, 0, NULL, 0, NULL, DERIVED, {0},
{1, 3, -1, 29, 0}, 231, -1.0e+100, 1.0e+100, VARIABLE, {My_birthday_name, My_birthday_spec,
NULL, My_birthday_comment}},
{"node01710", REAL, 0, NULL, 0, NULL, DERIVED, {0},
{1, 3, -1, 31, 0}, 233, -1.0e+100, 1.0e+100, COMPARTMENT, {Y_name, Y_spec, NULL,
Y_comment}},
{"node01712", REAL, 0, NULL, 0, NULL, DERIVED, {0},
{1, 3, -1, 33, 0}, 235, -1.0e+100, 1.0e+100, COMPARTMENT, {XNW_name, XNW_spec, NULL,
XNW_comment}},
{"arc01661", REAL, 0, NULL, 0, NULL, DERIVED, {0},
{1, 3, -1, 34, 0}, 236, -1.0e+100, 1.0e+100, FLOW, {Move_YAPEXNW_name,
Move_YAPEXNW_spec, NULL, NULL}},
{"arc01664", REAL, 0, NULL, 0, NULL, DERIVED, {0},
{1, 3, -1, 35, 0}, 237, -1.0e+100, 1.0e+100, FLOW, {Move_XNWAPEXNW_name,
Move_XNWAPEXNW_spec, NULL, NULL}},
{"node01718", REAL, 0, NULL, 0, NULL, DERIVED, {3, 0},
{1, 3, -1, 36, 0}, 238, -1.0e+100, 1.0e+100, VARIABLE, {My_Birthplace_name,
My_Birthplace_spec, NULL, My_Birthplace_comment}},
{"node01720", REAL, 0, NULL, 0, NULL, DERIVED, {3, 0},
{1, 3, -1, 37, 0}, 239, -1.0e+100, 1.0e+100, VARIABLE, {MyCoords_name, MyCoords_spec,
NULL, MyCoords_comment}},
{"node01723", REAL, 0, NULL, 0, NULL, DERIVED, {0},
{1, 3, -1, 38, 0}, 240, -1.0e+100, 1.0e+100, VARIABLE, {age_of__youngest_node_name,
age_of__youngest_node_spec, NULL, age_of__youngest_node_comment}},
{"node01726", REAL, 0, NULL, 0, NULL, DERIVED, {0},
{1, 3, -1, 41, 0}, 243, -1.0e+100, 1.0e+100, COMPARTMENT, {GROWING_INTERNODE_name,
GROWING_INTERNODE_spec, NULL, GROWING_INTERNODE_comment}},
{"node01725", REAL, 0, NULL, 0, NULL, DERIVED, {0},
{1, 3, -1, 42, 0}, 244, -1.0e+100, 1.0e+100, COMPARTMENT, {NODES_name, NODES_spec,
NULL, NODES_comment}},
{"arc01685", REAL, 0, NULL, 0, NULL, DERIVED, {0},
{1, 3, -1, 43, 0}, 245, -1.0e+100, 1.0e+100, FLOW, {In_name, In_spec, NULL, NULL}},
{"arc01688", REAL, 0, NULL, 0, NULL, DERIVED, {0},

