1.0 Introduction

This user's guide describes the third version of a model called LANDCARB that is designed to simulate the dynamics of living and dead pools of carbon in a forest landscape. LANDCARB 3.0 can be used to examine the effects that climate, tree species, succession, natural disturbances such as wildfire, and management activities such as timber harvest and salvage, site preparation, and artificial regeneration have on carbon dynamics at the landscape level. These results are presented in ASCII output files that can be used to examine the spatial and temporal pattern of carbon stores and fluxes. While this model also estimates the mass and volume of boles removed by harvest, it tracks the fate of this harvested material using a highly aggregated submodel. Another model should be used if one intends to examine changes in forest products mixes and uses (e.g., Harmon et al. 1996).

Because LANDCARB 3.0 is designed to operate at the landscape level, it has incorporated finer levels of resolution within its computations using a meta-model approach. The meta-model approach is used to capture the overall response of more detailed simulation models of a phenomenon without all the computational burdens entailed in including the full model. LANDCARB 3.0 has been set up to mimic the stand level behaviors found in STANDCARB 2.0. This includes heterogeneity of species and ages, age-specific changes in the tree mortality rate as well as temporal lags associated with heart-rot and decomposition. LANDCARB 3.0 includes all the carbon pools that STANDCARB 2.0 does and all the same stand-level processes. To the degree possible the parameter files used by two models are the same, although they may be used in slightly different ways. Because the models are closely interrelated one can examine the effects of similar treatments at different spatial extents and expect comparable answers. For example, one might use STANDCARB 2.0 to test the effects of the interval between harvests, and LANDCARB 3.0 to test how landscape level carbon stores will change if the harvest intervals are implemented on a given landscape forest history.

Temporally, LANDCARB 3.0 is a difference model that operates on an annual time step for all variables, except those used to estimate the effects of climate on tree establishment, growth, and decomposition. These climate related variables are calculated on a monthly time step. In addition, while disturbances are simulated annually, there are arbitrary semiannual timesteps that occur once the normal growth and decomposition related processes are addressed.

Spatially, LANDCARB 3.0 is designed to simulate the dynamics of a number of cells within a landscape. Each cell represents the area occupied by a stand of trees, which can range from 0.25 to 100 ha. The stand of trees need not be homogeneous, because LANDCARB 3.0 uses a tree cohort structure to allow disturbances that remove part of each stand. The spatial location of these cohorts is not modeled directly, but the spatial interactions of cohorts is included. This approach allows the model to efficiently mimic the substand level disturbances that can be modeled in STANDCARB 2.0 without greatly imcreasing the computational demands.

Landcarbon Version 3.0

This users guide is designed to explain how to use the LANDCARB 3.0 model to investigate the effects of various types of forest management at the landscape level on live and dead carbon stores. We first provide an overview of the objectives and structure of the model. This is followed by a description of the modules used to run simulations. A brief summary of each of the major sections of the model is then described with particular attention to the equations used for critical calculations. Finally, the types and structures of the input and output files are defined.

Before using the model a final word of caution is in order. LANDCARB 3.0 is a simulation model. As such, it represents our best representation of reality, but the results must be used with caution. There are many factors that may cause the projected results to deviate from what actually occurs. This is no different than the distinction between volume yield projections and the actual harvested volume. Bear in mind each simulation has a number of tacit assumptions, and when these are not met, the projected results may be entirely misleading. It may also be the case that the simulations are correct in a relative sense but not in an absolute sense. When interpreting results bear in mind that relative differences will always be more robust than absolute differences. Finally, it must be kept in mind that simulation models are only tools to be used primarily for planning or understanding how system works. They are not a substitute for actual measurements of the actual forest carbon stores of a particular landscape.

2-Model Overview

OBJECTIVES

The object of LANDCARB 2.0 is to simulate the accumulation of carbon over succession in a landscape with mixed species-mixed aged forest stands and spatially variable climate, soil, topography, and history. This version of the model is currently parameterized for the Pacific Northwest. There is no reason, however, that it could not be used for other types of forests as well. The model can be used to investigate the landscape level effects of various regeneration strategies, harvesting, herbiciding, salvage, patch cutting, tree species replacement by design or by natural succession, site preparation, and wildfires.

The model provides output on seven live state variables, nine detritus (partially decomposed) state variables, three stable (highly decomposed) state variables, two charcoal relate variables and the volume harvested (see Output files section for more details).

LANDCARB incorporates the notion of multiple biological levels that each contribute to how carbon changes in forests (Table 2-1). Many carbon models incorporate at least several biological levels, typically physiological and ecosystem ones. LANDCARB is somewhat unsual in that it includes five: physiological, population, community, ecosystem, and landscape. Through various parameterization of the model, it is possible to remove several of these biological levels, allowing to one to test the kinds of behaviors that disappear when they are removed.

Table 1. Levels of biological organization and their contribution to LANDCARB behavior.

