

**Leaf, Bud, and Branch Survival and Carbon Transportation in
ECOPHYS**

A THESIS
SUBMITTED TO THE FACULTY OF THE GRADUATE
SCHOOL OF THE UNIVERSITY OF MINNESOTA
BY

KYLE ROSKOSKI

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
MASTER OF SCIENCE

SEPTEMBER 2005

University of Minnesota

This is to certify that I have examined this bound copy of a
Master's Thesis by

Kyle Roskoski

And have found that it is complete and satisfactory in all respects,
and that any and all revisions required by the final examining
committee have been made.

KATHRYN E. LENZ

Name of Faculty Advisor

Signature of Faculty Advisor

Date

GRADUATE SCHOOL

ABSTRACT

In order to simulate multi-year forest tree and patch growth under varying environmental conditions it is essential to incorporate the within-tree growth and death dynamics of buds, leaves, and branches. As a forest tree grows over multiple years, some of its leaves and branches become stressed from environmental factors such as competition for light and air pollution. Leaves and branches die as the intensity and duration of stress increases. This affects tree architecture, size, and competitive advantage for light interception. This thesis presents and analyzes submodels within the multi-year juvenile *Populus* tree growth simulation ECOPHYS. The submodels concern the daily growth or death of individual buds, leaves, shoots and branches. They are each functions of daily carbon available for the growth and maintenance of the particular bud, leaf, shoot or branch. The models of leaf death, bud set and shoot death feature fading memory weighted averages of available photosynthate as well as probabilistic components accounting for unmodeled dynamics. The mathematical properties of various fading memory weighted averages are analyzed and discussed as options for inclusion in these three models. The exponential moving average was chosen because it is approximately equal to an exponential decay function weighted average and it gives increased computational performance. A model of spring bud dynamics is presented which determines the locations within the canopy of new shoots and their relative carbon sink strengths during the deterministic phase of growth. Modeling is also developed concerning the allocation of photosynthate from source leaves to shoot, branch and trunk internodes, and the response of the allocation pattern as individual shoots set bud over the course of each growing season.

Keywords: Weighted Averages, Plant Physiological Process Models, ECOPHYS, Mature Green-Leaf Death, Branch Death, Bud Dynamics, Photosynthate Allocation, Carbon Transportation.

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to my advisor, Dr. Kathryn Lenz, who brought me into the ECOPHYS research group as an undergraduate and has guided me through graduate school. Dr. Lenz is one of the most caring people I know and she has made many valuable suggestions and improvements contributing to this thesis.

I would like to thank Dr. Harlan Stech, who served on my examination committee. Dr. Stech has helped me with visualization software and with modeling. His modeling with dynamical systems course contributed greatly to my modeling skills and insight.

I would also like to thank Dr. George Host, Natural Resources Research Institute. Dr. Host provided significant guidance in Biology and Ecology. He helped to develop and confirm models described in this thesis.

I would like to thank Yongcheng Qi for serving on my committee. I have learned a lot about developing and testing regression models in the classes that I have taken with him over the last year.

I would like to thank all of the graduate and undergraduate students that I have worked with on this project over the last four years. There are too many to list, but you know who you are.

I would like to thank my wife, Jessica Roskoski, for her encouragement and patience with me through my trials during the writing of this thesis.

This work was funded jointly by the Computational Biology Program of the National Science Foundation, Grand No. DBI-972395, the Northern Global Change Program of the USDA Forest Service, the US Department of Energy under Interagency Agreement #DE-A105-800R20763, and the USDA Forest Service North Central Research Station Integrated Program, Agreement 03-JV-11231300-086.

Table of Contents

1. Introduction	
1.1 ECOPHYS	1
1.2 Aspen Free Air Carbon Dioxide Enrichment Experiment	3
1.3 Preview	4
2. Process Models	5
2.1 Modeling Mature Green-Leaf Death	6
2.2 Bud Set Due to Stress	9
2.3 Shoot and Branch Death	13
2.3.1 Shoot-Death Algorithm.....	14
2.3.2 Older Wood Branch-Death Algorithm.....	15
2.4 Bud Vigor and Death	16
2.5 Simulating Genetically - Determined Bud Set.....	19
3. Fading Memory Weighted Averages	20
3.1 Weighted Averages in ECOPHYS.....	20
3.2 Definition of a Weighted Average.....	21
3.3 Types of Weighted Averages.....	22
3.3.1 Linearly Weighted Averages	22
3.3.2 Exponential Decay Function Weights	23
3.3.3 A Reduced Range for α	25
3.3.4 Other Possible Types of Weighted Averages	27
3.4 Exponential Decay Weighted Average and Exponential Moving Average.....	30
3.4.1 Exponential Moving Average.....	30
3.4.2 Exponential Moving Average vs. Exponential Decay Average.....	31
3.4.3 Initial Conditions	34
3.5 Selection of α	35
3.6 New Averaging Scheme for ECOPHYS.....	38
4. Carbon Transportation	44
4.1 Combining Internodes.....	44
4.2 Distribution of Photosynthate Along a Branch	46
4.3 Changes in Carbon Allocation Following Bud Set.....	49
5. Conclusions and Discussion of Future Work.....	51
5.1 Calibration of Process Models	52
5.2 Analysis of FACE Data	55
5.3 Carbon Transportation	56
6. References.....	57
7. Appendices	61
Appendix A: Glossary of Terms	61
Appendix B: Other ECOPHYS Work.....	63
Appendix C: Technical Analysis Paper for Exponential Moving Average	72
Appendix D: Code for Process Models and Carbon Transportation.....	74

Chapter 1 - Introduction

1.1 ECOPHYS

ECOPHYS is a computer simulation designed to simulate the growth of clones of hybrid poplar (*Populus 'eurAmericana'*) and trembling aspen (*Populus Tremuloides*) at the organ, tree and patch levels. During the last six years, ECOPHYS has been developed within the context of trembling aspen. By design, ECOPHYS is used to model tree growth under conditions of global warming including elevated levels of carbon dioxide (CO₂) and ozone (O₃) gases, and variable temperature, light, and humidity. The model was first created in the late 1980's by Dr. Jud Isebrands, Dr. Michael Rauscher, Dr. George Host, and collaborators (Host, 1990). Since then it has been modified by many people including Dr. Host, Professor Kathryn Lenz, Professor Harlan Stech, and their graduate and undergraduate students. Some of the key features in ECOPHYS include leaf by leaf shading (Guan, 2002), carbon transportation within the tree (Zang, 1999), an analytic calculation of photosynthesis using the WIMOVAC model (Zhao, 2000), multiple and parallel simulations on a Beowulf cluster of computers (Holden, 2004), and simulation of the competition for light among neighboring trees (Stech, et al. 2004). Student research contributions to the development of ECOPHYS in recent years are summarized in Lenz and Stech, (2000), Tordsen, (2003), and Holden, (2004).

ECOPHYS performs tasks at three time step intervals: hourly, daily, and seasonally. Every hour ECOPHYS performs shade estimation and photosynthetic rate calculation for each leaf within the canopy. Each day ECOPHYS transports carbon and grows branches

and internodes. For each year, ECOPHYS simulates bud break, determinate growth, indeterminate growth, bud set, fall senescence, and dormancy.

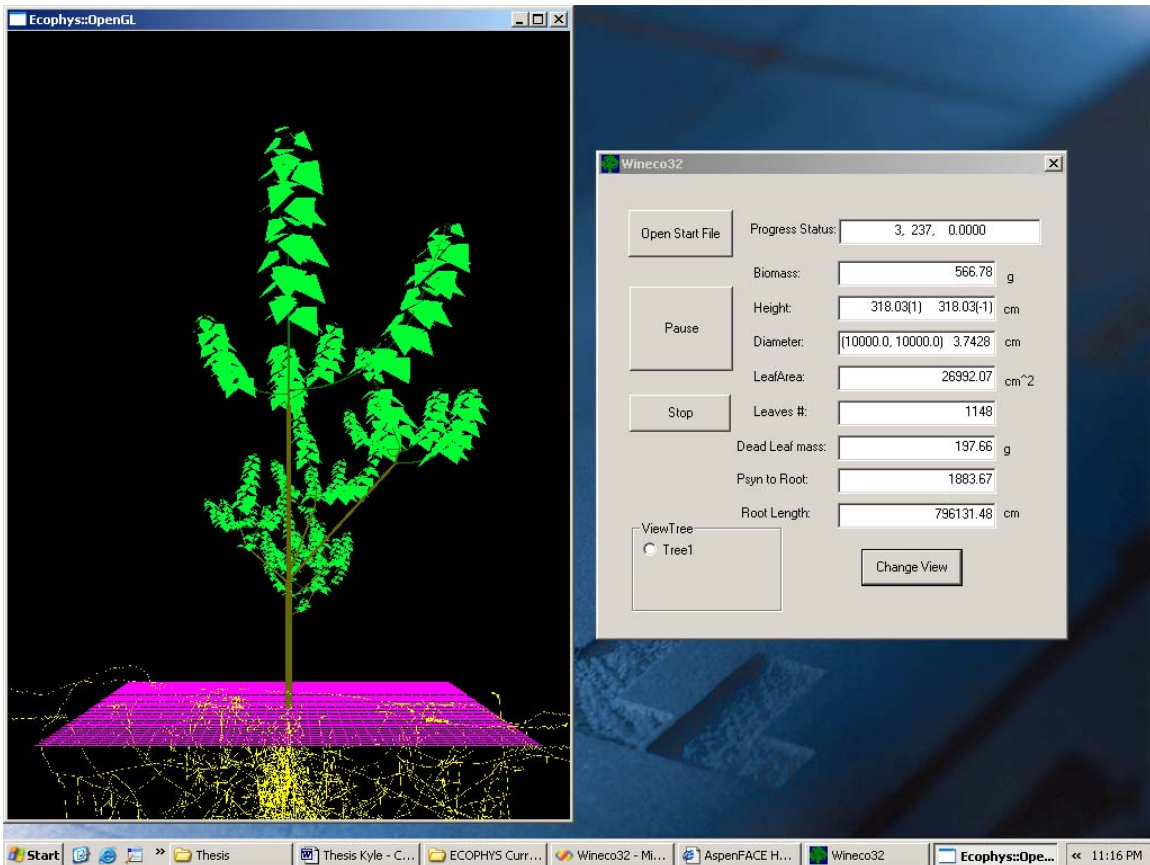


Figure 1.1 - An ECOPHYS tree growing in its 3rd year

One of ECOPHYS's inputs is a weather file read daily containing light intensity (measured in Photosynthetic Active Radiation (PAR)), temperature, relative humidity, carbon dioxide and ozone levels. ECOPHYS also reads in a genetic file containing bud-break date, bud-set date, fall senescence start date and rate, leaf initiation rate, and other clone-specific parameter values. ECOPHYS outputs include daily leaf count, leaf area, height, diameter, root length and other measurements. Further details can be found in Host (1990), Zang (1999) and Guan (2002). ECOPHYS's website can be found at <http://oden.nrri.umn.edu/ECOPHYS/>

1.2 Aspen Free Air Carbon Dioxide Enrichment Experiment

The Forest-Atmosphere Carbon Transfer and Storage II (FACTS II): Aspen Free Air Carbon Dioxide Enrichment (FACE) experiment was established in 1997 near Rhinelander, WI. The purpose of this experiment is to examine the interacting effects of elevated CO_2 and O_3 gases, alone and in combination, on the resultant productivity, sustainability, competitive interactions and carbon and nitrogen fluxes in a regenerating, northern hardwood ecosystem under field conditions over its life history (Dickson et al. 2000). At the site, there are 12 rings that originally had 660 trees each planted with trembling aspen, sugar maple, and paper birch. Three of these rings are considered control rings, where the trees grow with ambient levels of CO_2 and O_3 gases. Of the remaining rings, three are grown with elevated CO_2 , three with elevated O_3 , and three with both elevated CO_2 and elevated O_3 .



Figure 1.2 - An overview of the FACE site

The ECOPHYS research group continues to collaborate with researchers at the FACE experiment in order to replicate the FACE ring aspen growth. In particular, we currently collaborate with FACE investigators Dr. Neil Nelson, Dr. Mark Kubiske. Previously, we have collaborated with FACE investigators Dr. Jud Isebrands and Dr. Evan McDonald. They have provided us with valuable tree physiological and field expertise, as well as data. The Aspen FACE website is <http://aspenface.mtu.edu>.

1.3 Preview

The models of leaf death, bud set, and shoot death presented in this thesis each rely on a fading memory weighted average component and a probabilistic component. In this paper, we analyze various fading memory weighted averages and their suitability for use in modeling growth processes within ECOPHYS. We present new ECOPHYS physiological process models and carbon distribution strategies. The process models include leaf death, bud set, shoot and branch death, and bud survival and vigor. These models replace earlier models within ECOPHYS in order to improve the accuracy and predictive capabilities of ECOPHYS. A new model of spring bud survival and vigor is derived as a function of the previous growing season's photosynthate productivity of parent leaves and shoots as well as over-winter storage. Also implemented are improvements to ECOPHYS carbon allocation including a new algorithm to combine adjacent internodes on older wood, a shoot by shoot carbon transportation shift that occurs at bud set, and a new model of carbon distribution within branches.

Chapter 2 - Process Models

There are many interdependent internal processes involved in tree growth. To simulate this, ECOPHYS incorporates various model components at the tree organ level, including leaf and branch birth, growth and death, carbon transportation, inter-leaf shading, and photosynthesis. This thesis considers a number of ECOPHYS process models including leaf death, bud set, shoot and branch death, and bud survival and vigor. Models for bud set due to stress, shoot death, and mature green-leaf death rely on the same type of fading memory weighted average mechanism and probabilistic component. Spring bud survival and vigor is modeled as a function of the previous growing season's photosynthate productivity of parent leaves and shoots as well as over-winter storage. The ECOPHYS shoot growth model has been altered to include a deterministic stage, which is a function of bud vigor and the distribution of over-winter storage, followed by an autonomous stage of indeterminate growth, followed by individual-shoot bud set occurring from early summer through fall.

2.1 Modeling Mature Green-Leaf Death

Shading and other environmental stresses can lead to the death of mature leaves that cannot maintain a favorable carbon balance (Dickson, et al. 1991). Several years ago, a model was developed and implemented within ECOPHYS to determine mature green-leaf death (Martin, 2001; Zang, 1999); that is, leaf death occurring prior to fall senescence. This model used a 10-day moving average of net photosynthate, $P_{net/area}(d)$, within the leaf to determine whether or not the leaf might die. $P_{net/area}(d)$ is calculated in Figure 2.1.

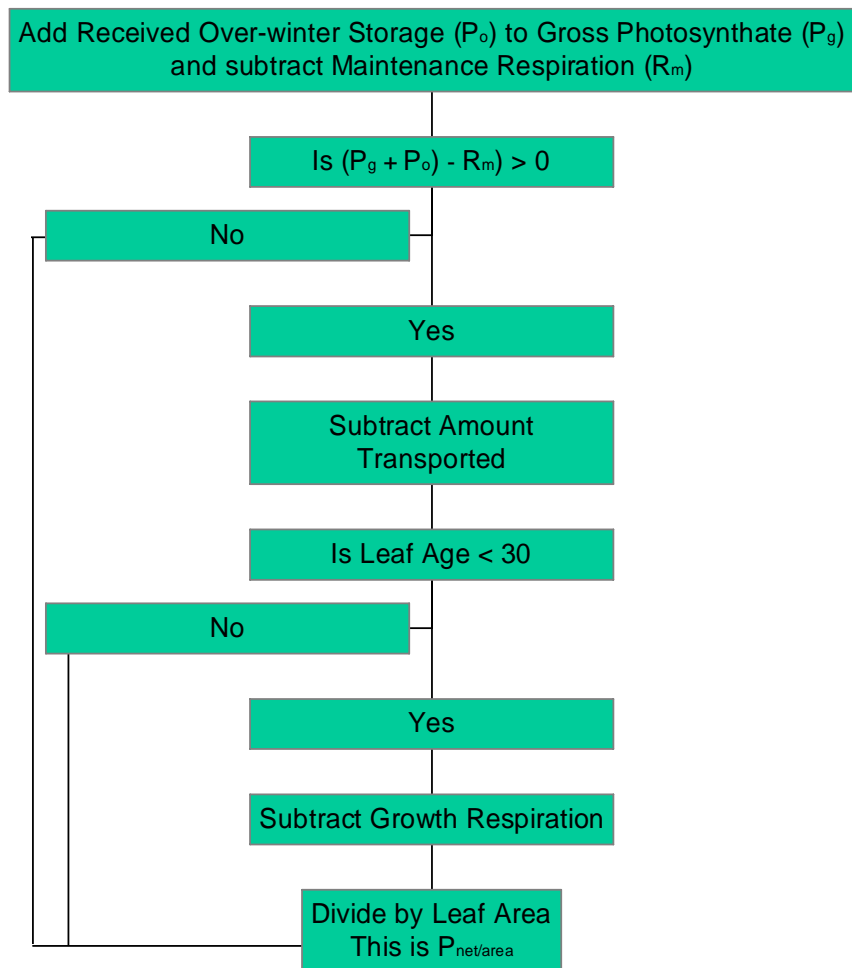


Figure 2.1 A flow chart showing how $P_{net/area}(d)$ is calculated

The equation for the rolling average is

$$Pa_{net/area}(d) = ((P_{net/area}(d) + P_{net/area}(d-1) + \dots + P_{net/area}(d-9))/10) \quad (2.1)$$

$Pa_{net/area}$ is the average of $P_{net/area}$ over the current and past nine days. In this model, if a leaf has $Pa_{net/area} < \tau$, where τ represents a threshold, then the leaf has a probabilistic chance of dying (Martin, 2001).