{1, 3, -1, 45, 0}, 247, -1.0e+100, 1.0e+100, FLOW, {Io_name, Io_spec, NULL, NULL}},
{"node01734", REAL, 0, NULL, 0, NULL, DERIVED, {0},
{1, 3, -1, 46, 0}, 248, -1.0e+100, 1.0e+100, VARIABLE, {b_A_name, b_A_spec, b_A_description, b_A_comment}},
{"node01731", REAL, 0, NULL, 0, NULL, DERIVED, {0},
{1, 3, -1, 47, 0}, 249, 14, 150, VARIABLE, {Canopy_Height__for_Light_name, Canopy_Height__for_Light_spec, Canopy_Height__for_Light_description, Canopy_Height__for_Light_comment}},
{"node01737", REAL, 0, NULL, 0, NULL, DERIVED, {0},
{1, 3, -1, 49, 0}, 251, -1.0e+100, 1.0e+100, COMPARTMENT, {LEAFBUNDLE_name, LEAFBUNDLE_spec, NULL, LEAFBUNDLE_comment}},
{"arc01704", REAL, 0, NULL, 0, NULL, DERIVED, {0},
{1, 3, -1, 51, 0}, 253, -1.0e+100, 1.0e+100, FLOW, {Ol_name, Ol_spec, NULL, NULL}},
{"node01732", REAL, 0, NULL, 0, NULL, DERIVED, {0},
{1, 3, -1, 52, 0}, 254, -1.0e+100, 1.0e+100, COMPARTMENT, {OLD__NODES_name, OLD__NODES_spec, NULL, OLD__NODES_comment}},
{"node01742", REAL, 0, NULL, 0, NULL, DERIVED, {0},
{1, 3, -1, 53, 0}, 255, -1.0e+100, 1.0e+100, VARIABLE, {Nutrient__Canopy_Factor_name, Nutrient__Canopy_Factor_spec, NULL, Nutrient__Canopy_Factor_comment}},
{"node01744", REAL, 0, NULL, 0, NULL, DERIVED, {0},
{1, 3, -1, 54, 0}, 256, -1.0e+100, 1.0e+100, VARIABLE, {malb_name, malb_spec, NULL, malb_comment}},
{"arc01715", REAL, 0, NULL, 0, NULL, DERIVED, {0},
{1, 3, -1, 56, 0}, 258, -1.0e+100, 1.0e+100, FLOW, {Gi_name, Gi_spec, NULL, NULL}},
{"node01746", REAL, 0, NULL, 0, NULL, DERIVED, {0},
{1, 3, -1, 57, 0}, 259, -1.0e+100, 1.0e+100, COMPARTMENT, {ROOTS___RHIZOMES_name, ROOTS___RHIZOMES_spec, NULL, ROOTS___RHIZOMES_comment}},
{"node01739", REAL, 0, NULL, 0, NULL, DERIVED, {0},
{1, 3, -1, 58, 0}, 260, -1.0e+100, 1.0e+100, COMPARTMENT, {OLDESTLEAF_name, OLDESTLEAF_spec, NULL, OLDESTLEAF_comment}},
{"node01750", REAL, 0, NULL, 0, NULL, DERIVED, {0},
{1, 3, -1, 61, 0}, 263, -1.0e+100, 1.0e+100, REPRODUCTION, {BRANCH_0_name, BRANCH_0_spec, BRANCH_0_description, BRANCH_0_comment}},
{"node01752", REAL, 0, NULL, 0, NULL, DERIVED, {0},
{1, 3, -1, 63, 0}, 265, -1.0e+100, 1.0e+100, LOSS, {Death_0_name, Death_0_spec, Death_0_description, Death_0_comment}},
{"arc01725", REAL, 0, NULL, 0, NULL, DERIVED, {0},
{1, 3, -1, 65, 0}, 267, -1.0e+100, 1.0e+100, FLOW, {Sl_name, Sl_spec, NULL, NULL}},
{"node01756", REAL, 0, NULL, 0, NULL, DERIVED, {0},
{1, 3, -1, 66, 0}, 268, -1.0e+100, 1.0e+100, VARIABLE, {maol_name, maol_spec, maol_description, maol_comment}},
{"node01758", INTEGER, 0, NULL, 0, NULL, DERIVED, {0},
{1, 3, -1, 67, 0}, 269, -268435455, 268435455, VARIABLE, {my_number__of_nodes_name, my_number__of_nodes_spec, NULL, my_number__of_nodes_comment}},
{"node01761", INTEGER, 0, NULL, 0, NULL, DERIVED, {0},
{1, 3, -1, 68, 0}, 270, -268435455, 268435455, VARIABLE, {one_name, one_spec, NULL, one_comment}},
{"node01763", REAL, 0, NULL, 0, NULL, DERIVED, {0},
{1, 3, -1, 69, 0}, 271, -1.0e+100, 1.0e+100, VARIABLE, {average_internode_length_name, average_internode_length_spec, NULL, average_internode_length_comment}},
{"node01765", REAL, 0, NULL, 0, NULL, DERIVED, {0},
{1, 3, -1, 70, 0}, 272, -1.0e+100, 1.0e+100, VARIABLE, {youngest__birthday_name, youngest__birthday_spec, NULL, youngest__birthday_comment}},
{"node01694", REAL, 0, NULL, 0, NULL, DERIVED, {0},

{1, 3, -1, 71, 0}, 273, -1.0e+100, 1.0e+100, VARIABLE, {Canopy_Height_name,
Canopy_Height_spec, NULL, NULL}},
{"node01722", FLAG, 0, NULL, 0, NULL, DERIVED, {0},
{1, 3, -1, 72, 0}, 274, 0, 1, VARIABLE, {Eve_name, Eve_spec, NULL, Eve_comment}},
{"node01769", INTEGER, 0, NULL, 0, NULL, DERIVED, {0},
{1, 3, -1, 73, 0}, 275, -268435455, 268435455, VARIABLE, {Eve_Status_name, Eve_Status_spec,
NULL, Eve_Status_comment}},
{"node01771", INTEGER, 0, NULL, 0, NULL, DERIVED, {0},
{1, 3, -1, 74, 0}, 276, -268435455, 268435455, VARIABLE, {My_Mother_name, My_Mother_spec,
NULL, My_Mother_comment}},
{"node01773", INTEGER, 0, NULL, 0, NULL, DERIVED, {0},
{1, 3, -1, 75, 0}, 277, -268435455, 268435455, VARIABLE, {Me_name, Me_spec, NULL,
Me_comment}},
{"node01775", REAL, 0, NULL, 0, NULL, DERIVED, {0},
{1, 3, -1, 76, 0}, 278, -1.0e+100, 1.0e+100, VARIABLE, {Internode_Length_name,
Internode_Length_spec, NULL, Internode_Length_comment}},
{"node01777", REAL, 0, NULL, 0, NULL, DERIVED, {0},
{1, 3, -1, 77, 0}, 279, -1.0e+100, 1.0e+100, VARIABLE, {Plastochrone_Interval_name,
Plastochrone_Interval_spec, NULL, Plastochrone_Interval_comment}},
{"node01779", REAL, 0, NULL, 0, NULL, DERIVED, {0},
{1, 3, -1, 78, 0}, 280, -1.0e+100, 1.0e+100, VARIABLE, {Di__branch_name, Di__branch_spec,
NULL, NULL}},
{"node01783", REAL, 0, NULL, 0, NULL, DERIVED, {0},
{1, 3, -1, 79, 0}, 281, -1.0e+100, 1.0e+100, VARIABLE, {Timed_Square_Meter_Density_name,
Timed_Square_Meter_Density_spec, NULL, Timed_Square_Meter_Density_comment}},
{"node01781", REAL, 0, NULL, 0, NULL, DERIVED, {0},
{1, 3, -1, 80, 0}, 282, -1.0e+100, 1.0e+100, COMPARTMENT, {DIRECTION_name,
DIRECTION_spec, NULL, DIRECTION_comment}},
{"node01792", REAL, 0, NULL, 0, NULL, DERIVED, {0},
{1, 3, -1, 81, 0}, 283, -1.0e+100, 1.0e+100, VARIABLE, {Squared_Distance_to_rcm_name,
Squared_Distance_to_rcm_spec, NULL, Squared_Distance_to_rcm_comment}},
{"node01786", REAL, 0, NULL, 0, NULL, DERIVED, {0},
{1, 3, -1, 82, 0}, 284, -1.0e+100, 1.0e+100, VARIABLE, {Square_Meter__Density_name,
Square_Meter__Density_spec, NULL, Square_Meter__Density_comment}},
{"node01787", REAL, 0, NULL, 0, NULL, DERIVED, {0},
{1, 3, -1, 83, 0}, 285, -1.0e+100, 1.0e+100, VARIABLE, {initial_ch_name, initial_ch_spec, NULL,
initial_ch_comment}},
{"node01897", REAL, 0, NULL, 0, NULL, DERIVED, {0},
{1, 3, -1, 84, 0}, 286, -1.0e+100, 1.0e+100, VARIABLE, {randomflower_name,
randomflower_spec, NULL, randomflower_comment}},
{"node01899", REAL, 0, NULL, 0, NULL, DERIVED, {0},
{1, 3, -1, 85, 0}, 287, -1.0e+100, 1.0e+100, VARIABLE, {b_S_name, b_S_spec, b_S_description,
b_S_comment}},
{"node01901", REAL, 0, NULL, 0, NULL, DERIVED, {0},
{1, 3, -1, 86, 0}, 288, -1.0e+100, 1.0e+100, VARIABLE, {b_name, b_spec, NULL, NULL}},
{"node01904", REAL, 0, NULL, 0, NULL, DERIVED, {0},
{1, 3, -1, 87, 0}, 289, -1.0e+100, 1.0e+100, VARIABLE, {GI_A_name, GI_A_spec, NULL,
NULL}},
{"node01906", REAL, 0, NULL, 0, NULL, DERIVED, {0},
{1, 3, -1, 88, 0}, 290, -1.0e+100, 1.0e+100, VARIABLE, {GI_S_name, GI_S_spec, NULL,
NULL}},
{"node01910", REAL, 0, NULL, 0, NULL, DERIVED, {0},
{1, 3, -1, 89, 0}, 291, -1.0e+100, 1.0e+100, VARIABLE, {delay_death_name, delay_death_spec,
NULL, delay_death_comment}},
{"node01912", INTEGER, 0, NULL, 0, NULL, DERIVED, {0},