Physiological-response to changes of climate, radiation, soil in space and time
Population-variable colonization rates, mortality rates, and heart-rot formation
Community-species changes in space and time (i.e., succession)
Ecosystem-flows of carbon, effects of disturbance
Landscape-propagation of disturbances, dispersal of seeds

BASIC APPROACH

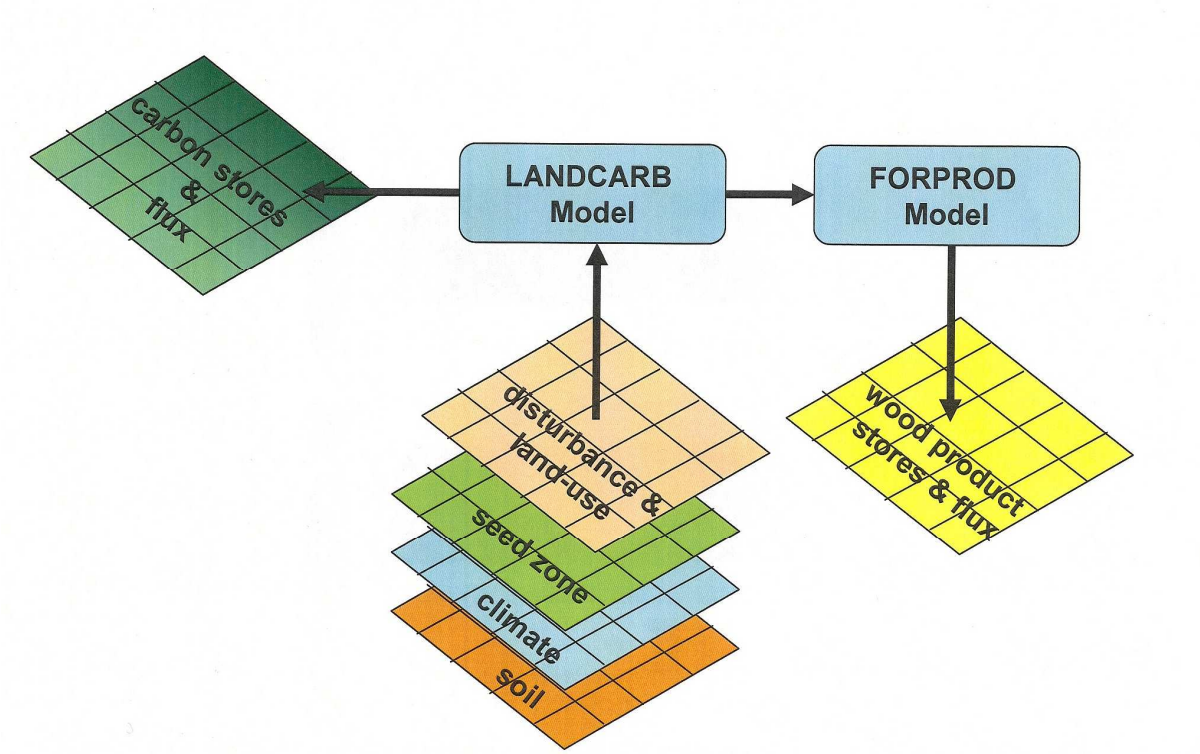

Figure 2-1. Relationship between spatial input and output data and the LANDCARB model. In the case of the Forest Sector Carbon Calculator, all spatial input data is assumed constant except for disturbance and land-use. A variation of the FROPROD model is used to estimate wood product stores, balances, and related biofuel offsets.

The approach used in LANDCARB 2.0 is to divide the landscape into a grid of cells, each grid representing a stand. Each stand is part of a specific climate, soil, radiation, disturbance and seed zone (Figure 2-1). The interactions of climate, soil, and radiation are modeled at the zone level and then applied to individual stands. Disturbance zones are used to characterize the sub-landscape disturbance regime, although the particular effect of each disturbance is calculated for each stand depending on its state and that of its neighbors. Seed zones specify the seed sources likely to be be present and their abundance, but as with disturbance the actual seed sources are dependent on the state of neighboring cells. Within each stand the abundance of different plant life forms is simulated by tracking colonization, growth, and mortality. Disturbances can create within-stand heterogeneity representing disturbances that do not kill the entire stand of trees. These cohorts of tree regeneration are allowed to interact to mimic the effect of older cohorts on younger cohorts.