Specifically, one defines $K = (Pa_{net/area}(d) - \tau) * \delta$, where δ is a probability factor affecting how likely it is for the leaf to die. For each leaf we select ε to be a random number $0 \leq \varepsilon \leq 1$, and assume that the leaf remains if $\varepsilon > 1 - e^K$. We include such randomness to account for unmodeled dynamics as well as imprecision in our model. If $\varepsilon \leq 1 - e^K$, the leaf will die.

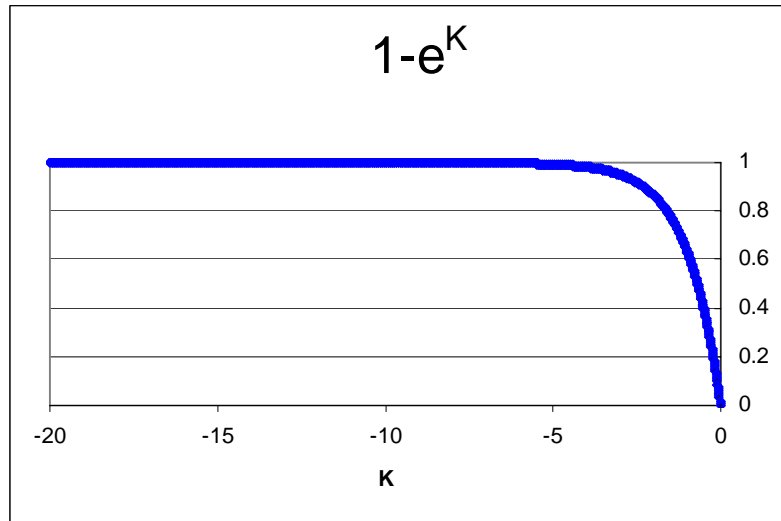


Figure 2.2 - A plot of K vs $1 - e^K$. The probability that a leaf dies when $Pa_{net/area}(d) < K$ is $1 - e^K$.

$P_{a_{n/a}}(d) - \tau$	δ	K	$1-e^K$
-0.05	5	-0.25	0.221199
-0.05	25	-1.25	0.713495
-0.1	5	-0.5	0.393469
-0.1	25	-2.5	0.917915

Table 2.1 - The effects of different ($P_{a_{net/area}} - \tau$) and δ on $1-e^K$

This model can be improved by replacing the straight 10-day average of $P_{net/area}$ with a fading memory weighted average. Computing the weighted average over about fifteen days is reasonable because after fifteen days of growth, a leaf has become autonomous with respect to carbon. Also, fifteen days is enough time for ozone damage to a leaf's photosynthetic rate to become significant. However, the model can be easily modified so that the weighted average is taken over a different number of days.

The alteration from a straight average to a fading memory weighted average makes the model less sensitive to how many days a leaf has to be doing poorly before it will die. In the straight 10-day average model, $P_{net/area}$ nine days ago is as important as $P_{net/area}$ of more recent days, while $P_{net/area}$ ten or more days ago has no effect. A fading memory weighted average allows $P_{net/area}$ values further in the past to have less impact on $P_{a_{net/area}}$ than recent $P_{net/area}$ values. In Chapter 3 we consider a variety of potential fading memory weighted averages which enables one to put a higher weight on the more recent $P_{net/area}$ of a leaf instead of equal weights on a selected number of them.

Specifically, we consider

$$Pa_{net/area}(d) = \sum_{t=-14}^0 w(t) * P_{net/area}(d+t) \quad (2.2)$$

where $w(t) = A * e^{\alpha * t}$ is a weighting function, $\sum_t w(t) = 1$. Here, α and A are chosen constants, resulting in a function with more weight on the current and recent days, and less weight on past days. See Section 3.6 for a numerical implementation of this. A similar averaging scheme can also be applied to the models of bud set and shoot death.

2.2 Bud Set due to Stress

Genetics largely determine how long a tree will grow its new shoots during a growing season and whether or not these shoots are going to be determinate, denova, or indeterminate with early bud set (Kubiske, personal communications),(Isebrands, personal communications). Even though the period of individual shoot growth is largely predetermined, growth is also affected by environmental factors. Adaptable species such as Aspen are able to respond relatively quickly to variations within the environment. There are many factors that can affect tree growth, such as water, ozone, shading, soil temperature, nutrients, light, air temperature, relative humidity, carbon dioxide and pests. ECOPHYS is currently responsive to environmental variability in shading, light, air temperature, relative humidity, carbon dioxide, and ozone levels. We currently assume that water and nutrient uptake are optimal and that there are no pests or viruses affecting tree growth. Pest interactions as well as water and nutrient limitations are expected to be added in the future.

In the past, ECOPHYS has assumed a fixed bud-set date at the end of each growing season. The user was required to enter in a bud-set date that determined when the denova shoots of the tree set their buds. The rest of the shoots set their buds based on their location within the tree. The further a shoot was along a second order branch, which is a branch that a shoot is connected to, the more likely it was to have a later bud-set date. The algorithm did not allow a shoot's bud set to respond to its environmental and growth conditions. We can observe in nature that the various trees within an aspen stand differ in their branch architecture and this is in part due to variations in shoot bud-set timing.

A new model for bud set due to stress can be constructed in a manner similar to the mature green-leaf death model of Section 2.1. If a branch is unable to meet its maintenance respiration needs while also continuing to produce new leaves, it will set its buds. This causes the shoot's photosynthate allocation pattern to shift away from new growth and towards shoot maintenance and carbon storage. Specifically, each day one can test branch net photosynthate relative to a threshold value τ_s . If larger than this threshold, tip growth is assumed to continue. If not, then there is a probabilistic chance of the branch setting its buds.

The algorithm is implemented as follows:

- 1) For each day d , find all long shoots that have not yet set their buds.
- 2) Record how much photosynthate is left on the shoot after carbon transportation and maintenance respiration have taken place.

We define

$$Pa_{net/mass}(d) = \sum_{t=-14}^0 w(t) * P_{net/mass}(d+t)$$

where $P_{net/mass}(d)$ is the net shoot's net photosynthate per unit shoot mass and $w(t)$ are appropriate weights.

3) Determine if the branch has enough photosynthate to support its maintenance needs by computing $Pa_{net/mass}(d) - \tau_s$.

4) If there is not enough photosynthate on the shoot to support its needs, then the shoot is assumed to set its buds using the approach taken in the leaf death algorithm.

Specifically, let $K = (Pa_{net/mass}(d) - \tau_s) * \delta_s$, where τ_s is a threshold and δ_s is a probability factor. For each shoot, we let ε be a random number, $0 \leq \varepsilon \leq 1$. The shoot does not set its buds if $\varepsilon > 1 - e^K$. If $\varepsilon \leq 1 - e^K$ the shoot will be assumed to set its buds.

5) If a shoot set its buds, it will then change its growing strategy and stop producing new leaves and internodes. It will continue to mature its juvenile leaves and internodes while shifting its carbon allocation strategy. We remark that there is still a static final bud-set date for the denova shoots.

Bud Set

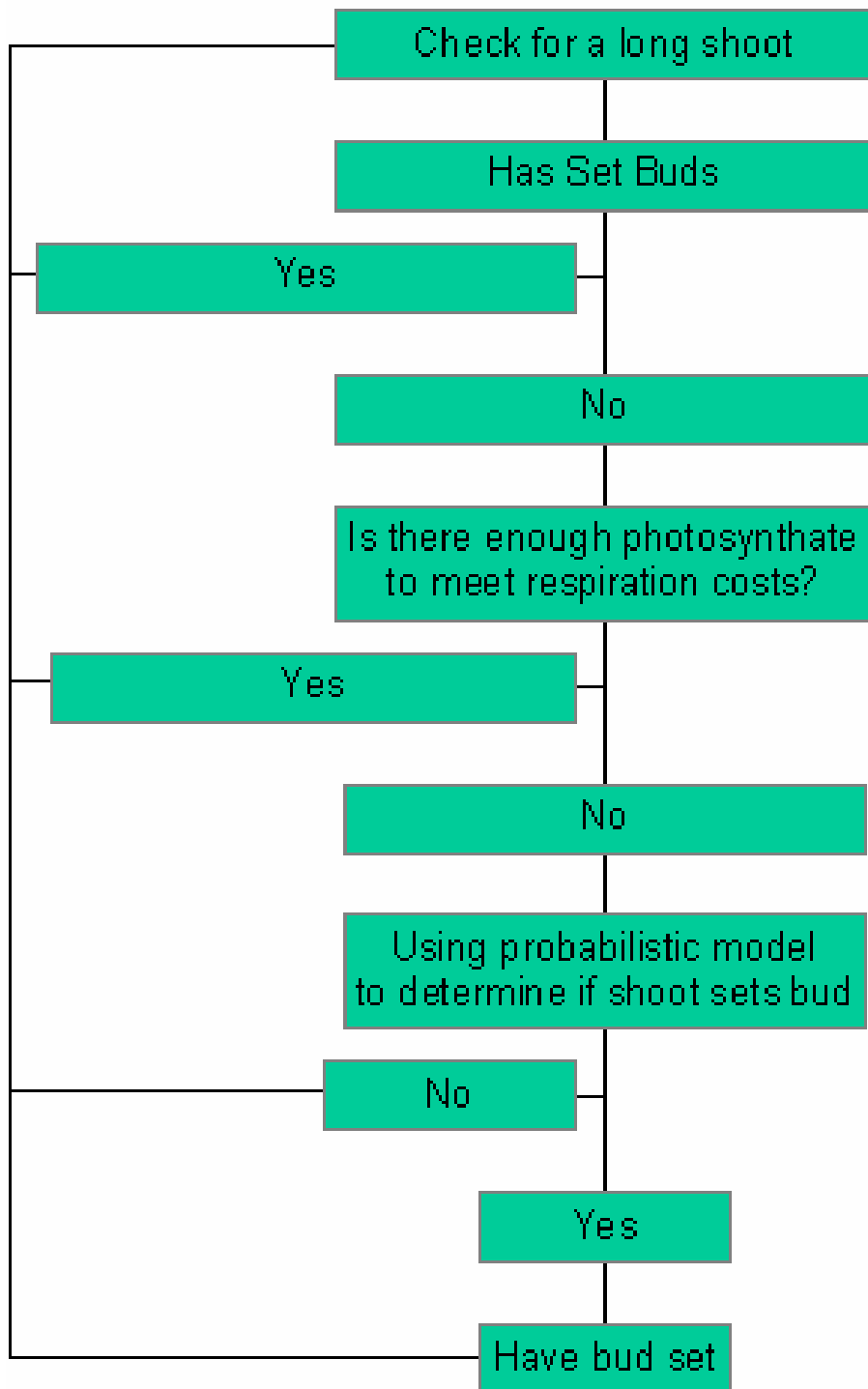


Figure 2.3 - A flow chart for the algorithm for shoot-level bud set due to stress.

2.3 Shoot and Branch Death

Without shoot and older wood death, ECOPHYS would have an unrealistically large number of branches after three or more years of growth. In addition to deviating greatly from real tree architectures, the resulting large number of leaves on the tree significantly slows the simulation. ECOPHYS previously had a recursive branch death algorithm (Wu, 1999). Starting in the third year of growth, this algorithm would kill branches on the tree at the end of each growing season based on branch location and random chance.

It is unrealistic to assume that productive shoots and branches die at random. This algorithm was not responsive to the tree's environment. According to (Dickson, et al. 1991), a branch that becomes stressed in the season, it is isolated from the nutrient flow and the branch dies. Also, we have observed that some branches will wither and die during the growing season. Because of this, we decided to create an algorithm based on shoot photosynthate productivity and check daily for shoot and older branch death.

A problem has developed when we initially implemented the new shoot death algorithm. It was observed that within ECOPHYS there was not enough photosynthate to support maintenance respiration in the shoots and branches. Previously, there was no mechanism that would cause a branch to die based on insufficient photosynthate productivity. Some branches that were not meeting their respiration needs remained alive and in many cases, continued to grow in length and diameter. This motivated changes in transportation (Chapter 4) and in maintenance respiration (Appendix B) in order to address this problem and allow the new shoot and branch death algorithms to be implemented.

2.3.1 Shoot-Death Algorithm

The new shoot-death algorithm comes into effect after a shoot has set its buds. This algorithm assumes that after the deterministic phase of growth, a shoot is autonomous with respect to carbon. (Dickson, 1991). Thus, a shoot must produce enough photosynthate to support itself. If a shoot does not generate enough photosynthate to support its own maintenance respiration, it will start to wither and die since it is incapable of receiving photosynthate from other parts of the tree. ¹⁴C studies have been performed to determine where leaves send their photosynthate throughout the tree (Sprugel 1991), (Isebrands, unpublished). Sprugel and colleagues found that under non-stressed conditions carbon transported out of a shoot was consistently sent downward to older wood, the trunk, and roots.

When a post bud-set shoot is not producing enough photosynthate to support its maintenance requirements over a period of time, we assume that the shoot will die following a probabilistic model of the type discussed previously.

Specifically, the algorithm for shoot death is as follows:

- 1) For each day d , search all of the shoots and find all long shoots that have set their buds.
- 2) For each of these shoots, record how much photosynthate is left on the shoot after transportation and respiration have taken place using the weighted average

$$Pa_{net/mass}(d) = \sum_{t=-14}^0 w(t) * P_{net/mass}(d+t)$$

where $P_{net/mass}(d)$ is the shoot's net photosynthate divided by shoot mass.

3) Determine if the shoot has enough photosynthate to support its maintenance needs by computing $P_{net/mass}(d) - \tau_s$.

4) If there is insufficient photosynthate on the shoot to support its needs, then the shoot dies using the same averages and probability mechanisms as used in the leaf death algorithm.

Specifically, let $K = (P_{net/mass}(d) - \tau_s) * \delta_s$, where τ_s is the threshold and δ_s is a probability factor. For the shoot, let ε be a random number $0 \leq \varepsilon \leq 1$. The shoot remains alive if $\varepsilon > 1 - e^K$. If $\varepsilon \leq 1 - e^K$, the shoot dies.

2.3.2 - Older Wood Branch-Death Algorithm

In ECOPHYS, if a shoot dies, it is removed from the tree. Each day after the dead shoots have been removed, we check for and remove any 2nd order branches that do not have any 1st order branches (shoots). Next, we check 3rd order branches similarly, and so on. This is illustrated in the following example.

Figure 2.4.1 shows three shoots that die on a given day (circled). After these shoots die, there are no branches currently growing on the circled 2nd order branch as shown in Figure 2.4.2. That branch dies, resulting in Figure 2.4.3.