{1, 3, -1, 90, 0}, 292, -268435455, 268435455, VARIABLE, {flower_time_name, flower_time_spec, NULL, NULL}},
{"node02553", REAL, 0, NULL, 0, NULL, DERIVED, {0},
{1, 3, -1, 91, 0}, 293, -1.0e+100, 1.0e+100, CREATION, {Initial___Nodes_name, Initial___Nodes_spec, Initial___Nodes_description, Initial___Nodes_comment}},
{"node02559", REAL, 0, NULL, 0, NULL, DERIVED, {0},
{1, 3, -1, 94, 0}, 296, -1.0e+100, 1.0e+100, IMMIGRATION, {Add_a__node__0_name, Add_a__node__0_spec, Add_a__node__0_description, Add_a__node__0_comment}},
{"node01546", VALUELESS, 0, NULL, 0, NULL, SPLIT, {START_VM, MEMBERS, MEMBERS, END_VM, 0},
{1, 4, -1, 0}, 297, 0, 0, SUBMODEL, {Neighbours_name, NULL, NULL, Neighbours_comment}},
{"node02583", FLAG, 0, NULL, 0, NULL, DERIVED, {0},
{1, 4, -1, 4, 0}, 302, 0, 1, CONDITION, {cond1_name, cond1_spec, NULL, cond1_comment}},
{"node02587", INTEGER, 0, NULL, 0, NULL, DERIVED, {0},
{1, 4, -1, 5, 0}, 303, -268435455, 268435455, VARIABLE, {index_0_name, index_0_spec, NULL, NULL}},
{"node00503", INTEGER, 0, NULL, 0, NULL, DERIVED, {0},
{1, 5, 0}, 304, -268435455, 268435455, VARIABLE, {countME_name, countME_spec, NULL, NULL}},
{"node00605", INTEGER, 0, NULL, 0, NULL, DERIVED, {0},
{1, 6, 0}, 305, -268435455, 268435455, VARIABLE, {sum_adults_name, sum_adults_spec, NULL, NULL}},
{"node01505", REAL, 0, NULL, 0, NULL, DERIVED, {0},
{1, 7, 0}, 306, -1.0e+100, 1.0e+100, VARIABLE, {sum_Roots_name, sum_Roots_spec, NULL, NULL}},
{"node00607", REAL, 0, NULL, 0, NULL, DERIVED, {0},
{1, 8, 0}, 307, -1.0e+100, 1.0e+100, VARIABLE, {NH_GB_TEMP_name, NH_GB_TEMP_spec, NULL, NULL}},
{"node00910", REAL, 0, NULL, 0, NULL, DERIVED, {0},
{1, 9, 0}, 308, -1.0e+100, 1.0e+100, VARIABLE, {NH_GB_PAR_name, NH_GB_PAR_spec, NULL, NULL}},
{"node00909", REAL, 0, NULL, 0, NULL, DERIVED, {0},
{1, 10, 0}, 309, -1.0e+100, 1.0e+100, VARIABLE, {VA_HIB_TEMP_name, VA_HIB_TEMP_spec, NULL, NULL}},
{"node00907", REAL, 0, NULL, 0, NULL, DERIVED, {0},
{1, 11, 0}, 310, -1.0e+100, 1.0e+100, VARIABLE, {abovebelow_ratio_name, abovebelow_ratio_spec, NULL, NULL}},
{"node00914", REAL, 0, NULL, 0, NULL, DERIVED, {0},
{1, 12, 0}, 311, -1.0e+100, 1.0e+100, VARIABLE, {NC_NERRS_TEMP_name, NC_NERRS_TEMP_spec, NULL, NULL}},
{"node01016", REAL, 0, NULL, 0, NULL, DERIVED, {0},
{1, 13, 0}, 312, -1.0e+100, 1.0e+100, VARIABLE, {South_Bay_2013_TEMP_name, South_Bay_2013_TEMP_spec, NULL, NULL}},
{"node00912", REAL, 0, NULL, 0, NULL, DERIVED, {0},
{1, 14, 0}, 313, -1.0e+100, 1.0e+100, VARIABLE, {VA_HIB_PAR_name, VA_HIB_PAR_spec, NULL, NULL}},
{"node01019", REAL, 0, NULL, 0, NULL, DERIVED, {0},
{1, 15, 0}, 314, -1.0e+100, 1.0e+100, VARIABLE, {NC_NERRS_PAR_name, NC_NERRS_PAR_spec, NULL, NULL}},
{"node01024", REAL, 0, NULL, 0, NULL, DERIVED, {0},
{1, 16, 0}, 315, -1.0e+100, 1.0e+100, VARIABLE, {Average_Iz_name, Average_Iz_spec, NULL, NULL}},
{"node01026", REAL, 0, NULL, 0, NULL, DERIVED, {0},
{1, 17, 0}, 316, -1.0e+100, 1.0e+100, VARIABLE, {averageIleaf_name, averageIleaf_spec, NULL, NULL}},