LANDCARB 2.0 has a number of levels of organization it uses to estimate changes in carbon stores within a landscape (Figure 2-2). At the highest level there is a **landscape** comprized of **stand grid cells**, each which represents part of a stand of trees (given this is a raster or gridbased landscape depiction, a stand is any set of adjacent cells which have a similar species composition, environment, and disturbance history). Each stand grid cell belongs to a

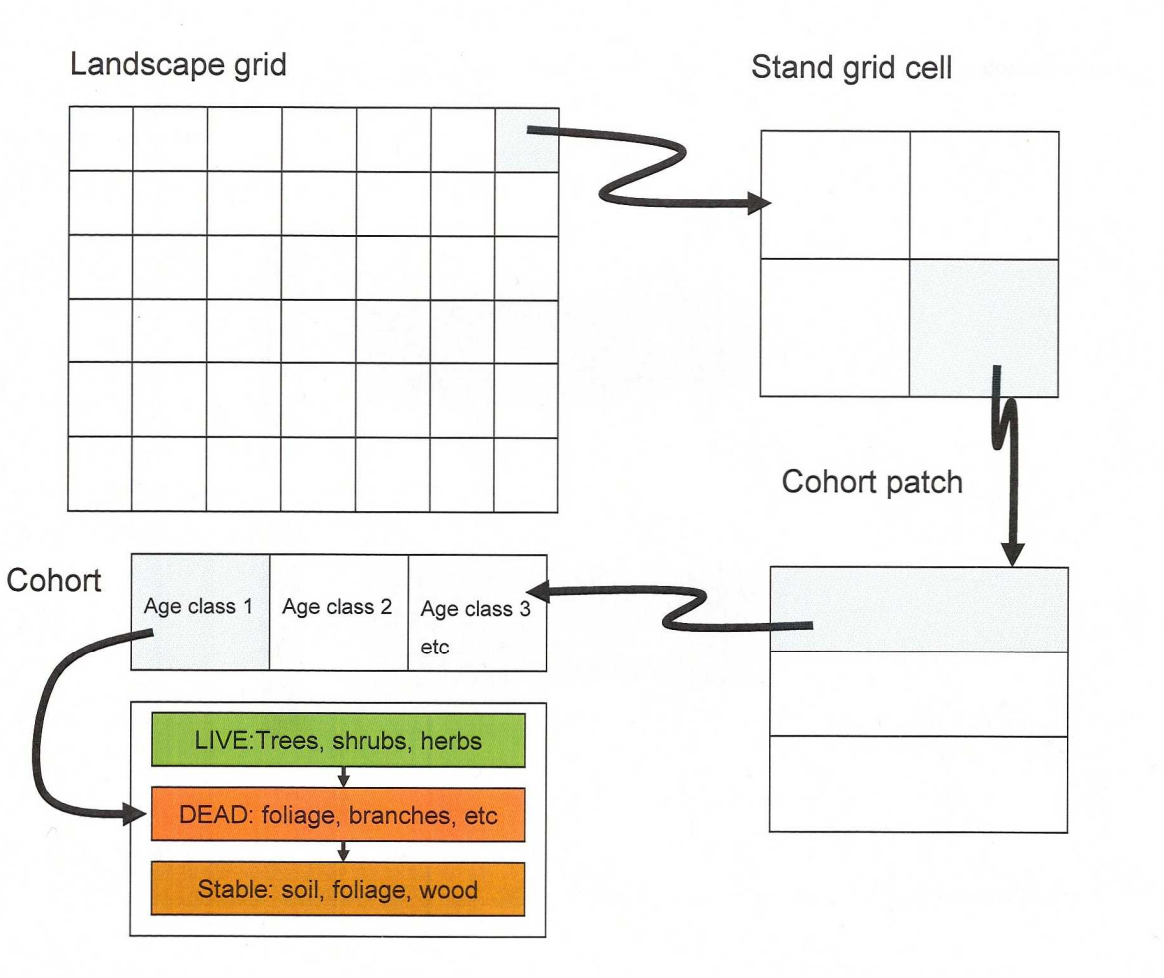

Figure 2-2. Spatial, temporal, and carbon pool units used within the LandCarb model.

series of **zones** describing key landscape attributes (disturbance regime, climate, soil, radiation, seed sources, etc). Each stand grid cell contains a number of **cohorts** (in the program this is called a cohort set) that represent different episodes of disturbance and colonization. Each cohort contains up to four **layers** of vegetation each having up to seven **live parts**, eight **dead pools**, three **stable pools** representing highly decomposed material, and two pools representing **charcoal**. The four layers of vegetation that can occur in each cohort are upper trees, lower trees, shrubs, and herbs. The two tree layers can have different species, whereas the shrub and herb layers are viewed as each representing a mix of species. Each cell can have any combination of layers except that lower trees can only occur when upper trees are present. Each layer of plants has an **age-class structure** reflecting gradual colonization of each cohort.