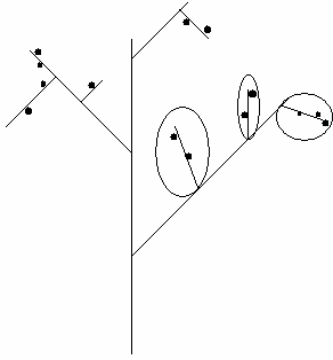


Figure 2.4.1

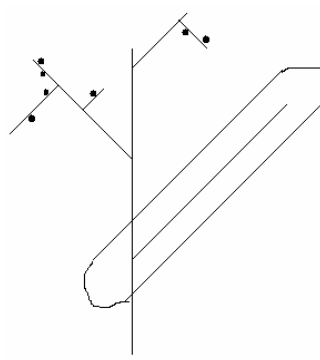


Figure 2.4.2

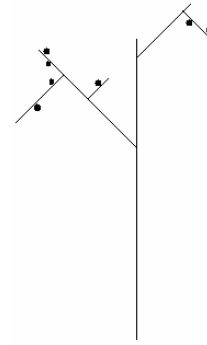


Figure 2.4.3

2.4 Bud Vigor and Death

After being dormant throughout the winter, aspen need energy for their buds to break and to provide shoot growth until maturing leaves become the primary sources for growth. This is the deterministic stage of growth. In the summer and fall, these shoots will start sending photosynthate down to storage in order to maintain over-winter storage to start growth the following spring (Dickmann, 2001). Over-winter stored carbon reserves are used for growth in the spring and to recover from damage that may take place while dormant (Dickmann, 2001). If there is not enough carbon available for a bud, the shoot will abort during initial development (Dickson, et al. 1991). ECOPHYS had a model in place to transport storage to the trunk, and then redistribute it to the new shoots in the spring (Zang, 1999).

In ECOPHYS each shoot contributes over-winter storage once it has set its buds. This continues until the end of the growing season and ends in the fall when all leaves have dropped. Even though trees send some of their photosynthate to the trunk of the tree, we

decided to modify ECOPHYS to store aboveground photosynthate at the branch level. The shoots that were doing well the previous season will now receive more photosynthate than the shoots that were not as productive.

Prior to our work in this area, all of the branches started contributing photosynthate to storage after the fixed bud-set date as explained in Section 2.2. After bud break at the start of the following season, the over-winter storage was distributed equally to all new shoots. The variation in productivity among shoots the previous season was ignored. In order to simulate unequal distribution of photosynthate among shoots, we have designed an algorithm that distributes over-winter storage to the emerging shoots based on their previous season's parent leaf productivity following the parent shoot's bud set. Thus, if a particular leaf produces relatively more photosynthate, then its associated shoot in the spring will receive relatively more photosynthate. The top bud on each terminal also receives an additional portion.

The bud vigor and death algorithm has the property that it gives buds "strength" based on how productive their parent leaves were (normalized by leaf area) after their branch's bud set the previous season. The top bud on each branch receives a percentage of the over-winter storage before the distribution to all buds to account for observed properties of hybrid poplars and aspen. The top bud also receives an additional amount during the distribution to all buds on the branch based on its parent leaf's productivity last season relative to the productivity of the other leaves on the branch.

This is the new algorithm implemented in ECOPHYS.

- 1) After bud set takes place on a given shoot, one measures how much net photosynthate per unit leaf area each leaf accumulates from bud set until fall senescence. Specifically, one computes

$$Total_leaf_psyn(LPI_T) = \sum_{i=Branch's\ Bud\ Set\ Date}^{Leaf\ Senescence\ Date} leaf_psyn(i) \quad (2.3)$$

- 2) One eliminates all potential buds corresponding to leaves for which the sum in (2.3) is zero or negative.
- 3) Next, each leaf for which (2.3) is positive, its $Total_leaf_psyn$ is normalized to the sum of all such leaf quantities on the shoot (including itself).

$$Leaf_percent(LPI_T) = \frac{Total_Leaf_psyn(LPI_T)}{\sum_{i=1}^n Total_leaf_psyn(i)} * 100 \quad (2.4)$$

- 4) A fixed percentage of over-winter storage is allocated to the top buds on each branch. This amount is then subtracted from the total over-winter storage within the branch.
- 5) The remaining stored carbon is distributed proportionate to the amount generated from its leaf after bud set in (2.4).

$$\begin{aligned} \text{Bud's over-winter storage} &= \text{Leaf_percent(LPI_T)} * (\text{Branch's Overwinter Storage} \\ &- \text{Amount allocated to top internode}) \end{aligned} \quad (2.5)$$

- 6) For each bud, compare the amount with a predetermined threshold amount under which buds die. Kill buds accordingly.

2.5 Simulating Genetically - Determined Bud Set

As mentioned before, genetics play an important role in determining the duration of shoot growth. To represent this, we developed an empirical model of bud set date based on the productivity of the leaves following bud set, and on over-winter storage as measured in the algorithm based on the leader's productivity after bud set.

The longest date that a shoot, say shoot k , can grow is based on the following equation.

$$\text{budset date}(k) = \text{budbreak date} + (\text{bud strength}(k) / \text{leaders bud strength}) * (\text{leader's budset date} - \text{budbreak date}). \quad (2.6)$$

That is, the bud-set date for shoot k is scaled proportionately according to bud strength, to the leader's bud-set date. Bud strength is an abstract term that we use to refer to a bud's general potential for early-season growth.

This date is an upper bound on the last day that shoot k will set bud in this season. It will set bud on this date unless it has already set bud due to insufficient photosynthate as described in Section 2.2.

Chapter 3 - Types of Weighted Averages

Throughout the growing season, trembling aspen and hybrid poplar initiate new leaves. As a tree grows, some leaves, internodes, and branches become stressed due to a variety of factors, including shading from other leaves, elevated ozone, pests, viruses, lack of nutrients and water. Some of these leaves, internodes and branches remain small or die during the summer. A leaf or branch is usually under stress for a period of time before it dies. In this thesis, we assume that stress effects at leaf or branch levels can be measured in terms of net photosynthate available to the leaf or branch. The current level of photosynthate available to the leaf or branch is important, but the photosynthate history is also important. Thus, we model mature green-leaf death, bud set due to stress and shoot death with fading memory weighted averages of leaf or shoot net photosynthate using the method discussed in Chapter 2.

3.1. Weighted Averages in ECOPHYS

ECOPHYS uses fading memory weighted averages of net photosynthate productivity to determine whether or not a leaf or shoot dies. We chose to implement an exponential decay function as the weighting function. However, there are other types of weighting functions that one can consider. This chapter focuses primarily on the following three types of weighting functions: linear, exponential decay, and an exponential moving average. Also, a brief overview is given of other weighting function types that could be used if there were biological evidence to support them. The analysis of Sections 3.4 establishes that the formula for an exponential moving average (Appendix C) gives the same result as the exponential decay weighted fading memory average in the theoretical

case where there is an infinite history of data values. The analysis of Sections 3.5 and 3.6 establish how the formulas compare when there is a finite history of data values, which is the case for ECOPHYS processes.

3.2 Definition of a weighted average

A **finite weighted average** over n data values $\{V(1), \dots, V(n)\}$ is of the form

$$Y = \sum_{i=1}^n w(i) * V(i) \quad (3.1)$$

where the $w(i)$ satisfy Definition 3.1. Similarly, an **infinite weighted average** over an infinite sequence of data points $\{V(1), V(2), \dots\}$ is of the form

$$Y = \sum_{i=1}^{\infty} w(i) * V(i) \quad (3.2)$$

where the $w(i)$ satisfy Definition 3.1.

Definition 3.1 The **weights** $\{w(1), w(2), \dots\}$ in a **weighted average** are such that:

i) $\sum_i w(i) = 1$

ii) $w(i) > 0 \quad \forall i$

For a **fading memory weighted average**, additionally,

iii) $w(i) < w(i+1) \quad \forall i$

The general form of the rolling fading memory weighted averages in ECOPHYS process models described in Chapter 2 is

$$Y(d) = \sum_{t=-n+1}^0 w(t) * V(d+t) \quad (3.3)$$

In the mature green-leaf death model $Y(d)$ is $P_{\text{net/area}}(d)$ and $V(d+t)$ is $P_{\text{net/area}}(d+t)$. In the bud set due to stress and the shoot death models, $Y(d)$ is $P_{\text{net/mass}}(d)$ and $V(d+t)$ is $P_{\text{net/mass}}(d+t)$.

3.3 Types of Weighted Averages

3.3.1 Linearly Weighted Averages

The most basic option for a fading memory weighted average over the current and previous $n-1$ days would have a linear weighting function

$$\begin{aligned} w(t) &= m * t + b \quad \text{for } t \geq (-n+1), m > 0 \text{ and } b > 0 \\ \text{and } w(t) &= 0 \quad \text{for } t \leq -n \end{aligned} \quad (3.4)$$

Note that if $m \leq 0$ or if $b \leq 0$ then Definition 3.1 is not satisfied. In order to have positive weight on the current day and the previous $n - 1$ days, we include the condition

$w(-n) = 0$. Using this and (i) from Definition 3.1, we have the system of equations,

$$\frac{n * (1-n) * m}{2} + n * b = 1 \quad \text{and } -n * m + b = 0, \quad (3.5)$$

with two unknowns, m and b (since n is already chosen). Note that with m positive and $w(-n) = 0$, then $w(t) > 0$ for $t > -n$, satisfying (ii) in Definition 3.1.

Solving (3.5), gives

$$w(t) = \frac{2}{n * (1+n)} * t + \frac{2}{(1+n)} = \frac{2 * (n+t)}{n * (1+n)} \quad (3.6)$$

and the weighted average formula

$$Y(d) = \sum_{t=-n+1}^0 \frac{2 * (n+t)}{n * (1+n)} * V(d+t) \quad (3.7)$$

Example 3.1: Let $n = 15$. Then

$$w(t) = \frac{1}{120} * t + \frac{1}{8} \tag{3.8}$$

where $t \in \{-14, \dots, 0\}$.

A plot of $w(t)$ when $n = 15$ is shown in Figure 3.1

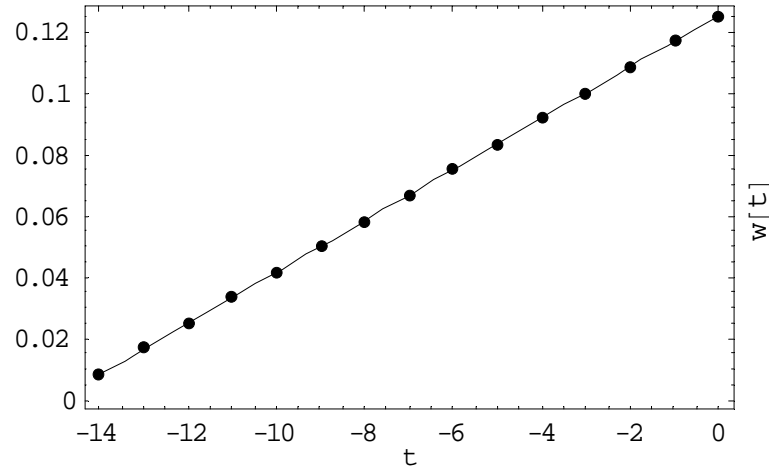


Figure 3.1 - The weights $w(t)$ of a fading linear weighted average when $n = 15$

This is a fading memory weighted average according to Definition 3.1. This is a simple kind of fading memory weighted average, but it does require that n data values, $V(d+t)$, be stored for each leaf or shoot each day during the simulated growing season.

3.3.2 Exponential Decay Function Weights

Consider a weighted average where the weighting function is an exponentially decaying function in negative time of the form

$$w(t) = A * e^{\alpha * t}, \tag{3.9}$$

where A and α are positive constants and α determines the rate at which the function decays as t decreases, $t \in \{0, -1, -2, \dots\}$.

Observe that the weight on the current day, $t = 0$, will be equal to A , and $\lim_{t \rightarrow -\infty} w(t) = 0$.

In general, for $w(t)$ in (3.9), an infinite weighted average is of the form

$$F(d) = \sum_{t=-\infty}^0 A * e^{\alpha * t} * V(d + t) \tag{3.10}$$

where $\sum_{t=-\infty}^0 A * e^{\alpha * t} = 1$.

An n -day weighted average is of the form

$$F_n(d) = \sum_{t=-n+1}^0 A_n * e^{\alpha * t} * V(d + t) \tag{3.11}$$

where,

$$\sum_{t=-n+1}^0 A_n * e^{\alpha * t} = 1 \tag{3.12}$$

For the applications in Chapter 2, $t = 0$ refers to the current day, $t = -1$ is the previous day, and so forth. Once the parameter $\alpha > 0$ is chosen, one solves for A_n satisfying (3.12).

Let $n = 15$. Figure 3.2 shows $w(t)$ when $\alpha = .2$. and $A_{15} = .190767$.

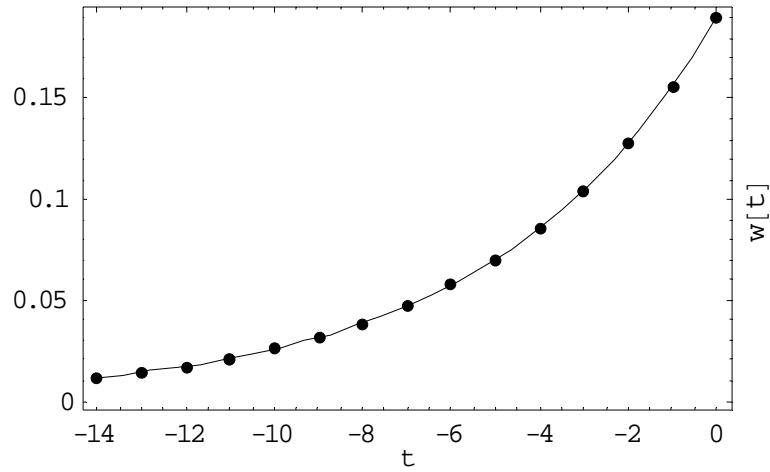


Figure 3.2 Graph of $w(t) = A_{15} * e^{\alpha * t}$ for $\alpha = .2$ and $A_{15} = .190767$

3.3.3 A Reduced Range for α

Recall that $\alpha > 0$ must be chosen. For the applications in Chapter 2, it is assumed that significant weight be given to data values $V(d+t)$ for at least ten days in the past. For any negative value of t , $A_n * e^{\alpha * t}$ decreases when α increases. When α is large, the decay of $A_n * e^{\alpha * t}$ in negative t is fast, putting almost all of the weight on the data values $V(d+t)$ for t near zero. As shown in Figure 3.3, for $\alpha = 5$, almost all of the weight is on the current days' data value. In fact, $w(-1) = .00669255$.

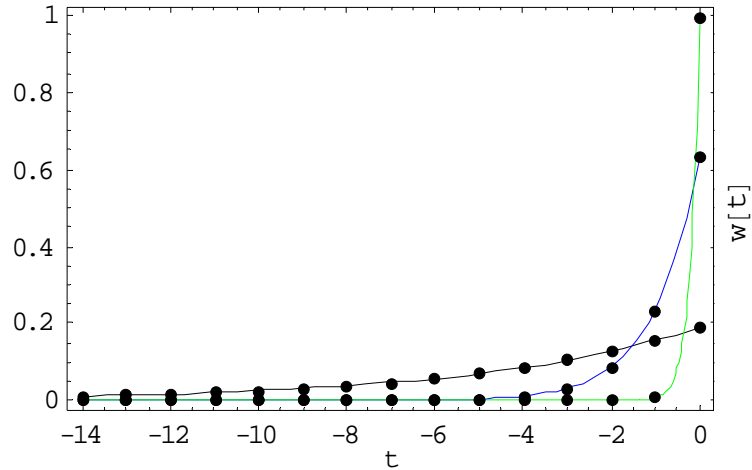


Figure 3.3 – This is a graph showing the effects of α on $w(t) = A_n * e^{\alpha*t}$ for $\alpha=0.2$ (slowest rate of decay), $\alpha=1$, and $\alpha=5$ (fastest rate of decay) .

As shown in Figure 3.3, when $\alpha = 0.2$ the data up to about 13 - 15 days into the past will be significant to the average, with $w(-13) = 0.0134635$, $w(-14) = 0.0116006$, and $w(-15) = 0.00902486$. When $\alpha = 1$, $w(-5) = 0.0042592$ which may be negligible depending on the range of the magnitude for the data. This results in essentially a 5-day weighted average based on the current and the previous four days of data. In order for at least ten days to be significant to the weighted averages of the applications in Chapter 2, one can limit the range of α to $0 < \alpha < 1$.

Recall from Chapter 2 that the general formula for the exponential decay weighted average in the mature green-leaf death algorithm is

$$P_{a_{net/area}}(d) = \sum_{t=-n+1}^0 A_n * e^{\alpha*t} P_{net/area}(d+t) \quad (3.13)$$

For a 15-day average in (3.13), the weights are reasonable when $\alpha = .2$ and $A_n = .190767$, with $w(-14) > .01$ and $w(-15) < .01$.

Figure 3.4 illustrates the effects of selecting different weighting functions. Shown are straight, linear, and exponential decay weighted rolling averages for a data set generated from an ECOPHYS test run using FACE data. The fifth leaf initiated was measured from Julian Days 159 - 193. Note that 14 data points prior to day 173 are not shown in the figure.