{"node01028", REAL, 0, NULL, 0, NULL, DERIVED, {0},
{1, 18, 0}, 317, -1.0e+100, 1.0e+100, VARIABLE, {avgkcanopy_name, avgkcanopy_spec, NULL, NULL}},
{"node01030", INTEGER, 0, NULL, 0, NULL, DERIVED, {0},
{1, 19, 0}, 318, -268435455, 268435455, VARIABLE, {Total_Branches_of_Eves_name, Total_Branches_of_Eves_spec, NULL, NULL}},
{"node01021", REAL, 0, NULL, 0, NULL, DERIVED, {0},
{1, 20, 0}, 319, -1.0e+100, 1.0e+100, VARIABLE, {NC1_NC2Comb_TEMP_name, NC1_NC2Comb_TEMP_spec, NULL, NULL}},
{"node01033", INTEGER, 0, NULL, 0, NULL, DERIVED, {0},
{1, 21, 0}, 320, -268435455, 268435455, VARIABLE, {sum_Eve_name, sum_Eve_spec, NULL, NULL}},
{"node01035", REAL, 0, NULL, 0, NULL, DERIVED, {0},
{1, 22, 0}, 321, -1.0e+100, 1.0e+100, VARIABLE, {average_Eve_Branching_name, average_Eve_Branching_spec, NULL, average_Eve_Branching_comment}},
{"node01037", REAL, 0, NULL, 0, NULL, DERIVED, {0},
{1, 23, 0}, 322, -1.0e+100, 1.0e+100, VARIABLE, {shoot_density_name, shoot_density_spec, NULL, shoot_density_comment}},
{"node01040", INTEGER, 0, NULL, 0, NULL, DERIVED, {0},
{1, 24, 0}, 323, -268435455, 268435455, VARIABLE, {Number_of__Shoots_name, Number_of__Shoots_spec, NULL, Number_of__Shoots_comment}},
{"node01501", REAL, 0, NULL, 0, NULL, DERIVED, {0},
{1, 25, 0}, 324, -1.0e+100, 1.0e+100, VARIABLE, {sum_LEAFbundle_name, sum_LEAFbundle_spec, NULL, NULL}},
{"node01503", REAL, 0, NULL, 0, NULL, DERIVED, {0},
{1, 26, 0}, 325, -1.0e+100, 1.0e+100, VARIABLE, {total_biomass_name, total_biomass_spec, NULL, NULL}},
{"node01542", REAL, 0, NULL, 0, NULL, DERIVED, {0},
{1, 27, 0}, 326, -1.0e+100, 1.0e+100, VARIABLE, {Total_Rhizome__Length_name, Total_Rhizome__Length_spec, NULL, Total_Rhizome__Length_comment}},
{"node01544", REAL, 0, NULL, 0, NULL, DERIVED, {0},
{1, 28, 0}, 327, -1.0e+100, 1.0e+100, VARIABLE, {Average_Internode__Length_name, Average_Internode__Length_spec, NULL, Average_Internode__Length_comment}},
{"node01547", REAL, 0, NULL, 0, NULL, DERIVED, {0},
{1, 29, 0}, 328, -1.0e+100, 1.0e+100, VARIABLE, {Greatest_Y_name, Greatest_Y_spec, NULL, NULL}},
{"node01549", REAL, 0, NULL, 0, NULL, DERIVED, {0},
{1, 30, 0}, 329, -1.0e+100, 1.0e+100, VARIABLE, {Greatest_XNW_name, Greatest_XNW_spec, NULL, NULL}},
{"node01551", REAL, 0, NULL, 0, NULL, DERIVED, {0},
{1, 31, 0}, 330, -1.0e+100, 1.0e+100, VARIABLE, {Least_XNW_name, Least_XNW_spec, NULL, NULL}},
{"node01553", REAL, 0, NULL, 0, NULL, DERIVED, {0},
{1, 32, 0}, 331, -1.0e+100, 1.0e+100, VARIABLE, {Least_Y_name, Least_Y_spec, NULL, NULL}},
{"node01555", REAL, 0, NULL, 0, NULL, DERIVED, {0},
{1, 33, 0}, 332, -1.0e+100, 1.0e+100, VARIABLE, {averageLL_name, averageLL_spec, NULL, averageLL_comment}},
{"node01557", REAL, 0, NULL, 0, NULL, DERIVED, {0},
{1, 34, 0}, 333, -1.0e+100, 1.0e+100, VARIABLE, {Average_Density_name, NULL, NULL, Average_Density_comment}},
{"node01559", REAL, 0, NULL, 0, NULL, DERIVED, {2, 0},
{1, 35, 0}, 334, -1.0e+100, 1.0e+100, VARIABLE, {rcmXNWY_name, rcmXNWY_spec, NULL, rcmXNWY_comment}},
{"node01561", REAL, 0, NULL, 0, NULL, DERIVED, {0},

```
 {1, 36, 0}, 335,  -1.0e+100,  1.0e+100, VARIABLE, {Radius_of_Gyration_name,
Radius_of_Gyration_spec, NULL, Radius_of_Gyration_comment}},
 {"node01563", INTEGER, 0, NULL, 0, NULL, DERIVED, {0},
 {1, 37, 0}, 336, -268435455, 268435455, VARIABLE, {sumflowers_name, sumflowers_spec, NULL,
sumflowers_comment}},
 {"node01565", INTEGER, 0, NULL, 0, NULL, DERIVED, {0},
 {1, 38, 0}, 337, -268435455, 268435455, VARIABLE, {sum_seeds_name, sum_seeds_spec, NULL,
sum_seeds_comment}},
 {"node01567", REAL, 0, NULL, 0, NULL, DERIVED, {0},
 {1, 39, 0}, 338,  -1.0e+100,  1.0e+100, VARIABLE,
{Seeds_that_will_Germinate__Seed_Bank__name, Seeds_that_will_Germinate__Seed_Bank__spec,
NULL, Seeds_that_will_Germinate__Seed_Bank__comment}},
 {"node01569", REAL, 0, NULL, 0, NULL, DERIVED, {0},
 {1, 40, 0}, 339,  -1.0e+100,  1.0e+100, VARIABLE, {Set_Initial_Density_name,
Set_Initial_Density_spec, NULL, NULL}},
 {"node01571", REAL, 0, NULL, 0, NULL, DERIVED, {0},
 {1, 41, 0}, 340,  -1.0e+100,  1.0e+100, VARIABLE, {delay__seeds_name, delay__seeds_spec,
NULL, delay__seeds_comment}},
 {"node01573", REAL, 0, NULL, 0, NULL, DERIVED, {0},
 {1, 42, 0}, 341,  -1.0e+100,  1.0e+100, VARIABLE, {BINTZ_MESO_PAR_name,
BINTZ_MESO_PAR_spec, NULL, NULL}},
 {"node01575", REAL, 0, NULL, 0, NULL, DERIVED, {0},
 {1, 43, 0}, 342,  -1.0e+100,  1.0e+100, VARIABLE, {BINTZ_MESO_TEMP_name,
BINTZ_MESO_TEMP_spec, NULL, NULL}},
 {"node01577", REAL, 0, NULL, 0, NULL, DERIVED, {0},
 {1, 44, 0}, 343,  -1.0e+100,  1.0e+100, VARIABLE, {sum_BRANCH_name, sum_BRANCH_spec,
NULL, NULL}},
 {"node01579", INTEGER, 0, NULL, 0, NULL, DERIVED, {0},
 {1, 45, 0}, 344, -268435455, 268435455, VARIABLE, {how_many_adults_true_name,
how_many_adults_true_spec, NULL, NULL}},
 {"node01657", REAL, 0, NULL, 0, NULL, DERIVED, {0},
 {1, 48, 0}, 347,  -1.0e+100,  1.0e+100, IMMIGRATION, {Germinated_Seeds_0_name,
Germinated_Seeds_0_spec, NULL, Germinated_Seeds_0_comment}},
 {"node01667", REAL, 0, NULL, 0, NULL, DERIVED, {0},
 {1, 49, 0}, 348,  -1.0e+100,  1.0e+100, CREATION, {Initial_germinated_Seedlings_name,
Initial_germinated_Seedlings_spec, NULL, Initial_germinated_Seedlings_comment}},
 {"node01704", REAL, 0, NULL, 0, NULL, DERIVED, {0},
 {1, 50, 0}, 349,  -1.0e+100,  1.0e+100, CREATION, {Adult__Initialization_name,
Adult__Initialization_spec, NULL, Adult__Initialization_comment}}};
#include <support2.cpp>
```