Each of the layers in a cohort can potentially have seven **live parts**: 1) foliage, 2) fine roots, 3) branches, 4) sapwood, 5) heartwood, 6) coarse roots, and 7) heart-rot (Figure 2-3). To make the layers correspond to the actual structure of certain life forms, herbs are restricted from having woody parts and shrubs cannot have heartwood or heart rot (as they do not form

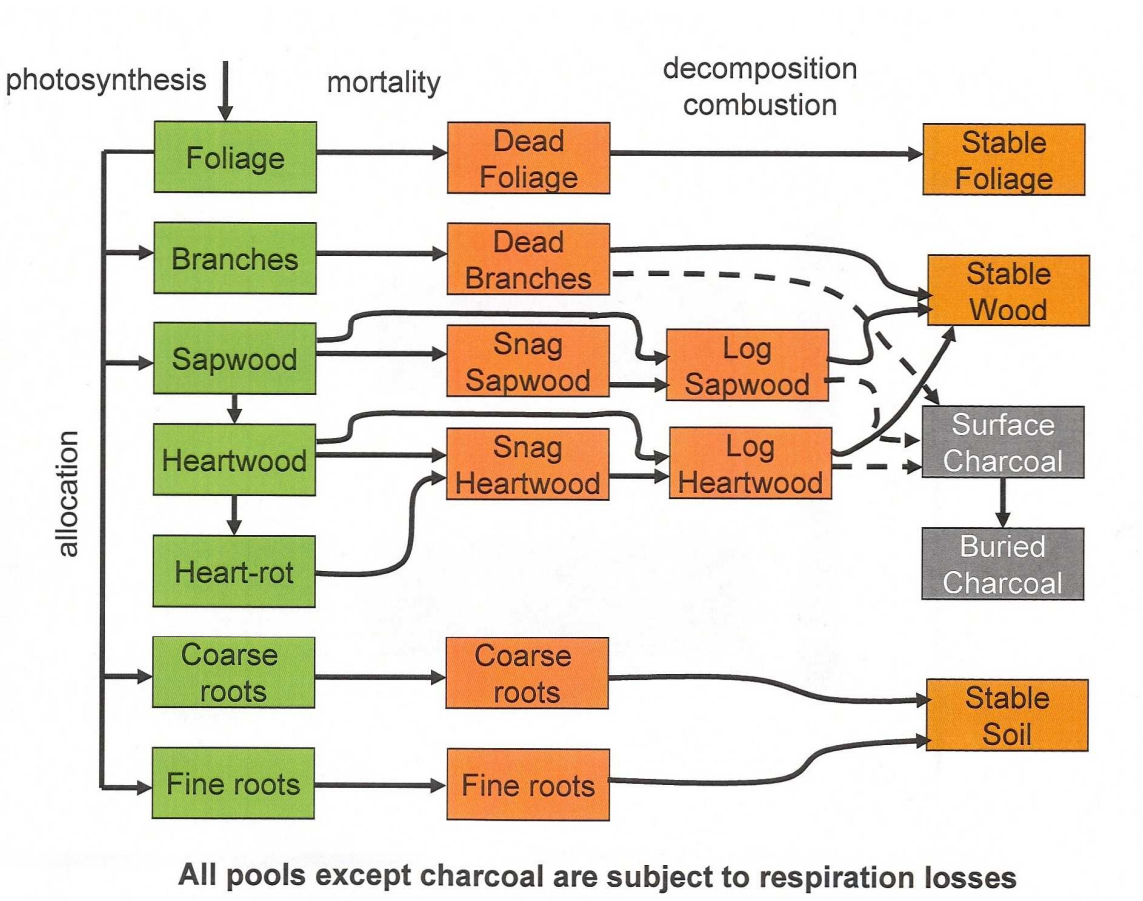

Figure 2-3 Pools of carbon tracked within LANDCARB and the processes controlling the flow of carbon between them.

a bole). All the live parts correspond to parts typically reported in the ecological literature with the exception of the bole. The later would be composed of sapwood, heartwood, and heart rot. In our model heartwood includes the heartwood and the outer bark as these are non respiring, decay-resistant layers. The sapwood includes the sapwood and the inner bark layers as these are respiring and decompose relatively quickly compared to outer bark or heartwood. Heart rot represents the portion of the heartwood that is being degraded by parasitic and saprophytic fungi inside the living trees.

Each of the live parts of each layer contributes material to a corresponding **dead pool**. Thus foliage adds material to the dead foliage, fine roots to dead fine roots, branches to dead branches, sapwood to dead sapwood, heartwood and heart rot to dead heartwood, and coarse roots to dead coarse roots. Rather than have every plant layer in each cell have its own dead pool, we have combined all the inputs from the layers of the cell to form a single detrital pool for each plant part. For example, the foliage from the four plant layers feeds into a single dead foliage pool. We have also separated dead sapwood and dead heartwood into snags versus logs so that the effects of position on microclimate can be modeled. Snags and logs

are further divided into salvageable versus unsalvagable fractions so that realistic amounts of dead trees can be removed during simulated salvage operations. All the detritus pools in a cell can potentially add material to one of three **stable pools** (stable foliage, stable wood, stable soil). The objective is to simulate a pool of highly decomposed material that changes slowly and is quite resistant to decomposition. Finally, fires can create surface **charcoal** from live parts or dead pools. Surface charcoal can be incorporated into the mineral soil and become protected from future fires as buried charcoal, whereas surface charcoal can be lost during subsequent fires.