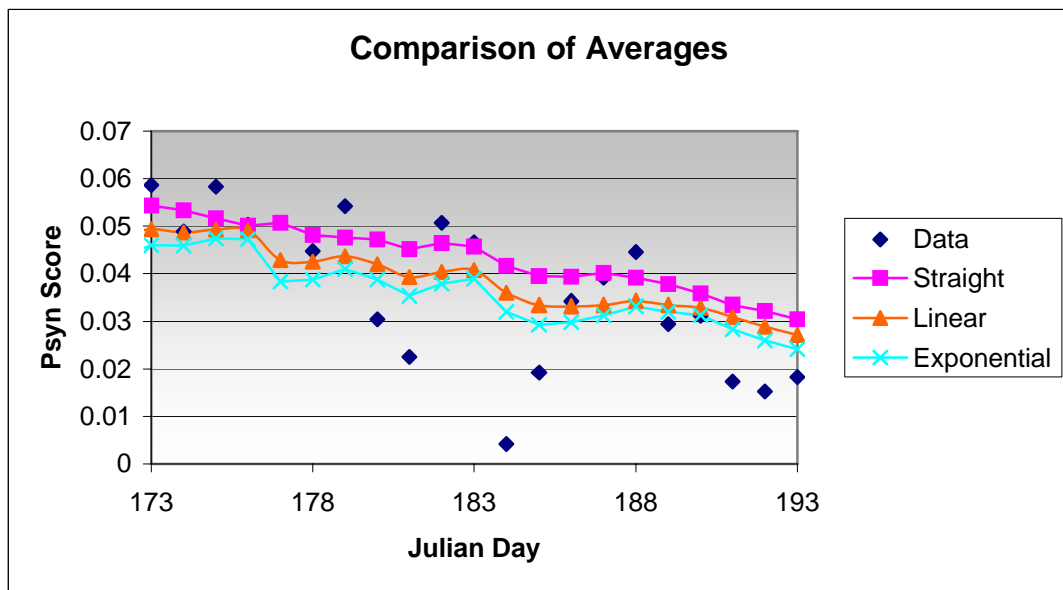


Figure 3.4 - A comparison of the straight, linear and exponential weighted averages when $n=15$. Note that averages on days 173 - 187 include data points not shown.

3.3.4 – Other Types of Weighted Averages

There are other types of fading memory weighted averages that we could use besides the linear or simple exponential decay types. For example,

$$F_1(d) = \sum_{t=-\infty}^0 A_1 * t * e^{\alpha * t} * V(d+t) = \sum_{t=-\infty}^0 -e^{-\alpha} * (e^{\alpha} - 1)^2 * t * e^{\alpha * t} * V(d+t), \quad (3.14)$$

$$F_2(d) = \sum_{t=-\infty}^0 A_2 * t^2 * e^{\alpha * t} * V(d+t) = \sum_{t=-\infty}^0 \frac{e^{-\alpha} * (e^{\alpha} - 1)^3}{e^{\alpha} + 1} * t^2 * e^{\alpha * t} * V(d+t), \quad (3.15)$$

$$F_3(d) = \sum_{t=-\infty}^0 A_3 * t^n * e^{\alpha * t} * V(d+t), \quad (3.16)$$

or finite rolling weighted averages of these types. The weighted averages have more weight on recent past data values than on the current day's datum.

For each average, Mathematica (Mathematica, 2003) was used to solve for A as a function of α so that Condition (i) of Definition 3.1 was satisfied. The weighting functions for these averages do not satisfy the fading memory requirement given by (iii) of Definition 3.1. However, these averages have fading memory in the sense that leftward from a peak the weighting function tends monotonically towards zero, as illustrated below in Figures 3.5, 3.6, and 3.7. For these graphs, $\alpha = 0.2$.

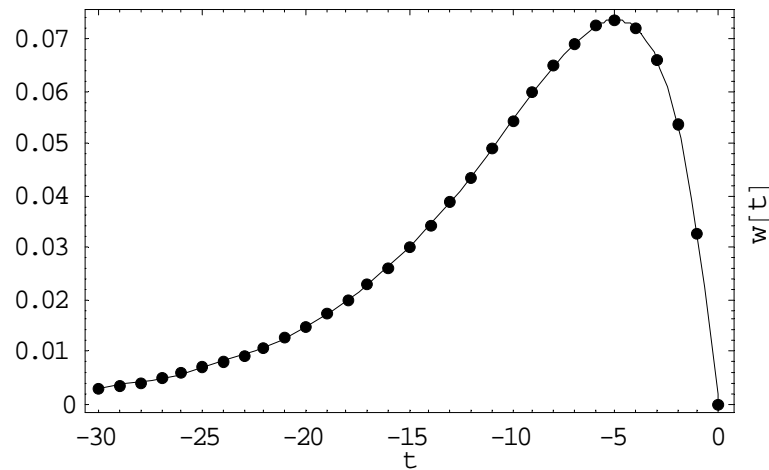


Figure 3.5: $F_1(d) = \sum_{t=-\infty}^0 A_1 * t * e^{\alpha * t} * V(d+t)$, $\alpha = .2$, $A_1 = -0.0401335$

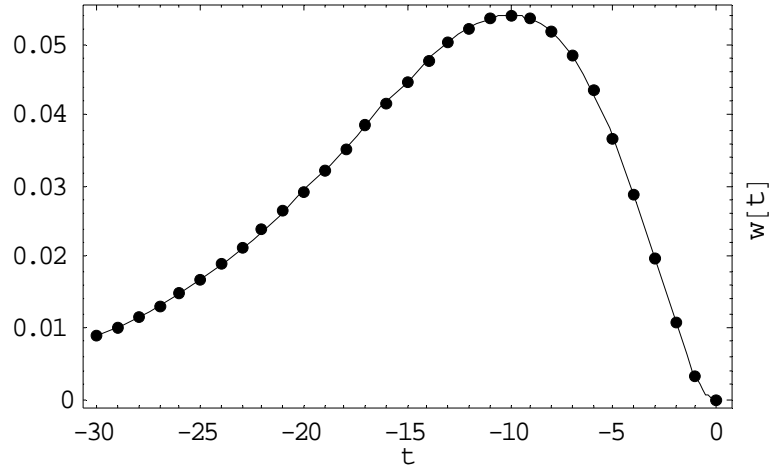


Figure 3.6: $F_2(d) = \sum_{t=-\infty}^0 A_2 * t^2 * e^{\alpha t} * V(d+t)$, $\alpha = .2$, $A_2 = 0.00400003$

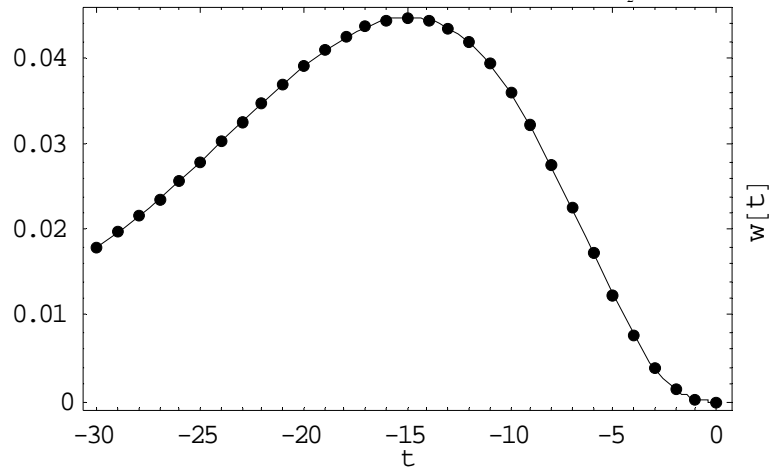


Figure 3.7: $F_3(d) = \sum_{t=-\infty}^0 A_3 * t^3 * e^{\alpha t} * V(d+t)$, $\alpha = .2$, $A_3 = -0.000266666$

These functions each would be appropriate for tree physiology or forest modeling applications where there is a lag in the response of one variable to another.

One could also construct a weighted average with the weighting function being a sum of k exponential decay functions.

$$F(d) = \sum_{t=-\infty}^0 \sum_{i=1}^k A(i) * e^{\alpha(i)*t} * V(d+t) \tag{3.17}$$

or a polynomial, P(t), times an exponential decay function

$$F(d) = \sum_{t=-\infty}^0 P(t) * e^{\alpha * t} * V(d + t) \tag{3.18}$$

where k is a positive constant. These weighting functions are not pursued further here.

One could also use any of the averages discussed in this thesis with a delay of c days.

That is,

$$F(d) = \sum_{t=-\infty}^0 w(t) * V((d - c) + t) \tag{3.19}$$

3.4 – The Relationship Between the Fading Memory Exponential Decay Weighted Average and the Exponential Moving Average

In Subsection 3.4.1, we describe the exponential moving average. In Subsection 3.4.2 we show that the exponential moving average and the exponential decay weighted average would be the same if one used an infinite number of data points. In Subsection 3.4.3, we analyze the weighted average over a finite number of days based on our findings in Subsection 3.4.2.

3.4.1 Exponential Moving Average

Another option we explore for modeling processes such as leaf and shoot death in Chapter 2 is to use an exponential moving average in place of a fading memory weighted average. The exponential moving average has been proposed for modeling the movement of stocks on the stock exchange at the website at Paritech (2005). In that application, an exponential moving average, $G(d)$, is calculated by applying a percentage of today's

stock value, $V(d)$, to yesterday's moving average $G(d-1)$ (Appendix C). The formula for an exponential moving average is

$$G(d) = (1 - \beta) * G(d - 1) + \beta * V(d), \text{ where } 0 < \beta < 1, \quad (3.20)$$

where $G(d)$ is today's current average, $V(d)$ is the current day's value, and β is the weight on the current day's value.

For example, one could put 30% of the weight on the current data point $V(d)$, and then 70% on the previous average, $G(d-1)$. The weighted average would then be

$$G(d) = (1 - 0.3) * G(d - 1) + 0.3 * V(d) \quad (3.21)$$

Observe that in this model one must provide an infinite time history or a starting point, $G(0)$.

This average is sometimes used as a tool for stock exchange modeling to predict whether a stock is going to rise or fall (Appendix C). However, in ECOPHYS, the application would be different. For example, one might look at the trend of a leaf's net photosynthate production to determine if the leaf is going to die or not on the current day. Thus, we are not predicting a future trend as in the stock application, but are instead evaluating a current status. The connection between the stock exchange model and ECOPHYS's leaf death model is through the use of a quantity's history, with current day and recent day values more important than distant past values.

3.4.2 Relating the Exponential Moving Average to the Exponential Decay Weighted Average

We will show that $F(d)$ of (3.10) satisfies

$$\Delta F(d) = F(d) - F(d-1) = A * (V(d) - F(d-1)) \quad (3.22)$$

In order to compare this with the exponential moving average, $G(d)$, we also establish that

$$\Delta G(d) = G(d) - G(d-1) = \beta * (V(d) - G(d-1)) \quad (3.23)$$

This shows that the rate of change in these two averages is proportionate. We first show that the exponential decay weighted average $F(d)$ satisfies (3.22) Then we show that the exponential moving average $G(d)$ satisfies (3.23).

Consider the exponential decay weighted average function

$$F(d) = \sum_{t=-\infty}^0 A * e^{\alpha * t} * V(d+t) \quad (3.24)$$

where d is the current day and each $V(d+t)$ is a data point. For example, $V(d+t)$ is the net photosynthate per unit leaf area, as used in the leaf death algorithm.

The previous day's weighted average, $F(d-1)$, is

$$F(d-1) = \sum_{u=-\infty}^0 A * e^{\alpha * u} * V((d-1)+u) \quad (3.25)$$

Substitute $t = u-1$ in (3.25). We have

$$F(d-1) = e^{\alpha} * \sum_{t=-\infty}^{-1} A * e^{\alpha * t} * V(d+t) \quad (3.26)$$

We now have

$$\Delta F(d) = F(d) - F(d-1) = \sum_{t=-\infty}^0 A * e^{\alpha * t} * V(d+t) - e^{\alpha} * \sum_{t=-\infty}^{-1} A * e^{\alpha * t} * V(d+t) \quad (3.27)$$

Thus,

$$\Delta F(d) = A * V(d) + \sum_{t=-\infty}^{-1} A * e^{\alpha * t} * V(d+t) - e^{\alpha} * \sum_{t=-\infty}^{-1} A * e^{\alpha * t} * V(d+t) \quad (3.28)$$

Simplifying,

$$\Delta F(d) = A * V(d) + (1 - e^{\alpha}) * \sum_{t=-\infty}^{-1} A * e^{\alpha * t} * V(d+t) \quad (3.29)$$

Note that from (3.26), we have $\sum_{t=-\infty}^{-1} A * e^{\alpha * t} * V(d+t) = \frac{F(d-1)}{e^{\alpha}}$. Therefore,

$$\Delta F(d) = A * V(d) + \frac{(1 - e^{\alpha}) * F(d-1)}{e^{\alpha}} \quad (3.30)$$

In order to normalize the weights we must have

$$\sum_{t=-\infty}^0 A * e^{\alpha * t} = 1 \quad (3.31)$$

The sum $\sum_{t=-\infty}^0 A * e^{\alpha * t}$ can be rewritten as $\sum_{t=0}^{\infty} A * (e^{-\alpha})^t$. This is a geometric series in $e^{-\alpha}$,

and so

$$\sum_{t=0}^{\infty} A * (e^{-\alpha})^t = \frac{A}{1 - e^{-\alpha}} = 1 \quad (3.32)$$

Thus,

$$A = 1 - e^{-\alpha} = \frac{(e^{\alpha} - 1)}{e^{\alpha}} \quad (3.33)$$

and so

$$\frac{(1-e^{-\alpha})}{e^{-\alpha}} = -A. \quad (3.34)$$

Now we substitute (3.34) into (3.30) to get,

$$\Delta F(d) = A * V(d) - A * F(d-1) = A * (V(d) - F(d-1)) \quad (3.35)$$

This completes the first part of the proof.

Next consider the exponential moving average

$$G(d) = (1-\beta)*G(d-1) + \beta*V(d) \quad (3.36)$$

From (3.36), we have

$$\Delta G(d) = G(d) - G(d-1) = (1-\beta)*G(d-1) + \beta*V(d) - G(d-1) \quad (3.37)$$

Thus,

$$\Delta G(d) = G(d-1) - \beta*G(d-1) + \beta*V(d) - G(d-1) \quad (3.38)$$

$$\Delta G(d) = \beta*(V(d) - G(d-1)) \quad (3.39)$$

Recall that the range for β is (0,1). This is also the range for $A = 1 - e^{-\alpha}$. Comparing

(3.35) and (3.39), we see that for $\beta = 1 - e^{-\alpha}$, $F(d)$ and $G(d)$ have the same rate of change

on any given day. Thus, $G(d) = F(d)$ when we have an infinite data history and

$$\beta = 1 - e^{-\alpha}.$$

3.4.3 Initial Conditions

For the applications of Chapter 2, we use weighted averages for a finite time history.

There is always an initiation date for each of the models after $V(d)$ values begin

accumulating. If we assume that $V(0) = 0$ and $V(1)$ is the first data value, we can think of

our photosynthate values being zero for all $d = (-1, -2, \dots)$. That is, for any given $d \geq 0$, $V(d+t) = 0$ for all $t < -d$. This allows us to write a finite average as an infinite average.

Thus,

$$F(d) = \sum_{t=-\infty}^0 A * e^{\alpha * t} * V(d+t) = \sum_{t=-\infty}^{-d} A * e^{\alpha * t} * V(d+t) + \sum_{t=-d+1}^0 A * e^{\alpha * t} * V(d+t) \quad (3.41)$$

where $\sum_{t=-\infty}^{-d} A * e^{\alpha * t} * V(d+t)$ is the tail, equal to zero. This allows us to compare a finite

weighted average with other weighted averages that possibly have infinitely many non-

zero values. We will show in Section 3.5 that if $V(d+t)$ is bounded and if the tail of an

infinite average, $\sum_{t=-\infty}^{-n} A * e^{\alpha * t} * V(d+t)$, has sufficiently small weights, then the infinite

average will be very close to the finite average, $\sum_{t=-n+1}^0 A * e^{\alpha * t} * V(d+t)$.