BIBLIOGRAPHY

Anderson, J. T., Panetta, A. M., & Mitchell-Olds, T. (2012). Evolutionary and Ecological Responses to Anthropogenic Climate Change: Update on Anthropogenic Climate Change. *Plant Physiology*, *160*(4), 1728–1740. http://doi.org/10.1104/pp.112.206219Baly, E. C. C. 1935. The kinetics of photosynthesis. Proclamations of the Royal Society of London Series B 117:218-239.

Arnold, T. M., Zimmerman, R. C., Engelhardt, K. A., Stevenson, J. C. (2017) 21st Century climate change and submerged aquatic vegetation in the Chesapeake Bay.

Bintz, J., & Nixon, S. (2001). Responses of eelgrass Zostera marina seedlings to reduced light. *Marine Ecology Progress Series, 223*, 133-141. Retrieved from http://www.jstor.org/stable/24865348

Bonser, S. P. (2013), High reproductive efficiency as an adaptive strategy in competitive environments. Funct Ecol, 27: 876–885. doi:10.1111/1365-2435.12064

Bouma T.J., De Vries M.B., Low E., Peralta G, Tánczos IC, et al. (2005) Trade-offs related to ecosystem engineering: A case study on stiffness of emerging macrophytes. Ecology 86: 2187–2199

Bradshaw W.E, Holzapfel C.M. Evolutionary response to rapid climate change. Science. 2006;312:1477–1478.