COMPUTING ENVIRONMENT.

LandCarb is developed under Microsoft Visual C++ for .NET, using Visual Studio 2005. Most of the program code is written in ANSI C++. A small portion of code, which manages the user interface, is written in C++/CLI. LandCarb is distributed as an executable program with supporting libraries. It runs under Microsoft Windows XP and requires version 2.0 of the .NET Framework (this is available as a free download at www.microsoft.com).

MODEL MODULE OVERVIEW

LANDCARB 2.0 consists of a number of modules which perform specific functions (Figure 2-4). The following section describes the general purpose of each module. A fuller description of each can be found under the Model Documentation section.

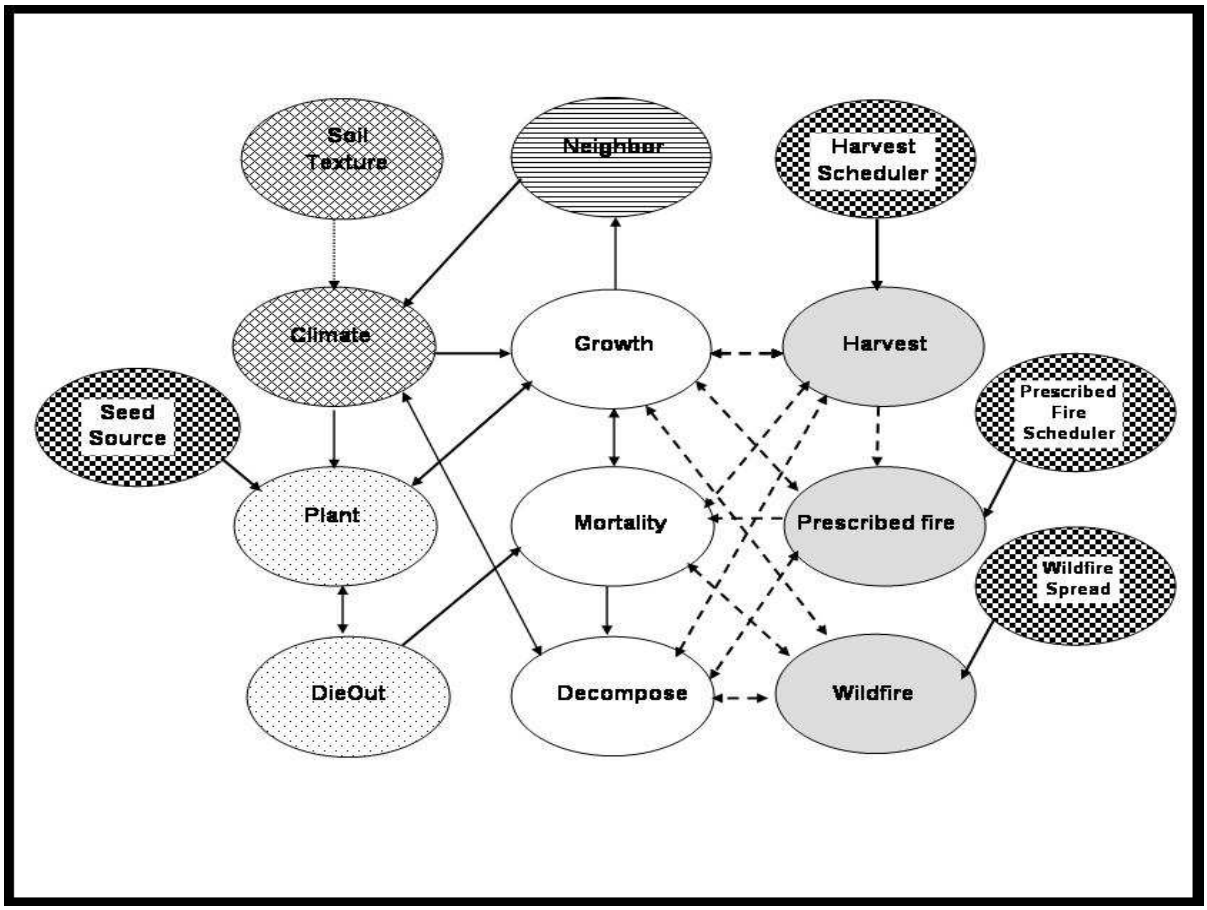

Figure 2-4. Relationship of the the modules used in the LANDCARB model.

CLIMATE. The purpose of this module is to determine the effect of climate on tree species establishment, growth, and decomposition. Climatic effects are calculated for each combination of climate-, soil-, solar radiation-zone assuming an old-growth stand structure. The results are then used for each stand grid cell that belongs to this combination of zones.