3.5 – Choosing α

The weighted average $F(d) = \sum_{t=-\infty}^0 w(t) * V(d+t)$ includes data from an infinite number of

past days, but for t far enough back in history, $w(t)$ becomes so small that one can

consider the weighted data insignificant to the weighted average. This is because we

assume that the data values $V(d+t)$ are bounded in magnitude by some number “M”. We

choose α so that $w(t)$ is sufficiently small for $t < -n+1$, where n is the number of days of

data that we want to be significant to the average. We have

$$A * \sum_{t=-\infty}^0 e^{\alpha * t} = A * \sum_{t=-\infty}^{-n} e^{\alpha * t} + A * \sum_{i=-n+1}^0 e^{\alpha * t} \quad (3.42)$$

Let ε be a level of tolerance, chosen so that if

$$A * \sum_{t=-\infty}^{-n} e^{\alpha * t} \leq \varepsilon, \quad (3.43)$$

we can consider the contribution of data n days or more into the past to be insignificant.

From (3.33), recall that $A = 1 - e^{-\alpha}$, so then equation (3.43) becomes

$$(1 - e^{-\alpha}) * \sum_{t=-\infty}^{-n} e^{\alpha * t} \leq \varepsilon. \quad (3.44)$$

Substituting $u = -(t + n)$,

$$(1 - e^{-\alpha}) * e^{-\alpha * n} \sum_{u=0}^{\infty} e^{-\alpha * u} = e^{-\alpha * n} \quad (3.45)$$

With this substitution we have

$$e^{-\alpha * n} \leq \varepsilon \quad (3.46)$$

$$\ln(e^{-\alpha * n}) \leq \ln(\varepsilon)$$

$$-\alpha * n \leq \ln(\varepsilon)$$

$$\alpha \geq \frac{-\ln(\varepsilon)}{n} \quad (3.47)$$

If one chooses $\alpha = \frac{-\ln(\varepsilon)}{n}$, then $w(-n+1)$ will have the largest weight possible while meeting the tolerance condition.

So given, $n \in \mathbb{N}$ and $0 < \varepsilon < 1$, select

$$\alpha = \frac{-\ln(\varepsilon)}{n}. \quad (3.48)$$

For $n = 15$ and tolerance $\varepsilon = .05$, Figure 3.8 shows the cutoff for the tail. This is a 15-day average, where $\alpha = 0.199715$ and $A = 0.181036$.

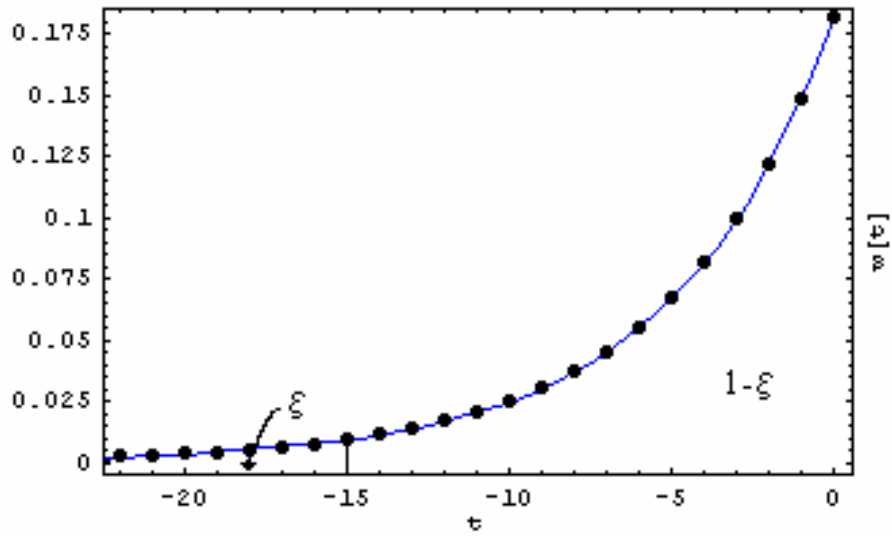


Figure 3.8 - A 15-day average showing that the first 15 days have a significance of $1 - \varepsilon$

In Figure 3.8, the tail will include everything previous to the 14th day back in history, while the current day and first 14 days in history will contribute significantly.

As illustrated in Figure 3.9, if a higher tolerance is chosen, then α will approach zero

since $\lim_{\varepsilon \rightarrow 1} \frac{-\ln(\varepsilon)}{n} = 0$. With a very small tolerance, α will be large.

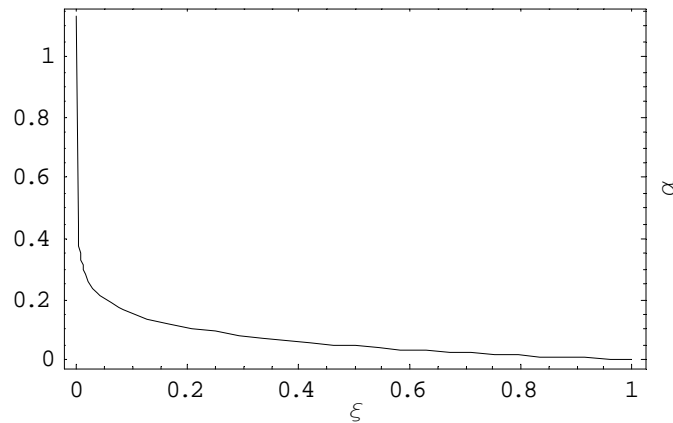


Figure 3.9 - As ε approaches 1, α approaches 0.

We also observe the effects on α of the number of days that we want to be significant. As n approaches infinity, for fixed tolerance ϵ , α will approach zero. However, if n is small, α will be larger.

For example, let $\epsilon = .05$ and let n vary. Figure 3.10 illustrates that as n increases or when ϵ increases, α decreases.

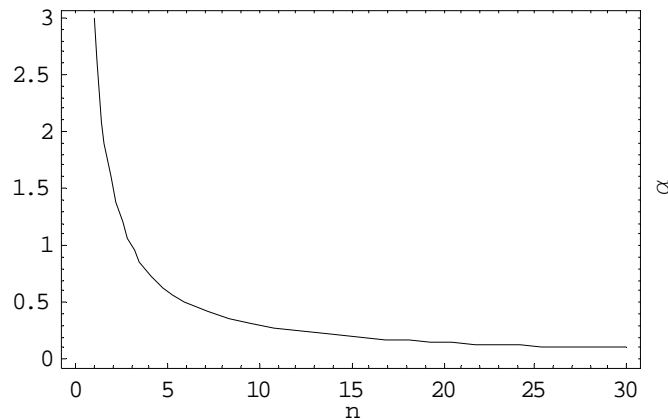


Figure 3.10 - As n approaches infinity, α approaches zero

3.6 – The Exponential Moving Average is Interchangeable With the Rolling Exponential Weighted Average

In Section 3.5 we showed that for an infinite average the exponential decay weighted average and the exponential moving average were in fact the same. Even though this is not true in the finite average case, we have decided to implement the moving average formulation into ECOPHYS so that only one data value and one average value have to be stored for each leaf instead of storing n data values in the weighted average for each leaf.

Since we do not have an infinite number of historic data points we have to start with an initial value. We start with $G_F(0) = V(0) = 0$, when the leaf is initiated, and build the average from there as follows:

$$G_F(d) = (1 - A) * G_F(d - 1) + A * V(d), \quad (3.49)$$

where $A = 1 - e^{-\alpha}$.

Then, beginning with the n th day, we assess daily whether or not a leaf dies.

This is not equivalent to the n -day rolling exponential decay fading memory weighted average that we were previously using. There are two differences between these formulas. First,

$$F_n(d) = \sum_{t=-n+1}^0 A_n * e^{\alpha * t} * V(d + t) \quad (3.50)$$

is a rolling average while $G_F(d)$ is not. After n days, $F_n(d)$ will not include $V(d - n)$ or any data values further in the past. In contrast, $G_F(d)$ will include all of the past data values.

These two functions also differ in that $A_n \neq A$. Recall that we choose A_n so that

$$\sum_{t=-n+1}^0 A_n * e^{\alpha * t} = 1.$$

In fact, $A_n > A$.

The more days that go by, the less significant $G(0)$ is to the average. We show below that having $G(0) = V(0) = 0$ for (3.49) is equivalent to having the tail of an infinite average be zero, as we have for a finite approximation discussed in Section 3.5. Suppose that we have data values $\{V(1), V(2), \dots\}$. For any given day, " d ", consider the finite average

$$H(d) = \sum_{t=-d+1}^0 A * e^{\alpha * t} * V(d+t). \quad (3.51)$$

As d increases, V(0) goes further into the past and $e^{-\alpha * d} * V(0)$ tends towards zero. That is, "-n" becomes more negative in equation (3.41). Thus, as d increases, H(d) - F(d) tends towards zero.

We show below that $H(d) = G_F(d)$ given that $H(0) = G_F(0)$. The steps are much the same as those in Subsection 3.4.2 establishing $F(d) = G(d)$.

As in equation (3.27)

$$\Delta H(d) = H(d) - H(d-1) = \sum_{t=-d+1}^0 A * e^{\alpha * t} * V(d+t) - e^{\alpha} * \sum_{t=-d+1}^{-1} A * e^{\alpha * t} * V(d+t)$$

$$\Delta H(d) = A * V(d) + \sum_{t=-d+1}^{-1} A * e^{\alpha * t} * V(d+t) - e^{\alpha} * \sum_{t=-d+1}^{-1} A * e^{\alpha * t} * V(d+t)$$

$$\Delta H(d) = A * V(d) + (1 - e^{\alpha}) * \sum_{t=-d+1}^{-1} A * e^{\alpha * t} * V(d+t)$$

$$\Delta H(d) = A * V(d) + (1 - e^{\alpha}) * \frac{F(d-1)}{e^{\alpha}}$$

$$\Delta H(d) = A * V(d) - A * F(d-1) = A * (V(d) - F(d-1)).$$

Now

$$G_F(d) = (1-A) * G_F(d-1) + A * V(d)$$

$$\Delta G_F(d) = G_F(d) - G_F(d-1) = (1-A) * G_F(d-1) + A * V(d) - G_F(d-1)$$

$$\Delta G_F(d) = G_F(d-1) - A * G_F(d-1) + A * V(d) - G_F(d-1)$$

$$\Delta G_F(d) = A * (V(d) - G_F(d-1))$$

Since $\Delta H(d) = \Delta G_F(d)$ and $H(0) = G_F(0) = 0$, $H(d) = G_F(d)$ for all $d = \{1, 2, \dots\}$

The models proposed in Chapter 2 use a rolling weighted average $F_n(d)$. Below we establish that $|F_n(d) - G_F(d)| \leq 2 * \varepsilon * M$.

For $F_n(d)$ given in equation (3.50) and $G_F(d)$ given in equation (3.49), let $\varepsilon > 0$ and $n \geq 1$ be given. Let $F_n(0) = G_F(0) = 0$ and Suppose that $|V(d+t)| \subseteq [0, M]$, where M is finite.

Recall from Section 3.5 that $\alpha = \frac{-\ln(\varepsilon)}{n}$.

Case 1: Let $d = n$. Then

$$|F_n(d) - G_F(d)| = \left| \sum_{t=-n+1}^0 A_n * e^{\alpha * t} * V(d+t) - \sum_{t=-n+1}^0 A * e^{\alpha * t} * V(d+t) \right|$$

$$|F_n(d) - G_F(d)| = (A_n - A) * \left| \sum_{t=-n+1}^0 e^{\alpha * t} * V(d+t) \right|.$$

The range of $V(d+t)$ is between 0 and M so

$$|F_n(d) - G_F(d)| = (A_n - A) * \left| \sum_{t=-n+1}^0 e^{\alpha * t} * V(d+t) \right| \leq (A_n - A) * \left| \sum_{t=-n+1}^0 e^{\alpha * t} * M \right|.$$

Using Mathematica, we have

$$|F_n(d) - G_F(d)| \leq \frac{1 - e^{-\alpha}}{e^{\alpha * n} - 1} * \frac{e^{\alpha} - e^{\alpha * (1-n)}}{e^{\alpha} - 1} * M$$

$$|F_n(d) - G_F(d)| \leq e^{-\alpha * n} * M$$

$$|F_n(d) - G_F(d)| \leq \varepsilon * M.$$

Case 2: Let $d > n$. Then

$$|F_n(d) - G_F(d)| = \left| \sum_{t=-n+1}^0 A_n * e^{\alpha * t} * V(d+t) - \sum_{t=-d+1}^0 A * e^{\alpha * t} * V(d+t) \right|$$

$$|F_n(d) - G_F(d)| = \left| \sum_{t=-n+1}^0 A_n * e^{\alpha * t} * V(d+t) - \sum_{t=-d+1}^{-n} A * e^{\alpha * t} * V(d+t) - \sum_{t=-n+1}^0 A * e^{\alpha * t} * V(d+t) \right|.$$

By the triangle inequality,

$$|F_n(d) - G_F(d)| = |(A_n - A) * \sum_{t=-n+1}^0 e^{\alpha * t} * V(d+t)| + \left| \sum_{t=-d+1}^{-n} A * e^{\alpha * t} * V(d+t) \right|.$$

From Case 1,

$$|F_n(d) - G_F(d)| \leq \varepsilon * M + \left| \sum_{t=-d+1}^{-n} A * e^{\alpha * t} * V(d+t) \right|.$$

$$\text{Since } \left| \sum_{t=-d+1}^{-n} A * e^{\alpha * t} * V(d+t) \right| \leq \left| \sum_{t=-\infty}^{-n} A * e^{\alpha * t} * V(d+t) \right|,$$

$$|F_n(d) - G_F(d)| \leq \varepsilon * M + M * \sum_{t=-\infty}^{-n} A * e^{\alpha * t}.$$

From Section 3.5, $\sum_{t=-\infty}^{-n} A * e^{\alpha * t} = \varepsilon$, so

$$|F_n(d) - G_F(d)| \leq \varepsilon * M + M * \varepsilon$$

$$|F_n(d) - G_F(d)| \leq 2 * \varepsilon * M.$$

This concludes the proof.

Example: Let us compare the 15-day exponential decay weighted average $F_n(d)$ with the moving average $G_F(d)$ for a particular data set. Based on the analysis in Section 3.5, choose $\alpha = 0.199715$. It then follows that $A_n = 0.190564$ and $A = 0.181036$.

Plotting the two functions simultaneously we have the following graph,

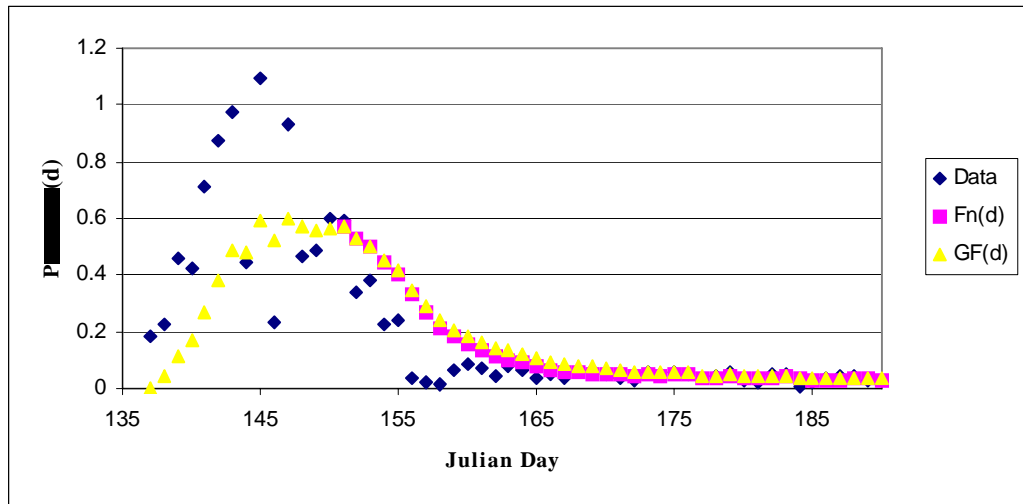


Figure 3.11 - A plot of $F_n(d)$ and $G_F(d)$ and data values $P_{net/area}(d)$ obtained from an ECOPHYS leaf during a simulation using Aspen FACE simulation data.

We can see from this graph, that although these two averages are not identical, they are quite close from the fifteenth day on.

Chapter 4 – Carbon Transportation

After photosynthate is synthesized in the leaves, the carbon allocation mechanisms within ECOPHYS move some of it to other leaves, internodes, branches, trunk and roots (Guan, 2002). In this chapter, we discuss the ECOPHYS transportation issues of combining internodes, distribution of photosynthate across a branch, and photosynthate transportation strategy following bud set. Recall that in Section 2.4 we discussed mobilization of carbon in the determinate growth phase.