Brush, M. J., Brawley, J. W., Nixon, S. W., & Kremer, J. N. (2002). Modeling phytoplankton production: problems with the Eppley curve and an empirical alternative. *Marine Ecology Progress Series*, *238*, 31-45.

Brylawski, B. J., & Miller, T. J. (2006). Temperature-dependent growth of the blue crab (callinectes sapidus): A molt process approach. Canadian Journal of Fisheries and Aquatic Sciences, 63(6), 1298-1308. Retrieved from https://search.proquest.com/docview/219296396?accountid=14696

Cabello-Pasini, Alejandro, Celina Lara-Turrent, and Richard C. Zimmerman. "Effect of storms on photosynthesis, carbohydrate content and survival of eelgrass populations from a coastal lagoon and the adjacent open ocean." *Aquatic botany* 74.2 (2002): 149-164.

Carr J, D'Odorico P, McGlathery K, Wiberg PL (2012) Stability and resilience of seagrass meadows to seasonal and interannual dynamics and environmental stress. J Geophys Res 117:G01007 doi:10.1029/2011JG001744

Christianen, M. J. A., van Belzen, J., Herman, P. M. J., van Katwijk, M. M., Lamers, L. P. M., van Leent, P. J. M., & Bouma, T. J. (2013). Low-Canopy Seagrass Beds

Still Provide Important Coastal Protection Services. *PLoS ONE, 8*(5), e62413. http://doi.org/10.1371/journal.pone.0062413

Christiaen, B., Lehrter, J., Goff, J., & Cebrian, J. (2016). Functional implications of changes in seagrass species composition in two shallow coastal lagoons. Marine Ecology Progress Series, 557, 111-121. doi:10.3354/meps11847

Churchill, A. C. (1983). Field studies on seed germination and seedling development in Zostera marina L. Aquatic Botany 16:21-29.

Clausen, K.K., Krause-Jensen, D.,Olesen, B., Marba, N. (2014). Seasonality of eelgrass biomass across gradients in temperature and latitude. Marine Ecology Progress Series, 506:71-85.

Cole, S. G., & Moksnes, P. O. (2016). Valuing multiple eelgrass ecosystem services in Sweden: fish production and uptake of carbon and nitrogen. *Frontiers in Marine Science, 2*, 121.

Davis, M., & Shaw, R. (2001). Range Shifts and Adaptive Responses to Quaternary Climate Change. *Science, 292*(5517), 673-679. Retrieved from http://www.jstor.org/stable/3083537

Dennison, W. C., Aller, R. C., & Alberte, R. S. (1987). Sediment ammonium availability and eelgrass (Zostera marina) growth. Marine Biology, 94(3), 469-477.

Duarte CM, Middelburg J, Caraco N. Major role of marine vegetation on the oceanic carbon cycle. Biogeosciences. 2005;2:1–8.

Duarte, C.M. (2014). Global change and the future ocean: a grand challenge for marine science. Frontiers in Marine Science, Specialty Grand Challenge Article.

Duarte, C. M., Borja, A., Carstensen, J., Elliott, M., Krause-Jensen, D., & Marbà, N. (2015). Paradigms in the recovery of estuarine and coastal ecosystems. *Estuaries and Coasts*, *38*(4), 1202-1212.

Dooley, Frederick D., et al. "Tolerance and response of Zostera marina seedlings to hydrogen sulfide." *Aquatic botany* 105 (2013): 7-10.

Fazlioglu, F., Al-Namazi, A., & Bonser, S. P. (2016). Reproductive efficiency and shade avoidance plasticity under simulated competition. Ecology and Evolution, 6(14), 4947–4957. http://doi.org/10.1002/ece3.2254

De Frenne, P., Graae, B. J., Rodríguez-Sánchez, F., Kolb, A., Chabrerie, O., Decocq, G., De Kort, H., De Schrijver, A., Diekmann, M., Eriksson, O., Gruwez, R., Hermy, M., Lenoir, J., Plue, J., Coomes, D. A. and Verheyen, K. (2013), Latitudinal gradients

as natural laboratories to infer species' responses to temperature. J Ecol, 101: 784–795. doi:10.1111/1365-2745.12074

Dennison, W.C., Orth, R.J., Moore, K.A., Stevenson, J.C., Carter, V., Kollar, S., Bergstrom, P.W., Batiuk, R.A., (1993). Assessing water quality with submersed aquatic vegetation. Bioscience 43, 86–94.

Gatsuk L.E., Smirnova O.V., Vorontzova L.I., Zaugolnova L.B., Zhukova L.A. (1980). Age states of plants of various growth forms: a review. Journal of Ecology 68: 675–696.

Green, E. P., & Short, F. T. 2003. World atlas of seagrasses. Berkeley, Calif: University of California Press.

Greiner J.T., McGlathery K.J., Gunnell J., McKee B.A. (2013) Seagrass Restoration Enhances “Blue Carbon” Sequestration in Coastal Waters. PLOS ONE 8(8): e72469.https://doi.org/10.1371/journal.pone.0072469

Grimm, V. et al. 2006. A Standard Protocol for Describing Individual-Based and Agent Based Models. Ecological Modelling. 198. 115-126. 10.1016/j.ecolmodel.2006.04.023.

Grimm, V., Berger, U., DeAngelis, D. L., Polhill, J. G., Giske, J., & Railsback, S. F. 2010. The ODD protocol: A review and first update. Ecological Modelling, 221(23), 2760-2768. DOI: 10.1016/j.ecolmodel.2010.08.019.