SOILTEXTURE. This module is used to calculate the effects of soil texture, depth and rockiness on the water holding capacity of soils in a soil zone. These results are used by the CLIMATE module.

PLANT. This module determines the age-class structure of plant layers in a cohort. The age-class structure of all layers is based on a fixed probability. Each cohort is given a limited time to have its layers established and if all the area is not used in the allotted time, then a new

cohort is established. This assures that each cohort has a narrow age range and age classes will be compariable in terms of size, species, etc. The upper tree layer is planted at the same time as the lower tree layer. This is unlike STANDCARB, in which the lower tree is always planted after the upper tree. The probability of a tree species colonizing a site is a function of shade tolerance, temperature, moisture limits, and the local abundance of species in surrounding stand grid cells.

DIEOUT. The purpose of this module is simulate the upper tree layer dying out in a cohort. This reflects the fact that above a certain age, trees are unable to spread horizontally. When trees in this state die either through normal mortality processes or disturbance they can not replace themselves, hence the area they cover decreases. This allows more light to reach the lower trees and consequently the underlying tree layer grows faster and eventually assumes dominance relative to the upper tree layer. This allows LANDCARB 3.0 to simulate species replacement within a cohort.

GROWTH. This module determines the rate that living plant parts grow in a cohort. The living parts tracked by the GROWTH module include foliage, branches, sapwood, heartwood, heart-rot, coarse roots, and fine roots. The rate of growth is dependent upon the amount of foliage within a cohort and the maximum rate of net production as determined by the CLIMATE module. The growth of foliage for each layer is dependent on the amount of light it receives, and the layers light extinction rate and light compensation point. Foliage mass is adjusted to reflect the lags in growth caused by the age-class structure.

MORTALITY. This module determines the rate of detrital production when a cohort has not been harvested or burned. For foliage and fine roots, a fixed proportion is assumed to die each year. These proportions are functions of the species (e.g., deciduous trees and herbs lose all their leaves each year). The proportion of branches, and coarse roots lost to pruning is a function of the light environment, as calculated in the GROWTH module, so that as the light passing through the foliage of a layer approaches the light compensation point, the pruning rate reaches a maximum. The proportion of bole-related parts, branches, and coarse roots lost to mortality varies with tree age. Initially mortality is a function of the light environment, so that as the light passing through the foliage of a layer approaches the light compensation point, the mortality rate reaches a maximum. Once trees reach their maximum horizontal extent, mortality is determined from the maximum age of trees. The transition from one form of mortality (i.e., density dependent to density independent) is influenced by the age-class structure of the tree layers. The mortality function also determines the proportion of trees dying from natural causes that form snags versus logs. This is a function of the zone a stand grid cell occurs in and the age of the cohort.

DECOMPOSE. This module determines the balance of inputs from normal mortality, harvesting and fires, and the losses from decomposition and fire for each cohort. These balances are calculated for each of the eight detritus pools (dead foliage, dead fine roots, dead coarse roots, dead branches, dead sapwood, and dead heartwood; the latter two subdivided into snags and logs) and three stable pools (stable foliage, stable wood, and stable soil organic matter). In addition to these 11 pools, this module calculates total detritus (excludes

stable pools), total stable, and total dead stores. The MORTALITY, HARVEST, PRESCRIBED FIRE, and WILDFIRE module are used to calculate detritus inputs. The rates of decomposition of each pool are determined by the species contributing detritus to a plot, and climatic effects as calculated in the CLIMATE module. Losses from fires are determined by the SITEPREP module. Lags associated with the formation of stable material, degradation of salvageable wood, and snag fall are approximated by altering the relevant transfer rate-constants when inputs exceed the long-erm average value.

HARVEST. This module determines how a stand grid cell is to be harvested or salvaged and the amount of live plant parts removed from each cohort versus the amount added to detrital pools. Harvest can remove part or all of the upper or lower tree layers and dead wood that is salvagable. For each simulation, the user can set the levels of utilization standards (the amount cut and removed). Only sapwood and heartwood (i.e., boles) either alive or dead can be removed from the simulated forest. All other living pools (leaves, branches, fine roots, and coarse roots) are transferred to the appropriate dead pools after a harvest.

PRESCRIBED FIRE. This module determines the effect of prescribed fires in a cohort. Prescribed fires can occur on their own or after timber harvest. This module kills plant layers, burns live parts as well as dead and stable pools, and forms charcoal.

WILDFIRE. This module determines the effect of wildfires fires in a cohort. This module kills plant layers, burns live parts as well as dead and stable pools, and forms charcoal.