4.1 Combining Internodes

After a few years of growth, individual internodes become harder to distinguish on a tree. Currently ECOPHYS has a fixed leaf initiation rate measured in hours. Based on this rate, a new leaf is created, along with its internode. Within ECOPHYS, each internode has always remained separate for the purpose of carbon transportation within the tree. Previously, internodes received photosynthate based on their locations. With the implementation of a new transportation algorithm by Yongtao Guan (Guan, 2002), the relative amount of photosynthate received in an older wood internode from a given source leaf is determined by the length of the internode. This transportation algorithm allows the internodes of older wood to be combined for faster transportation computation, because there are fewer internodes to track. Traversing the tree is also quicker with fewer internodes. For this reason, we created an algorithm that combines adjacent internodes whenever possible.

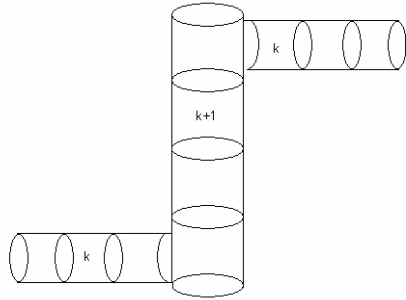


Figure 4.1a

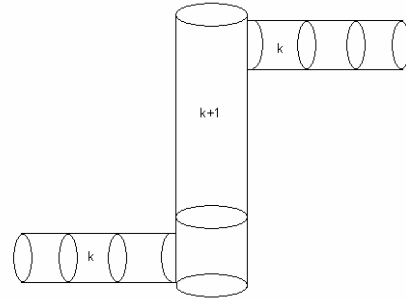


Figure 4.1b

Internodes of a branch section before they are combined together (Figure 4.1a). Branch section after they have been combined (Figure 4.1b).

The branches in Figure 4.1 are labeled based on their orders k and $k+1$. In Figure 4.1a, the four internodes of the branch of order $k+1$ are vertical and the internodes of the branches of order k are horizontal. After the function `combine_internodes()` is called, the internodes of branch $k+1$ that are between consecutive branches of order k are combined resulting in a single internode as shown in Figure 4.1b.

In the function `combine_internodes()`, two internodes of order $k+1$ are joined at a time until all of the order $k+1$ internodes between consecutive order k branches are combined. First the volumes, lengths, and masses of the two internodes are added together. Second, the diameter of the new “internode” is recalculated based on its length and volume. We then check to make sure the list of internodes is still linked together properly, deleting the internodes that are no longer linked together.

4.2 Distribution of Photosynthate Along a Branch

A photosynthate report, created in order to monitor where photosynthate was going within a tree, and to confirm that photosynthate transportation was accurate (Appendix B), revealed a problem in ECOPHYS. It was observed that there were places in a branch where one internode had enough photosynthate to grow in length or diameter, while other internodes within the same branch did not have enough photosynthate to support their maintenance respirations. The internodes with negative net photosynthate (more photosynthate needed for respiration than what they had) were, in effect, creating photosynthate to meet the respiration costs. This happened because these internodes would have their net photosynthate reset to zero at the end of each day, when it should have been negative. This was a problem because it resulted in ECOPHYS utilizing more photosynthate than it had generated. In addition to this, the inequitable distribution of photosynthate to internodes resulted in some internodes on the same branch growing in size, while internodes around them did not have enough photosynthate for respiration. It is unrealistic that one internode would be struggling, while another above it has excess photosynthate allowing thickening. Biologically, if this happened, the entire branch could conceivably die because the lowest internode dies and thus the connection from the branch to the rest of the tree.

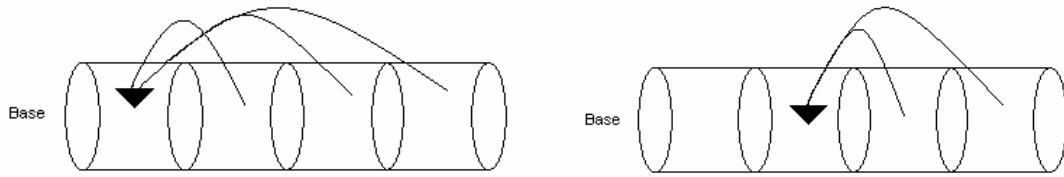
To resolve this problem, we have developed an algorithm called `redistribute_psyn_among_internodes()` that goes through all branch internodes making sure that each of the branch's internodes have enough photosynthate to support its

maintenance respiration before any of the internodes are allowed to grow in length or diameter.

This is achieved as follows. After the branches distribute the photosynthate based on the transportation models already incorporated in ECOPHYS, `redistribute_psyn_among_internodes()` is called. This function checks to see if any of the internodes have a negative photosynthate amount following maintenance respiration. If one does, then if other internodes have a positive photosynthate amount, they will send some or all of their photosynthate to the deficient internode(s).

The following source/sink algorithm has been implemented in ECOPHYS:

- 1) Traverse the tree starting with the base internode on an order k branch. This is the branch's closest internode to the attached branch of order $k+1$. If on the trunk, then start at the base of the tree.
- 2) Check to see if the current internode has a negative amount of photosynthate. If it does, then draw equal amounts from each internode farther out on the branch to raise this internode amount to zero. Continue this for each internode until we reach the outer end of the branch.



F

Figure 4.2 - a) The base internode is receiving equal amounts of photosynthate from all of the internodes along the branch regardless of how much photosynthate they have. b) The second internode is receiving from the internodes farther along on the branch.

3) After reaching the end of the branch, check the internode at the far end (top) of the branch. It is the only internode that can have a negative photosynthate amount at this stage.

4) If there is a negative amount found, check the next internode for a positive amount. If this internode has enough photosynthate to meet the needs of the top internode, it will send the amount needed. However, if it does not have enough photosynthate to meet the top internode's needs, it will send all of its available photosynthate. Go to the next internode and repeat until the top internode is brought up to zero. Figure 4.3 shows the top internode drawing photosynthate from other internodes.

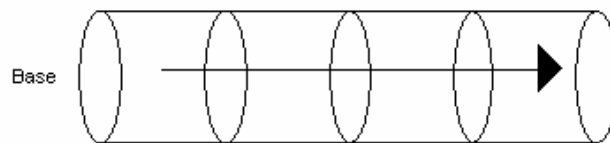


Figure 4.3 - If needed, the top internode draws photosynthate from the internodes below it.

5) If there is still a negative amount in the top internode, nothing else happens. The branch will record a negative net photosynthate amount for the day, and no internodes in the branch will grow. The branch will record photosynthate $P_{n/a} < 0$ at the top internode and use whatever it receives the next day to remove the deficit before meeting the current day's maintenance respiration.

4.3 Distribution After Bud Set

While performing test runs in ECOPHYS after the implementation of the dynamic branch death algorithm described in Chapter 2, it was noticed that in some cases shoots higher on the tree were dying before shoots lower on the tree. After further investigation, it was discovered that a shift in carbon allocation strategy that was supposed to take place after bud set was not implemented in the code.

After bud set hybrid poplars and aspen trees change their allocation strategy to prepare for the dormant season. (Isebrands unpublished). As listed in Zang (1999) shoots in ECOPHYS were supposed to change carbon allocation strategy five days before the bud-set date, which was a fixed whole-tree date. The shoots should gradually send a higher percentage of their exported photosynthate downward (instead of upward to shoot tips) until five days after bud set. At that point, 80% of the photosynthate that previously would have gone upward should now go downward.

With the new bud-set algorithm described in Chapter 2, we do not know ahead of time on what day bud set will occur for any of the branches. Because of this, we can not start the

shift in carbon allocation on a shoot until it reaches bud set. We modified the Zang (1999) algorithm to shift more carbon downward beginning on the bud-set date and continuing for ten days until 80% of the photosynthate that was previously going upward to downward. This shift in photosynthate allocation and the fact that many branches begin this shift well before the final bud-set date causes the tree to store more carbon over winter, resulting in larger growth the following year.

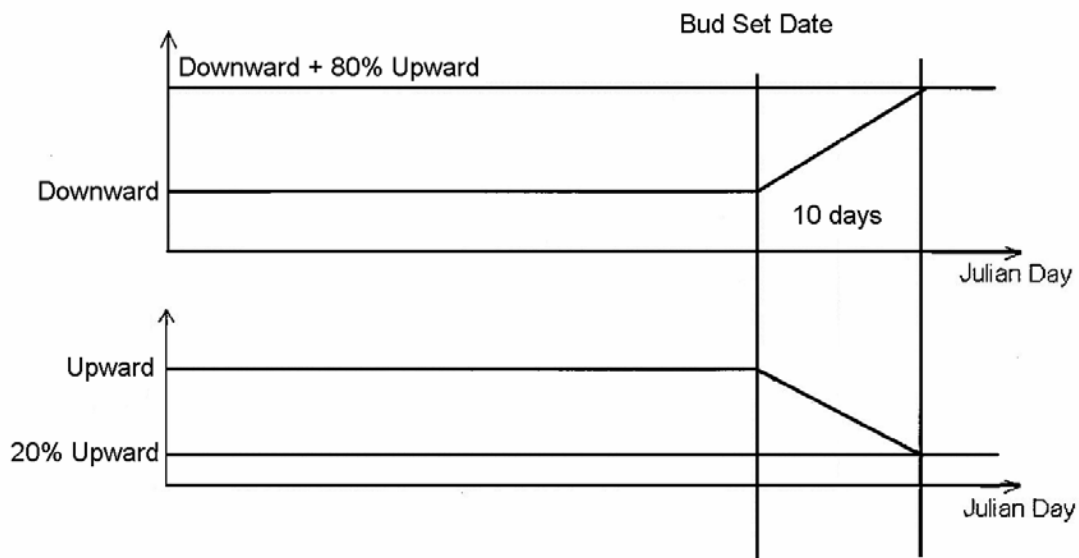


Figure 4.4 (adapted from Zang (1999)). The shoot-level shift in carbon allocation that takes place according to linear interpolation starting at the bud-set date.

Chapter 5 - Conclusions and Discussion of Future Work

After a few years of growth, there are many leaves and branches on a *Populus* tree. For instance, in the fourth year of growth, there can be over 40,000 leaves on a simulated ECOPHYS tree. Many of these are shed throughout the growing season due to the green-leaf death algorithm. We replaced the green-leaf death model (Martin et al 2001), which computed a 10-day straight average of net leaf photosynthate, with a model computing a 15-day rolling exponential decay weighted average. This improvement placed more weight on the more recent states of the leaf instead of weighting each of the most recent ten leaf states equally. We recently found that an implementation computing an exponential moving average instead of the rolling average implementation would produce approximately the same results. The motivation for implementing the exponential moving average model in ECOPHYS was increased computational performance. The rolling exponential decay weighted average implementation requires storage of 15 data values for each leaf, while the exponential moving average implementation requires storage of only the current day's datum and the previous day's weighted average.

Also implemented in ECOPHYS are process models based on the photosynthate productivity of leaves and branches. These models replace previous models based on location of leaves and branches within the tree and random chance. The shoot death and the bud set due to stress algorithms have the same structure as the mature green-leaf death algorithm. The following models were also developed and implemented to a genetics based shoot-level bud-set model determining a maximum number of days that a given shoot can grow before setting its buds. Each shoot will set its buds either according

to the bud set due to stress algorithm or by the genetics-based bud set algorithm, depending on which of these determines the earlier bud-set date. An over-winter storage algorithm based on productivity of a shoot's parent leaf determines the amount each new shoots receives based on how productive the parent leaf was the previous growing season relative to the productivity of other leaves on the branch.

We added a carbon allocation model that changes the shoot-level strategy of photosynthate transportation after a shoot sets its buds so that more photosynthate is sent downward to internodes, trunk, roots, and storage rather than upward to new shoot growth. We also added an algorithm that combines adjacent internodes when there are no sources of photosynthate in-between them. Finally, we created an algorithm to redistribute photosynthate along a branch when internodes on a branch do not have enough photosynthate to meet their respiration requirements.

5.1 Calibration of Process Models

After implementing the exponential moving average into ECOPHYS, numerical experiments were performed to test parameter sensitivity and parameterize the mature green-leaf death algorithm. I fixed $\delta=30$, which takes away most of the randomness for leaf death. Referring back to Figure 2.1, we can see that $\lim_{K \rightarrow -\infty} 1 - e^K = 1$, so as K becomes large and negative, a leaf that's photosynthate level has dropped below the threshold will more certainly die. I then varied the threshold, τ , in an attempt to find a level that would only kill the leaves that do not contribute significantly anymore. We want to kill non-

contributing leaves because if a leaf is not contributing, then its presence will only slow down the simulation and cause more shading than there should be.

When checking the sensitivity of the leaf death parameter τ , it was noticed that it had a significant effect on the height and the diameter during 3-year simulations. It was observed that this was not because the leaves that are dying do not contribute significantly, but rather because the mature green-leaf death algorithm indirectly affects other algorithms in ECOPHYS as well.

Adjusting the mature green-leaf death parameters in ECOPHYS caused chain reaction effects. First it affected the fall senescence algorithm. In the fall senescence algorithm, ECOPHYS was dropping a fixed number of leaves per day, regardless of how many leaves were on the tree. If there were fewer leaves on a branch, this resulted in the top leaves on the tree, which are the most productive to die earlier in the season than when there is a larger number of leaves. This, in turn, affected the diameter of the tree since in the fall, the top leaves continued to contribute photosynthate downward to internodes, trunk and roots until they dropped. The next-year's height was then affected because if top leaves continued to produce photosynthesis longer, then there was more available for over-winter storage. When the top leaves had a longer period of time to grow at the end of the season, they had more time to be productive. The resulting increase in over-winter storage resulted in more bud vigor and available carbon for the determinate stage of shoot growth the following early season. Shoots had stronger early-season growth, and this was observed to cause more length growth in shoot internodes over the season.

Experience at the FACE site did not support ECOPHYS' original fall senescence algorithm, which dropped a fixed number of leaves per day. To better model fall senescence, we have changed the algorithm to be based on the age of leaves rather than on their leaf plastochron index (LPI). Before starting the simulation, we choose a constant, C . When senescence begins, any leaf that has an age older than C drops. We then decrement C by a constant factor, B , each day, until all of the leaves have dropped. This new algorithm allows a fixed number of days for senescence to take place on the tree, regardless of the number of leaves. It also decouples fall leaf senescence from the green-leaf death algorithm. The previous fall senescence algorithm was sensitive to parameter values in the mature green-leaf death algorithm.

There are other factors to consider when testing the contribution of a leaf to the growth of the tree. 1) Photosynthate transportation is based on a leaf's LPI which amounts to numbering n leaves on a shoot $\{0, 1, \dots, n - 1\}$ starting from the tip of the shoot. If a leaf dies during the growing season, the LPI of each leaf below it on the shoot is decremented by one. In ECOPHYS, This can cause older leaves to change their carbon transportation strategy, which is based on LPI. 2) Inter-leaf shading also affects growth. If a leaf dies, it allows more sunlight to reach the leaves below it, enabling these leaves to contribute more to the growth of the tree. Currently, ECOPHYS' computation of photosynthetic rate is the same algorithm for sun as for shade leaves. This is something we plan to change in the near future based on the recommendation by Mark Kubiske. 3) Some leaves that die when threshold τ is larger might live if τ is chosen smaller, because if a leaf survives, it can potentially recover and later contribute to the growth of the tree.

The algorithms for mature green-leaf death, bud-set and shoot death still need to be parameterized. One way to parameterize the mature green-leaf death algorithm is to make τ sufficiently negative so that many noncontributing leaves remain on the tree. One can then raise the value of τ and observe the effects. If we then raise τ and observe the same tree growth, we can assume this to be a lower bound for τ . By this approach we can check various threshold levels, and determine which levels result in no significant reduction in tree growth. Significant growth differences will be measured in terms of tree height and diameter. For parameterizing the models it would make the most sense to first parameterize leaf death, then bud set due to stress and genetics, then shoot death, and finally distribution of over-winter storage to buds and new shoots.

5.2 Analysis of FACE Data

In the summer of 2005 FACE researchers will take various measurements and record observations that will help parameterize and test some of the models developed in this thesis.

In mid-May, they will record the number of broken and unbroken buds and bud lengths, which will give us some information on bud sizes and bud break. They are also going to measure how some of the shoots resulting from these buds grow in length throughout the season, which can help us determine if a larger bud will produce a larger branch.