Harlin, M. M., Thorne-Miller, B. (1981). Nutrient enrichment of seagrass beds in a Rhode Island coastal lagoon. Mar. Biol. 65: 221-229.

Harris, L. A. (2006) The Virtual Eelgrass Meadow. (Doctoral dissertation).

Harrison, P. G. (1991). Mechanisms of seed dormancy in an an- nual population of Zostera marina (eelgrass) from the Netherlands. Canadian Journal of Botany 69:1972-197

Harwell, M., & Orth, R. (2002). Seed Bank Patterns in Chesapeake Bay Eelgrass (Zostera marina L.): A Bay-Wide Perspective. *Estuaries, 25*(6), 1196-1204.

Hemminga, M. A., & Duarte, C. M. (2000). Seagrass ecology. Cambridge University Press.

Hemminga, M. A., Koutstaal, B. P., Van Soelen, J., & Merks, A. G. A. (1994). The nitrogen supply to intertidal eelgrass (Zostera marina). Marine Biology, 118(2), 223-227.

Hoffmann, A. A., & Sgrò, C. M. (2011). Climate change and evolutionary adaptation. *Nature*, *470*(7335), 479-485.

Hughes, A.R., Stachowicz, J.J. & Williams, S.L. Oecologia (2009) 159: 725. doi:10.1007/s00442-008-1251-3

Idso, S. B., Jackson, R. D. and Reginato, R. J. (1978), Extending the "Degree Day" Concept of Plant Phenological Development to Include Water Stress Effects. Ecology, 59: 431–433. doi:10.2307/1936570

Jarvis, J. C., Brush, M. J., & Moore, K. A. (2014). Modeling loss and recovery of Zostera marina beds in the Chesapeake Bay: The role of seedlings and seed-bank viability. Aquatic Botany, 113, 32-45. doi:10.1016/j.aquabot.2013.10.010

Jarvis J.C., Moore K.A., & Kenworthy W.J. (2012). Characterization and ecological implication of eelgrass life history strategies near the species' southern limit in the western North Atlantic. Marine Ecology Progress Series, 444, 43-56. doi:10.3354/meps09428

Jassby, A. D. and T. Platt. 1976. Mathematical formulation of the relationship between photosynthesis and light for phytoplankton. Limnology and Oceanography 21:540-547.

Johnson, A. J., Moore, K. A., & Orth, R. J. (2017). The influence of resource availability on flowering intensity in Zostera marina (L.). *Journal of Experimental Marine Biology and Ecology*, *490*, 13-22.

Kemp, W., Boynton, W., Adolf, J., Boesch, D., Boicourt, W., Brush, G., . . . Stevenson, J. (2005). Eutrophication of Chesapeake Bay: Historical trends and ecological interactions. *Marine Ecology Progress Series, 303,* 1-29. Retrieved from http://www.jstor.org/stable/24869811

Kilminster, K., McMahon, K., Waycott, M., Kendrick, G. A., Scanes, P., McKenzie, L., ...Udy, J. (2015). Unravelling complexity in seagrass systems for management: Australia as a microcosm. Science Of The Total Environment, 534, 97-109. doi:10.1016/j.scitotenv.2015.04.061

Koch EW, Barbier EB, Silliman BR, Reed DJ, Perillo GME, et al. (2009) Non-linearity in ecosystem services: temporal and spatial variability in coastal protection. Frontiers in Ecology and the Environment 7: 29–37

Koch, M. S. (2016). Impacts and effects of ocean warming on seagrass. In: Laffoley, D., & Baxter, J.M. (editors). 2016. Explaining ocean warming: Causes, scale, effects and consequences. Full report. Gland, Switzerland: IUCN. pp. 121-130.

Kim S.H., Kim J.H., Park S.R., Lee K.S. 2014. Annual and perennial life history strategies of *Zostera marina* populations under different light regimes. Mar Ecol Prog Ser 509:1-13. https://doi.org/10.3354/meps10899
Lee, K. S., Park, S. R., & Kim, Y. K. (2007). Effects of irradiance, temperature, and nutrients on growth dynamics of seagrasses: a review. *Journal of Experimental Marine Biology and Ecology*, *350*(1), 144-175.

Les, D. (1988). Breeding Systems, Population Structure, and Evolution in Hydrophilous Angiosperms. *Annals of the Missouri Botanical Garden*, *75*(3), 819-835. doi:10.2307/2399370

Manley, S. R., Orth, R. J., & Ruiz-Montoya, L. (2015). Roles of dispersal and predation in determining seedling recruitment patterns in a foundational marine angiosperm. Marine Ecology Progress Series, 533, 109-120.

Marsh, Jr., J. A., Dennison, W. C., & Alberte, R. S. (1986). Effects of temperature on photosynthesis and respiration of eelgrass (Zostera marina L.). Journal of Experimental Marine Biology and Ecology, 101, 257-267.

Marsden, B.W. (2015). Enhancing Vallisneria americana restoration using knowledge of genotypic and phenotypic diversity. (Doctoral dissertation).

Mcleod, E., Chmura, G. L., Bouillon, S., Salm, R., Björk, M., Duarte, C. M., ... & Silliman, B. R. (2011). A blueprint for blue carbon: toward an improved understanding of the role of vegetated coastal habitats in sequestering CO2. Frontiers in Ecology and the Environment, 9(10), 552-560.

Miller-Rushing, A.J. & Primack, R.B. (2008) Global warming and flowering times in Thoreau's Concord: a community perspective. *Ecology*, 89, 332–341.