NEIGHBOR. The purpose of this module is to determine the overall light environment of a cohort and the interaction with neighboring cohorts and stands. The degree of blocking of light is determined by the relative height of cohorts and the average distance between cohorts. A similar process is used to determine the influence of one stand gird cell upon another. Height of the tree layer is a function of the age of that layer in a cohort. The tree height of a stand grid cell is the average height of all the cohorts in a stand grid cell. The average distance between cohorts is determined from the number of within stand grid cell patches a disturbance forms. The greater the number of patches, the shorter the average distance between cohorts. The distance between stand grid cells is determined by the stand grid cell width used to portray the landscape. The heights and distances are used to compute the average angle between cohorts or stand grid cells. The steeper the angle, the lower the light amount passed onto the lower cohort or stand grid cell.

SEED DISPERSAL. This module determines the mixture of species and the abundance of tree seed sources. Seeds are assumed to originate from neighboring cells, the maximum dispersal distance determines the number of neighboring stand grid cells considered. **For carbon calculator runs, this module is turned off.**

WILDFIRE SCHEDULER. This module determines when and where a wildfire occurs. The occurrence of wildfire is random, but determined by an average interval between fires. In addition it determines the most likely fire severity that will occur.

HARVEST SCHEDULER. This module determines when and where a timber harvest occurs. In addition it determines the harvest level employed.

PRESCRIBED FIRE SCHEDULER. This module determines when and where a prescribed fire occurs. Prescribed fire can occur after harvest as a site preparation step or be independent of harvest.

ECOSYSTEM FLOWS. This uses data produced by the model to calculate important ecosystem flows related to the carbon cycle of forests such as net primary production (NPP), heterotrophic respiration (Rh), and net ecosystem production (NEP). This allows one to compare results to other forest ecosystem models and field data.

3-CLIMATE

The purpose of CLIMATE is to estimate the effect of temperature, precipitation, radiation, and soil physical properties on the establishment of tree species, growth of plants, and decomposition of dead pools. The data used to drive these estimates are found in the ClimateZone.dvr, RadiationZone.dvr, SoilZone.drv, and TopographicZone.drv files. In terms of climate and radiation, it is possible to either use average monthly climate values or to have a long term record of monthly climate and radiation variables.

This module contains 15 functions. TempConvert, DegreeDays, the functions that estimate interception (CanInterception, LogIntercept, ForFloorInterception, and Total Interception), water stores (PET & Transpire, WaterStore, and WaterPot), the effects of climate on decomposition (MoistDecayIndex, TempDecayIndex, and AbioticDecayIndex) as well as growth (TempProdIndex, MoistProdIndex, and ProdIndex) are calculated each month on each stand grid cell. The variables are named using differing prefixes depending upon the time step used and whether they are annual averages or totals. Variables calculated on a monthly time step have the prefix *Mon*. Variables that are averaged over the year have the prefix Annual, and those that are yearly totals have the prefix TotalAnnual.

To speed processing, CLIMATE is solved for every combination of the climate, radiation, soil, and topographic zone. While there can be a significant number of combinations, there is still considerably less than if each stand grid cell is modeled separately. LANDCARB creates a number of extra stand grid cells, one to represent each zone combination. Because each stand grid cell in a the actual landscape can be in different stages of succession, the CLIMATE module assumes successional status of the extra stand grid cells representing the zone combinations. Given the lack of disturbance in these extra stand grid cells, for most of a simulation the forests in these are in the old-growth stage of succession. This is suitable to provide a general indication of climate effects, but will not represent losses via run-off or other processes sensitive to forest age.

The variables calculated in this module are used by the PLANT, GROWTH, asnd DECOMPOSE modules. To keep the time step the same as used in these modules the output information has been converted to annual means or sums depending on the variable.

TempConvert Function.

This function converts mean daily temperature based on a 24 hour period into the mean day time temperature required by the TempProdIndex function. Temperature is converted to the mean daytime temperature using the mean monthly 24 hour temperature (Temp24) and the mean maximum temperature (TempMax):

*Mon*TempDay= 0.212* (*Mon*TempMax-*Mon*Temp24) +*Mon*Temp24

where *Mon*TempDay is the daytime temperature.

For the purposes of computing the respiration costs of living plant parts, the mean annual temperature (MeanAnnualTemp) is computed from the *Mon*Temp24 values.

DegreeDays Function.

This function computes the degree days for the site. This is computed once per simulation run. The degree days (DDays) is the sum of all temperatures for all the days exceeding 5.56 C. To compute DDays we first compute the mean daily temperature from the mean monthly values stored in the ClimateZone.dvr file. This is done by linearly interpolating between the midpoint of each month. The daily change in temperature between each month is:

TempChange*Mon*1=Temp24*Mon*2-Temp24*Mon*1/Julian*Mon*2-Julian*Mon*1

where these are the daily temperatures (Temp24) for the respective months and the Julian day of the midpoint of months 1 and 2. Given the daily rate of change, the Julian day, and the mean monthly temperature the daily temperature (TempDaily) is computed:

TempDaily=Temp*Mon*1+TempChange*Mon*1*(Julian-Julian*Mon*1)

where Julian is the Julian day, and the other variables are as defined above. The total degree days is then computed by adding all the temperatures of the days exceeding 5.56 C.