On individual branches, they are going to be taking weekly measurements including leaf lengths, petiole lengths, internode lengths and diameters, number of leaves, and leaf death date. The results of this thesis suggest the need to work on a model for a more dynamic leaf initiation rate. Currently ECOPHYS initiates leaves every 46 hours until the branch has set bud. Studies have shown that the rate of initiation varies throughout the growing season under different weather and growth conditions (Pieters, 1996, 2001). The FACE data gathered in the growing season will also help us to parameterize the algorithms listed above.

5.3 Carbon Transportation

There are still improvements that need to be made on the transportation model described earlier. Leaves are sending a certain amount of photosynthate to the trunk and lower branches each day, even if the branch that the leaves are on has a photosynthate deficit. Future work includes redistributing downward photosynthate to balance the degree to which respiration needs are met within the shoot and older branches, trunk, and roots. We will be looking at some different ways to redistribute photosynthate according to source-sink relationships (Lacointe, et al. 2002).

Chapter 6 - References

Dickmann, D. I., J.G. Isebrands, J.E. Eckenwalder, J. Richardson, (2001). *Poplar Culture in North America*. NRC Research Press, Ottawa, Ontario, Canada. 397 pp.

Dickson, R.E., J.G. Isebrands (1991). Leaves as Regulators of Stress Response. *In: Mooney, H.A., W.E. Winner, E.J. Pell, eds. Response of plants to multiple stresses. San Diego, CA: Academic Press Inc.*

Dickson, R.E., K.F. Lewin, J.G. Isebrands, M.D. Coleman, W.E. Heilman, D.E. Riemenschneider, J. Sober, G.E. Host, D.R. Zak, G.R. Hendrey, K.S. Pregitzer, and D.F. Karnosky (2000). *Forest Atmosphere Carbon Transfer and Storage (FACTS-II) The Aspen Free-air CO₂ and O₃ Enrichment (FACE) Project: An Overview*.

Guan, Y., (2002). *Modeling and Computational Advances Regarding the Predictions of Tree Growth Responses to Environmental Stress*. Masters Thesis in Applied and Computational Mathematics, University of Minnesota Duluth

Holden, C., (2004). *A Parallel-Processing Implementation of ECOPHYS, a Functional-Structural Tree Growth Metabolism Model*. Mathematics Department Technical Report TR 2004-5, University of Minnesota Duluth.

Host, G.E., H.M. Rauscher, J.G. Isebrands, D.I. Dickmann, R.E. Dickson, T.R. Crow, and D.A. Michael (1990). *The Microcomputer Scientific Software Series 6. The ECOPHYS User's Manual*. USDA Forest Service North Central Forest Experimental Station General Technical Report **NC-141**.

Lacointe, A.L., J.G. Isebrands, G.E. Host (2002). A new way to account for the effect of source-sink relationships in whole plant carbon allocation models. *Canadian Journal of Forest Research* 32(10): 1838-1848.

Lenz, K.E., H.W. Stech, (2000). Student Research Participation in Multidisciplinary Tree-Soil-Atmosphere Modeling. *Proceedings of the 2000 Conference on Mathematical Modeling in the Undergraduate Curriculum*, University of Wisconsin La Crosse.

Martin, M.J., G.E. Host, K.E. Lenz, and J.G. Isebrands (2001). Simulating the growth response of aspen to elevated ozone: a mechanistic approach to scaling a leaf-level model of ozone effects on photosynthesis to complex canopy architecture. *Environmental Pollution* **115**:425-436.

Mathematica (2003). Wolfram Research Inc. Version 5.0. <http://www.wolfram.com>

Michigan Forests Forever Teachers Guide. Michigan State University Extension, (2004). <<http://www.dsisd.k12.mi.us/mff/Environment/TreePhys.htm>>.

Paritech, Australia, Technical Analysis - Exponential Moving Average (2005).

<<http://www.paritech.com.au/education/technical/indicators/trend/movavg-1.asp>>.

Pieters, G.A. (1996). The Growth of Aspen, Exposed to Ozone and CO₂ in Open Top Chambers. Report on a half year stage at the US Forest Service, North Central Forest Experiment Station, Rhinelander, WI.

Pieters, G.A., M.E. Van Den Noort, and J.A. Van Nijkerken (2001). Growth adaptation of leaves and internodes of poplar to irradiance, day length and temperature. *Tree Physiology* **19**:933-942.

Sprugel, D.G., T.M. Hinckley, W. Schapp (1991). The theory and practice of branch autonomy. *Annual Review of Ecology and Systematics* **22**:309-334.

Stech, H.W., G.E. Host, K.E. Lenz, (2004). Elementary image analysis techniques for calibrating and testing canopy light interception models. *In: Godin, Christopher; Hanan, Jim; Kurth, Winfried, eds. Proceedings of the 4th International workshop on Functional-Structural Plant Models. Montpellier, France.*

Tordsen, L. (2003). *Computational Advances in the Prediction of Light Interception in Deciduous Trees*. Masters Thesis in Applied and Computational Mathematics, University of Minnesota Duluth.

Wu, G. (1999). *A Parallel Implementation of Numerical Experiments Investigating the Shading Characteristics of Populus Eugenei*. Masters Thesis in Applied and Computational Mathematics, University of Minnesota Duluth.

Zang, G. (1999). *Development of Regulatory Components for ECOPHYS*. Masters Thesis in Applied and Computational Mathematics, University of Minnesota Duluth.

Zhao, W, (2000). *An Analytical Solution for a Mechanistic Photosynthesis Model*. Masters Thesis in Applied and Computational Mathematics, University of Minnesota-Duluth.

Appendix A - Glossary of Terms

Ambient CO₂ – The background level of carbon dioxide in the air

Ambient O₃ – The background level of ozone in the air

Autonomous –Existing as an independent entity

Bud Break – New branches emerge from buds at the beginning of a season after they swell and burst open.

Bud Set – Bud set occurs when a branch stops producing new leaves and start to develop buds.

Denova Shoots – Denova shoots are shoots that continue to grow until the final bud-set date.

Determinate Growth – The early season growth of leaves and internodes utilizing over-winter storage is referred to as determinate growth.

Fall Senescence – Fall senescence occurs when leaves routinely drop at the end of each growing season.

Growth Respiration - Growth respiration is that which is used for the synthesis of new plant material.

Indeterminate Growth – The indeterminate growth stage of leaves, internodes, and roots begins after over-winter storage has been exhausted following the determinite stage of growth.

Internode – An internode is the segment of the shoot or branch in-between, or was at one time in-between two consecutive leaves.

Long Shoot – A long shoot is a shoot that grows beyond the first spring flush of leaves.

Maintenance Respiration - Maintenance respiration is that which is used to keep existing tissue alive, functioning and healthy.

Older Wood – All aboveground wood that is not part of a shoot is referred to in this thesis as older wood.

Over-winter Storage – Over-winter storage is the carbohydrate that a tree stores over the winter and utilizes during the early part of the growing season to start the growth of shoots and roots.

PAR - Photosynthetic Active Radiation

Respiration - The chemical process by which sugars and starches produced by photosynthesis are converted into energy is called respiration.

Shoot – A shoot is a branch that was initiated at bud break (For *Populus* trees, all the leaves are on shoots).

Short Shoot – A shoot that produces only a few leaves and sets bud almost immediately in the beginning of the season and then ceases to generate new internodes or leaves is referred to as a short shoot.

Appendix B – Related ECOPHYS Work

Respiration of the Cambium Layer

The trunk diameter output that we were getting at the end of each year was smaller than it should have been. After two years, the trunk of the tree was not receiving enough photosynthate to support its maintenance respiration throughout the year. We realized that the maintenance respiration was based on the total mass of an internode instead of the part that was metabolically active.

Most of the wood on a tree is dead. The only part that is living is a narrow band called the cambium layer (Michigan, 2004). Previously, ECOPHYS had been basing maintenance respiration on the total mass of the internode. This does not make sense because most of the wood on a branch is used as a transportation route or for structural support of the canopy (Michigan, 2004). Based on this, we changed our maintenance respiration function to be based on cambium layer instead of total mass. We created a variable called “cambium_layer” for each internode. This variable keeps track of how much of an increase in volume there has been for the current season. It then gets reset to zero at the start of a new season.

Maintenance respiration is now based on the following formula:

$(tI \rightarrow \text{cambium_layer} * 1.47 * 1000) * \text{MRF}$, where 1.47 and 1000 are used for conversion and MRF is based on the respiration rate and temperature.

It is important that we pay close attention to the maintenance respiration aspect of tree growth because it is typically the most important priority for a tree (Michigan, 2004).

Each leaf has a unique vascular connection from itself to the trunk (Dickmann, 2001). When a leaf dies, this pathway remains, but it does not grow anymore. Thus, this part of the cambium layer would not need to respire anymore. Based on this, in ECOPHYS, I have created a function to reduce the cambium layer whenever a leaf or a branch dies. If a leaf dies, I reduce the cambium layer by a fraction of $\frac{1}{\text{number of leaves}}$

Photosynthate Report

We created a report that tracks where the photosynthate in a given branch is going on a daily basis. This report was created for a better understanding of exactly where photosynthate was going throughout the tree and to be used to compare with the ¹⁴C data received from Jud Isebrands, but it can also be used for error checking in general. This report is currently set up to output the total photosynthate distributed at the branch level. It can be modified, however, to give outputs at tree level as well if desired. The following information is put out by this report:

Variable	Definition
Day	Julian Day
Gross Photosynthate	The entire amount of photosynthate generated that day by the leaves on a shoot before any respiration deductions have been made

Leaf Maintenance Respiration	The amount of photosynthate respired to maintain a leaf
Total Net Photosynthate	Total branch photosynthate remaining after leaf maintenance respiration
Up to Leaves	The total amount transported in the shoot going up to leaves
Up to Internodes	The total amount transported in the shoot going up to internodes
Down to Internodes	The amount of photosynthate transported downward to internodes from the shoot
Psyn to Roots	The amount of photosynthate transported from the shoot to the roots
Total Transported	The amount of photosynthate allocated to be transported from the shoot
Leaf Growth Respiration	The amount of photosynthate respired within a leaf for growth
Remaining in Leaves	Amount of photosynthate remaining for growth in the leaves on the shoot after respiration and photosynthate transportation have taken place
Phantom Photosynthate	Amount of photosynthate created in the shoot. By default when a negative amount of photosynthate is set to 0

Psyn Lost to Leaves	Amount of photosynthate allocated to "Leaf thickness" within the shoot
Internode Maintenance Respiration	The amount of photosynthate used to maintain the internodes on the shoot
Internode Growth Respiration	Amount of photosynthate used for respiration for all internodes on the shoot in order to grow an internode in length or diameter
Branch Internode Photosynthate	Amount used to grow the internodes on the shoot in length or diameter

Specific Gravity

Looking at the model for specific gravity currently in ECOPHYS, we noticed two problems. One problem is that the range of values for specific gravity over the current terminal is larger than the range of field measurements recorded at the 2000 FACE harvest. The second problem occurs when a green-leaf dies. The equation to handle an internode without a leaf appears to be wrong based on the following:

If there is a leaf on a branch, the equation for the specific gravity of a leaf's internode is

$$SG = \beta + \alpha * LPI, \quad (\mathbf{B.1})$$

where α is fixed at .008 and β is a clonal parameter. ECOPHYS is currently growing approximately 60 leaves on the current terminal each year. When this is the case, SG will vary by as much as 0.48. The field data has a range of about 0.2. The β parameter is currently set at 0.46 so this creates a large range that is inconsistent with the field data.

For an internode without a leaf,

$$SG = \beta + \alpha * 10. \quad \text{(B.2)}$$

This applies to a shoot internode after its leaf dies as well as for internodes on older branches. One problem with this equation is that when a leaf dies, its internodes' LPI parameter is set to 10 no matter what the internode's LPI used to be. Because of this, there can potentially be consecutive shoot internodes with a quite different specific gravity. Another problem with this equation involves older branches. It appears that older branch internodes should have a higher SG than shoot internodes, based on field data from the 2000 FACE harvest shown in Appendix A.

To solve this problem, we came up with two possible models to handle specific gravity in ECOPHYS. The first model is to find the range of SG based on the number of internodes on the current terminal. The first step involves two equations. The first equation sets the average of the internodes on the current terminal to the average of the field data from the 2000 FACE harvest. The specific gravity of older branches is then set to equal the average of the SG of older branches from the harvest. These two equations are as follows where α , and β are the same as before and n is equal to the number of leaves on the current terminal. I have renamed SG to SGC for the current terminal and SGB for older branches because the value of SG is not the same in both equations.

For the shoots, we have:

$$SGC = \frac{\sum_{i=0}^{n-1} (\alpha * i + \beta)}{n} \quad (\text{B.3})$$

Solving for β :

$$SGC = \frac{\beta * n + \frac{(n-1) * (n) * (\alpha)}{2}}{n} \quad (\text{B.4})$$

$$SGC = \beta + \frac{(n-1) * \alpha}{2} \quad (\text{B.5})$$

$$\beta = SGC - \frac{(n-1) * \alpha}{2} \quad (\text{B.6})$$

For older branches, we have:

$$SGB = \beta + n * \alpha \quad (\text{B.7})$$

$$\beta = SGB - n * \alpha \quad (\text{B.8})$$

Combining (B.6) and (B.8), we have

$$SGC - \frac{(n-1) * \alpha}{2} = SGB - n * \alpha \quad (\text{B.9})$$

Solving for α gives us

$$\alpha = \frac{2 * (SGB - SGC)}{n + 1} \quad (\text{B.10})$$

Putting alpha back into a previous equation:

$$\beta = SGC - \frac{(SGB - SGC) * (n-1)}{n + 1} \quad (\text{B.11})$$

Therefore, the equation for SG on the current terminal becomes:

$$SG = SGC - \frac{(SGB - SGC) * (n - 1)}{n + 1} + \frac{2 * (SGB - SGC) * LPI}{n + 1} \quad (\text{B.12})$$

which can be simplified to

$$SG = SGC + \frac{(SGB - SGC)}{n + 1} * (-n + 1 + 2 * LPI) \quad (\text{B.13})$$

For older branches, SG would always equal SGB (replace LPI with n in (B.13)), where n will be the current number of leaves and SGB, SGC, and LPI will be known.

Using these equations for α and β the first model would let n be the current number of leaves each time the function, GrowInternode(), is called.

A second model makes an estimate of how many leaves there will be at the end of the growing season and to use this estimate to have fixed parameters for α and β for each clone.

Densities at each height growth increment(HGI) from the 2000 FACE harvest					
Aspen 216		2000	1999	1998	1997
		CT	HGI	HGI	HGI
		Density	Density	Density	Density
Clone	Ring	g/cm ³	g/cm ³	g/cm ³	g/cm ³
216	1.1	0.350	0.390	0.422	0.457
216	2.1	0.373	0.355	0.433	0.427
216	3.1	0.356	0.384	0.393	0.418
216	1.2	0.366	0.384	0.454	0.509
216	2.2	0.370	0.402	0.368	0.396
216	3.2	0.352	0.396	0.392	0.406
216	1.3	0.404	0.460	0.430	0.451
216	2.3	0.416	0.406	0.385	0.410
216	3.3	0.316	0.373	0.349	0.372
216	1.4	0.527	0.389	0.381	0.420
216	2.4	0.388	0.402	0.414	0.406
216	3.4	0.351	0.361	0.366	0.416
Average		0.381	0.392	0.399	0.424
Average of Older Branches				0.405	

Densities at each height growth increment (HGI) from the 2000 FACE harvest					
Aspen 259					
		2000	1999	1998	1997
		CT	HGI	HGI	HGI
		Density	Density	Density	Density
Clone	Ring	g/cm ³	g/cm ³	g/cm ³	g/cm ³
259	1.1	0.380	0.412	0.472	0.436
259	1.2	0.381	0.452	0.392	0.447
259	1.3	0.424	0.518	0.409	0.437
259	1.4	0.450	0.440	0.390	0.433
259	2.1	0.298	0.359	0.335	0.350
259	2.2	0.434	0.396	0.378	0.455
259	2.3	0.382	0.357	0.350	0.396
259	2.4	0.353	0.317	0.326	0.400
259	3.1	0.417	0.391	0.350	0.377
259	3.2	0.282	0.336	0.332	0.403
259	3.3	0.369	0.363	0.316	0.384
259	3.4	0.346	0.348	0.345	0.409
Average		0.376	0.391	0.366	0.411
Average of the older branches				0.389	

Densities at each height growth increment (HGI) from the 2000 FACE harvest					
Aspen 271					
		CT			
		Density	Density	Density	Density
Clone	Ring	g/cm ³	g/cm ³	g/cm ³	g/cm ³
271	1.1	0.313	0.438	0.392	0.410
271	2.1	0.392	0.383	0.396	0.392
271	3.1	0.467	0.475	0.395	0.391
271	1.2	0.355	0.344	0.344	0.378
271	2.2	0.344	0.355	0.350	0.413
271	3.2	0.451	0.422	0.367	0.456
271	1.3	0.433	0.483	0.380	0.440
271	2.3	0.406	0.403	0.484	0.416
271	3.3	0.418	0.396	0.348	0.403
271	1.4	0.367	0.403	0.393	0.437
271	2.4	0.382	0.392	0.360	0.410
271	3.4	0.344	0.355	0.353	0.416
Average		0.389	0.404	0.380	0.414
Average of the older branches				0.399	

Appendix C - Technical Analysis Paper for Exponential Moving Average

Exponential Moving Average

Description

An exponential (or exponentially weighted) moving average is calculated by applying a percentage of today's closing price to yesterday's moving average value.