Murphy, R., L. Orzetti and W. Johnson. 2011. Plant fact sheet for eelgrass (*Zostera marina).* USDA, Natural Resources Conservation Service, Norman A. Berg National Plant Materials Center. Beltsville, MD 20705

Moore, K. A., R.J. Orth, and J. F. Nowak. (1993). Environmental regulation of seed germination in Zostera marina L. (eelgrass) in Chesapeake Bay: Effects of light, oxygen and sediment burial. Aquatic Botany 45:79-9

NOAA (National Oceanic and Atmospheric Administration). 2016. Extended reconstructed sea surface temperature (ERSST.v4). National Centers for Environmental Information. Accessed March 2016. www.ncdc.noaa.gov/data-access/marineocean-data/extended-reconstructed-sea-surface-temperature-ersst.

Olesen, B., 1999. Reproduction in Danish eelgrass (Zostera marina L.) stands: size dependence and biomass partitioning. Aquat. Bot. 65, 209–219.

Olesen, B., & Sand-Jensen, K. (1993). Seasonal acclimatization of eelgrass Zostera marina growth to light. Marine Ecology Progress Series, 94, 91-99. doi:10.3354/meps094091

Olesen, B., Sand-Jensen, K., 1994. Demography of shallow eelgrass (Zostera marina) populations–shoot dynamics and biomass development. J. Ecol. 82, 379–390

Orth RJ, et al. A global crisis for seagrass ecosystems. Bioscience. 2006;56:987–996.

Orth, R. J., S. R. Marion, and K. A. Moore. 2007. A summary of eelgrass (*Zostera marina*) reproductive biology with an emphasis on seed biology and ecology from the Chesapeake Bay region. SAV Technical Notes Collection. ERDC/TN SAV-07-1. Vicksburg, MS: U.S. Army Engineer Research and Development Center.

Plummer, M., Harvey, C., Anderson, L., Guerry, A., & Ruckelshaus, M. (2013). The Role of Eelgrass in Marine Community Interactions and Ecosystem Services: Results from Ecosystem-Scale Food Web Models. *Ecosystems, 16*(2), 237-251. Retrieved from http://www.jstor.org/stable/23501483

Railsback, S.F., 2001. Concepts from complex adaptive systems as a framework for individual-based modelling. Ecol. Model. 139, 47–62

Ralph, P. J., Tomasko, D., Moore, K., Seddon, S., & Macinnis-Ng, C. M. (2007). Human impacts on seagrasses: eutrophication, sedimentation, and contamination. In SEAGRASSES: BIOLOGY, ECOLOGY AND CONSERVATION (pp. 567-593). Springer Netherlands. Chicago

Robbins WJ. 1957. Physiological aspects of aging in plants. American Journal of Botany 44: 289–294.

Root TL , et al.2003 Fingerprints of global warming on wild animals and plants. Nature 421:57–60.

Palacios, S.L., & Zimmerman, R.C. 2007. Response of eelgrass Zostera marina to co2 enrichment: Possible impacts of climate change and potential for remediation of coastal habitats. Marine Ecology Progress Series, 344, 1-13. doi: 10.3354/meps07084
Short, F. T., & Neckles, H. A. (1999). The effects of global climate change on seagrasses. Aquatic Botany, 63(3), 169-196. doi:10.1016/S0304-3770(98)00117-X

Roberts, M., R.J. Orth, and K.A. Moore 1987. Growth of Zostera marina L. seedlings under laboratory conditions of nutrient enrichment. Aquatic Botany. 20:321- 328.

Short, F., 1983. The seagrass, Zostera marina L, plant morphology and bed structure in relation to sediment ammonium in Izembek Lagoon, Alaska. Aquat. Bot. 16, 149–161.

Short, F. T., R.G. Coles, and C. Pergent-Martini. 2001. "Global seagrass distribution." *Global seagrass research methods*: 5-30.

Silberhorn, G. M., Orth, R. J., & Moore, K. A. (1983). Anthesis and seed production in Zostera marina L. (eelgrass) from the chesepeak by. Aquatic Botany, 15(2), 133-144. doi:10.1016/0304-3770(83)90024-4

Steffen, W., Sanderson, R. A., Tyson, P. D., Jäger, J., Matson, P. A., Moore, B. III., et al. 2006. Global Change and the Earth System: A Planet Under Pressure, Berlin: Springer.

Taylor, A.R.A., 1957. Studies on the development of *Zostera marina* L. II. Germination and seedling development. Can. J. Bot., 35: 681-695.

Tooke, F., N. H. Battey. 2010. Temperate flowering phenology, Journal of Experimental Botany, Volume 61, Issue 11, Pages 2853–2862, https://doi.org/10.1093/jxb/erq165.

Williams, S. E., Shoo, L. P., Isaac, J. L., Hoffmann, A. A., & Langham, G. (2008). Towards an Integrated Framework for Assessing the Vulnerability of Species to Climate Change. *PLoS Biology*, *6*(12), e325. http://doi.org.proxy-um.researchport.umd.edu/10.1371/journal.pbio.0060325

Van der Heide T., Eklöf J.S., van Nes E.H., van der Zee E.M., Donadi S, et al. (2012) Ecosystem Engineering by Seagrasses Interacts with Grazing to Shape an Intertidal Landscape. PLoS One7(8): e42060.

Zimmerman, R.C., R.D. Smith, and R.S. Alberte. 1987. "Is growth of eelgrass nitrogen limited? A numerical simulation of the effects of light and nitrogen on the growth dynamics of Zostera marina." Marine Ecology Progress Series 41.2.

Zimmerman, R. C., Hill, V. J., & Gallegos, C. L. (2015). Predicting effects of ocean warming, acidification, and water quality on Chesapeake region eelgrass. *Limnology and Oceanography*, *60*(5), 1781-1804.