Because most investors feel more comfortable working with time periods rather than with percentages, MetaStock Pro converts days into an exponential percentage. For example, if a 21-day exponential moving average is requested, a 9% moving average is calculated.



To calculate a 9% exponential moving average of IBM: First, we would take today's closing price and multiply it by 9%. We would then add this product to the value of yesterday's moving average multiplied by 91% (100% - 9% = 91%).

$$m.a. = [(today's\ close) \times 0.09] + [(yesterday's\ m.a.) \times 0.91]$$

The formula for converting days to exponential percentages is as follows:

$$exponential\ percentage = \frac{2}{time\ periods + 1}$$

For example, to calculate a 10-day exponential moving average, you would use 0.18:

$$0.18 = \frac{2}{10 + 1}$$

To convert an exponential percentage into time periods, you would use the following formula:

$$time\ periods = \frac{2}{percentage} - 1$$

Using our previous example, we can check to see that a 0.18 exponential moving average is actually a 10-day average.

$$10 = \frac{2}{0.18} - 1$$

The method used to calculate an exponential moving average puts more weight toward recent data and less weight toward past data than does the simple moving average method. This method is often called exponentially weighted.

Appendix D – Code for Process Models and Carbon Transportation

Mature green-leaf death

```
bool Branch::CheckLeafDeath(double x, double * result)
{
    // int i;
    // double aver_net_psyn;
    Internode * tI;
    double K;
    double a;
    double b;
    double c;
    double d;

    *result = x;

    for (tI = m_base_internode; tI != NULL; tI = tI->m_next_internode)
    {
        if(tI->B != NULL)
        {
            if(tI->B->m_number_of_leaves >= 5)
            {
                tI->B->Productive = true; //this means it is not a short shoot
            }
            tI->B->CheckLeafDeath(x,result);
            x=*result;
        }
        else
        {
            if((tI->L != NULL) && (tI->L->age > 15))
            {
                //This is for our weighted average //Referece Roskoski Thesis

                K = (tI->L->m_weighted_score - m_tree->m_owner-
>get_leaf_drop_threshold()) * m_tree->m_owner->get_leaf_drop_factor();

                a = rand() / (double)RAND_MAX;
                b = 1 - exp(K);

                if(K < 0 && a < b)
                {
                    tI->reduce_cambium_layer(1); //we want to do this before we
delete the leaf

                    x += tI->L->leaf_mass;
                    *result = x;
                    delete tI->L;
                    tI->L = NULL;
                    tI->m_psyntotal = 0;
                    tI->allowBranch = false; //when a leaf dies, it can not have a
branch

                    m_number_of_leaves--;
```

```

    }
    }
}
return true;
}

```

The following function will change the weights for the green-leaf drop.

```

void Branch::UpdateLeafPsynScore()
{
    Internode * tI = m_base_internode;
    while(tI != NULL)
    {
        if(tI->B != NULL)
            tI->B->UpdateLeafPsynScore();
        else
        {
            if(tI->L != NULL)
            {
                tI->L->m_psyn_score = tI->L->m_psynLeaf / (tI->L->l_area);
                tI->L->m_weighted_score += m_tree->m_owner->get_alpha_weight()
* (tI->L->m_psyn_score - tI->L->m_weighted_score);
                double a = tI->L->m_weighted_score;
                tI->L->m_weighted_score = (1-m_tree->m_owner-
>get_alpha_weight()*tI->L->m_weighted_score+m_tree->m_owner->get_alpha_weight()*tI->L-
>m_psyn_score);
                double b = tI->L->m_weighted_score;
            }
        }
        tI = tI->m_next_internode;
    }
}

```

Bud Set Due to Stress

```

void Branch::daily_bud_set(int _day)
{
    double k;

    for(Internode * tI = m_base_internode; tI != NULL; tI = tI->m_next_internode)
    {
        if(tI->B)
            tI->B->daily_bud_set(_day);
    }

    k = (branch_weight - m_tree->m_owner->get_bud_set_threshold()) * m_tree->m_owner-
>get_bud_set_factor();
    double a = rand() / (double) RAND_MAX;
}

```

```

double b = 1 - exp(k);

if (((k < 0.0) && (a < b)) && (LgrowthMaxed == false))
{
    LgrowthMaxed = true;
    budsetdate = _day;
}
}

```

for the weights...

```

void Branch::Update_Carbon()
{
    current_day_branch_psyn = 0;
    for (Internode * tI = m_base_internode; tI != NULL; tI = tI->m_next_internode)
    {
        if(tI->B)
            tI->B->Update_Carbon();
        current_day_branch_psyn += tI->i_psyn;
    }
    branch_weight += m_tree->m_owner->get_alpha_weight() * (current_day_branch_psyn -
branch_weight);
}

```

Shoot Death

```

void Branch::shoot_death(int _day)
{
    Internode * tI = NULL;
    double K;
    int number_of_leaves_to_delete = 0;

    for (tI = m_base_internode; tI != NULL; tI = tI->m_next_internode)
    {
        if (tI->B)
        {
            if ((_day - tI->B->budsetdate >= 15) && (tI->B->budsetdate != 0))
            {
                K = (tI->B->branch_weight - m_tree->m_owner-
>get_shoot_death_threshold()) * m_tree->m_owner->get_shoot_death_factor();
                double a = rand() / (double) RAND_MAX;
                double b = 1 - exp(K);
                if (K < 0.0 && a < b && tI->B->bud_dynamics_bud_set_date !=
m_tree->m_owner->get_bud_break_date()) //we do not want to remove short shoots at this point
                { //it appears that short shoots die, but they're just long shoots that do
not get long
                    //reduce cambium layer before deleting branch
                    number_of_leaves_to_delete = tI-
>number_of_leaves_deleted();
                    tI->reduce_cambium_layer(number_of_leaves_to_delete);
                    m_number_of_leaves -= number_of_leaves_to_delete;
                    delete tI->B;
                    tI->B = NULL;
                }
            }
        }
    }
}

```

```

        tI->allowBranch = false;
    }
}
}

for (tI = m_base_internode; tI != NULL; tI = tI->m_next_internode)
{
    if (tI->B != NULL)
    {
        tI->B->shoot_death(_day);
    }
}
check_old_branches();
}

```

Older-wood Branch Death

```

void Branch::check_old_branches()
{
    //TODO confirm that if the upper half of the branch is dead, that we delete the internodes with
    //no branches above them
    for (Internode * tI = m_base_internode; tI != NULL; tI = tI->m_next_internode)
    {
        bool does_branch_live = false;
        if (tI->B)
        {
            tI->B->check_old_branches();
            //we stay on a branch of lower order so that we can delete the higher order
            branches if necessary
            for (Internode * aI = tI->B->m_base_internode; aI != NULL; aI = aI-
            >m_next_internode)
            {
                if (aI->allowBranch) //this will check if an internode has branches or if
                it can grow branches
                {
                    good enough.
                    does_branch_live = true; //if one internode meets criteria,
                    break;
                }
            }
            if (does_branch_live == false) //if the branch does not and can not have
            branches, then kill it
            {
                delete tI->B;
                tI->B = NULL;
                tI->allowBranch = false;
            }
        }
    }
}

```

Bud Vigor and Death

//This function assigns weights to each bud based on how far on a branch they are

void Branch::bud_weight_assignment()

```
{
    for (Internode * tI = m_base_internode; tI != NULL; tI = tI->m_next_internode)
    {
        if (tI->B != NULL)
        {
            tI->B->bud_weight_assignment();
        }
        else if (tI->m_psyntotal > 0 && tI->allowBranch == true)
        {
            //we use true LPI here like the biologists, so add 1 in order that we dont divide
            //tI->m_psyntotal /= (tI->leaf_lpi + 1);
            m_weight_total += tI->m_psyntotal;
        }
        else
        {
            tI->m_psyntotal = 0.0;
        }
    }
}
```

//Based on our totals from bud_weight_assignements, normalize the weights to each bud

void Branch::calculate_overwinter_storage_percentage_to_internodes()

```
{
    for (Internode * tI = m_base_internode; tI != NULL; tI = tI->m_next_internode)
    {
        if (tI->B != NULL) //If there is a branch
        {
            tI->B->calculate_overwinter_storage_percentage_to_internodes(); //traverse this
            branch
            tI->m_psyntotal = 0.0;
        }
        else if (tI->allowBranch == true)
        {
            tI->m_psyntotal /= m_weight_total; //divide each branch's total by overall total
        }
        else
        {
            tI->m_psyntotal = 0.0; //no branch allowed from internode so do not assign it a
            weight
        }
    }
}
```

//This function distributes overwinter storage to new branches at the beginning of

//a new season

void Branch::distribute_overwinter_storage_to_new_branches(int tree_age)

```
{
    Internode * tI = NULL;
```

```

Internode * tA = NULL;
double remaining_overwinter_storage;

for (tI = m_base_internode; tI != NULL; tI = tI->m_next_internode)
{
    if (tI->B)
    {
        tI->B->distribute_overwinter_storage_to_new_branches(_tree_age);
    }
}
if (m_branch_order == (_tree_age - 1))
{
    leaders_bud_strength = OverWinterStorage * m_tree->m_owner-
>get_top_internode_overwinter_storage_percentage();
    remaining_overwinter_storage = OverWinterStorage - leaders_bud_strength;
    //find the top branch of this branch
    for (tI = m_base_internode; tI != NULL; tI = tI->m_next_internode)
    {
        if (tI->m_next_internode == NULL)
            tA = tI;
    }
    //FILE * filename;
    //filename = fopen("file.txt", "a");

    for (tI = m_base_internode; tI != NULL; tI = tI->m_next_internode)
    {
        if (tI->m_psyntotal >= 0)
        {
            tI->bud_strength = remaining_overwinter_storage * tI->m_psyntotal;
            if (tI == tA)
            {
                tI->bud_strength += leaders_bud_strength;
                leaders_bud_strength = tI->bud_strength;
            }
            tI->m_psyntotal = 0.0;

            //check to see if the branch is "newly old wood" and see if it has
            //to allow a branch to be created
            if ((tI->bud_strength <= 10) && (m_branch_order == (_tree_age - 1)))
            {
                tI->allowBranch = false;
            }
        }
    }
    //fclose(filename);
}
}

```

Bud-set Date Due to Genetics

//This method will set the bud dynamics bud set date based on it is own bud_strength

```

//relative to the leaders bud_strength
void Branch::set_bud_dynamics_bud_set_date()
{
    Internode * tI = m_base_internode;

    if(tI->ParentBranch->BranchInternode)
    {
        if(bud_dynamics_bud_set_date==366)
            bud_dynamics_bud_set_date = (int)(m_tree->m_owner->get_bud_break_date()
+ (OverWinterStorage/tI->ParentBranch->BranchInternode->ParentBranch-
>leaders_bud_strength)*(m_tree->m_owner->get_bud_set_date() - m_tree->m_owner-
>get_bud_break_date()));
    }
    for(tI = m_base_internode; tI != NULL; tI = tI->m_next_internode)
    {
        if(tI->B)
        {
            tI->B->set_bud_dynamics_bud_set_date();
            //check if the branch should be a short shoot or not
            if (tI->B->OverWinterStorage <= 10)
            {
                tI->B->bud_dynamics_bud_set_date = m_tree->m_owner-
>get_bud_break_date();
            }
        }
    }
}

```

Combine Internodes

```

//This function will append internodes that do not have a leaf
//to an internode that does forming one larger internode
void Branch::combine_internodes(int _tree_age)
{
    Internode * tI = m_base_internode;
    Internode * tA;
    Internode * tN;
    bool marker = 0; //checkbit 0 if previous internode had a branch, 1 if no branch
    double volume1, volume2, volume;
    while ( tI != NULL)
    {
        if (tI->B != NULL)
        {
            tI->B->combine_internodes(_tree_age);
        }
        if ((marker == 1) && (_tree_age != m_branch_order)) //dont perform this on current
branches
        {
            if (tI->B)
            {
                marker = 0;
            }
            volume1 = M_PI * sqrt(tI->i_diameter / 2) * tI->i_length;

```

```

        volume2 = M_PI * sqrt(tA->i_diameter / 2) * tA->i_length;
        //combine two internodes together
        volume = volume1 + volume2;
        tI->i_length += tA->i_length;
        tI->cambium_layer += tA->cambium_layer;
        tI->i_mass += tA->i_mass;
        tI->i_diameter = 2 * volume * volume / (tI->i_length * tI->i_length * M_PI *
M_PI);

        tI->radius = tI->i_diameter / 2;
        tI->p1.x = tA->p1.x;
        tI->p1.y = tA->p1.y;
        tI->p1.z = tA->p1.z;
        tI->i_psyn += tA->i_psyn;

        //make sure the linked list is still linked
        if (tA->m_previous_internode != NULL)
        {
            tI->m_previous_internode = tA->m_previous_internode;
            tN = tA->m_previous_internode;
            tN->m_next_internode = tI;
        }
        else
        { //make this the first internode on the branch
            m_base_internode = tI;
            tI->m_previous_internode = NULL;
            tI->ParentBranch = this;
        }
        delete tA;
        tA = tI;
    }
    else if ((tI->B == NULL) && (m_branch_order != _tree_age))
    {
        //if there isnt a branch and its not a current internode
        marker = 1;
        tA = tI;
    }
    else
    {
        marker = 0;
    }
    tI = tI->m_next_internode;
}
tI = m_base_internode;
tA = NULL;
while (tI)
{
    if (tI->B)
    {
        tA = tI;
    }
    tI = tI->m_next_internode;
}
if (tA != NULL)
{

```

```

        tA->m_next_internode = NULL;
    }
}

```

Redistribute Photosynthate Among Internodes

```

void Branch::redistribute_psyn_among_internodes()
{
    Internode * tI;
    Internode * tA;
    int count = 0;
    double amount;

    for (tI = m_base_internode; tI != NULL; tI = tI->m_next_internode)
    {
        if (tI->B)
        {
            tI->B->redistribute_psyn_among_internodes();
        }
        count++;
    }
    for (tI = m_base_internode; tI != NULL; tI = tI->m_next_internode)
    {
        count--;
        amount = tI->i_psyn;
        if (amount < 0)
        {
            amount *= -1;
            amount /= count;
            for (tA = tI->m_next_internode; tA != NULL; tA = tA->m_next_internode)
            {
                tA->i_psyn -= amount;
                tI->i_psyn += amount;
            }
        }
    }
    //get to the last internode
    for (tI = m_base_internode; tI->m_next_internode != NULL; tI = tI->m_next_internode);
    //now work our way backwards
    for (; tI != NULL; tI = tI->m_previous_internode)
    {
        if (tI->i_psyn < 0)
        {
            for (tA = tI->m_previous_internode; ((tA != NULL) && (tI->i_psyn < 0)); tA =
tA->m_previous_internode)
            {
                //if there is enough i_psyn in tA to bring the i_psyn of tI out of the
negative, do this
                if (tA->i_psyn > -tI->i_psyn)
                {
                    tA->i_psyn += tI->i_psyn; //tI->i_psyn is negative, so this
brings tA->i_psyn down a certain amount

```

```
        t1->i_psyn = 0; //get t1->i_psyn out of the negative
//if there is any i_psyn in tA, give it to t1->i_psyn
    }
    else if (tA->i_psyn > 0)
    {
        t1->i_psyn += tA->i_psyn;
        tA->i_psyn = 0; //we gave all of our i_psyn to t1->i_psyn
    }
}
}
}
}
```