CanInterception Function.

This function calculates the amount of canopy interception based upon the plant life-form, mean monthly precipitation, and the mass of foliage as calculated in the GROWTH module. Each layer occupying a stand grid cell is capable of intercepting precipitation.

The interception rate generally decreases with increasing precipitation (Rothacher 1963, Lee 1980, Ward and Robinson 1990, Figure 3-1). This relationship is simulated by:

*MonLayer*CanIntRate= *Layer*CanopyInterMin
+(1- *Layer*CanopyInterMin)*Exp(-0.75**MonLayer*Precip)

where *Layer*CanopyInterMin is the minimum interception per Mg of foliage as set by the user in the GrowLayer.prm file and *MonLayer*Precip is the amount of precipitation falling through a layer.

The interception by each layer also increases linearly with increasing foliage mass (Figure 3-2). This simulates the increase in interception observed with stand age and density (Ward and Robinson 1990). As leaf mass varies seasonally for some species, canopy interception also varies monthly depending on the life-form. Deciduous trees (i.e., those trees with a foliage turnover rate of 1.0 in the Mort.prm file), herbs, and shrubs have minimal interception (5%)

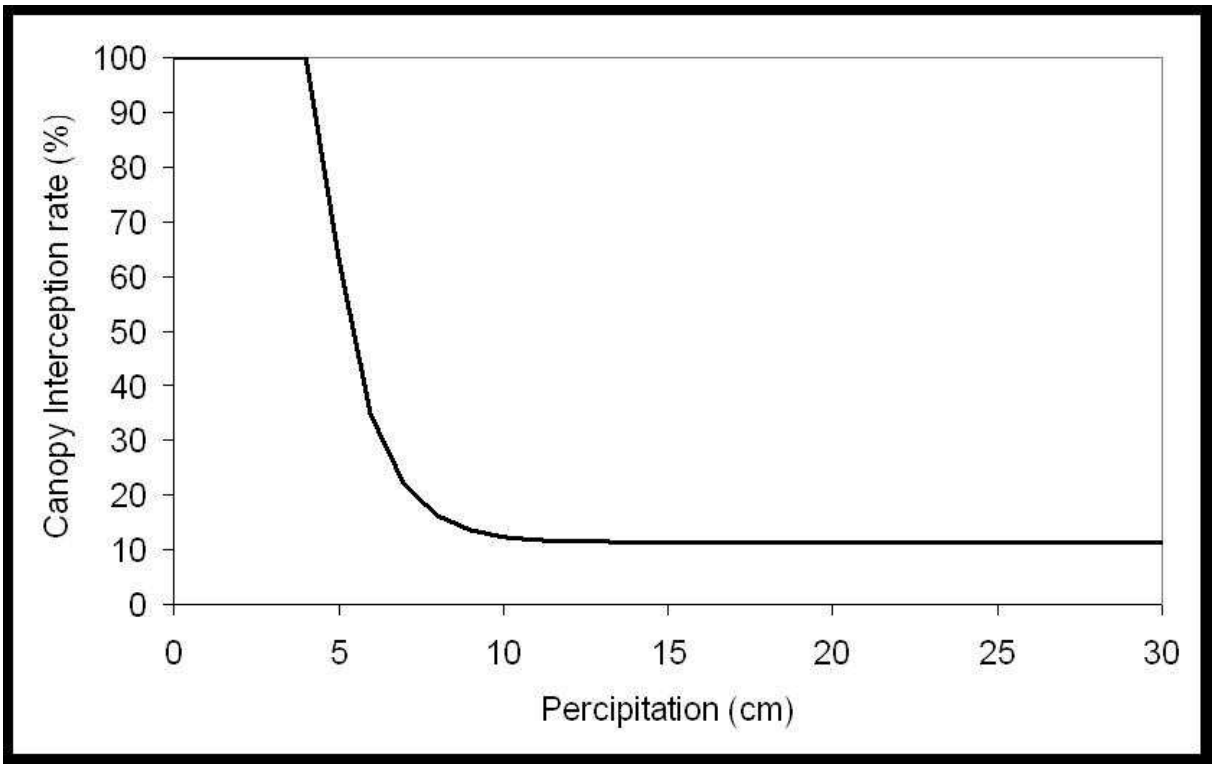

Figure 3-1. Relationship between the monthly precipitation rate and canopy interception used in the LANDCARB model.

during the non-growing season (i.e., the months of November through April). In contrast, evergreen trees have the potential for high interception year round. The proportion of precipitation intercepted by foliage of all the layers in a given month and stand grid cell is:

*Mon*CanInterception=*Mon*Precip* Σ (*MonLayer*CanIntRate**Layer*Foliage)

where *Layer*Foliage is the mass of foliage for a particular layer in a stand grid cell. Canopy throughfall is the fraction of the precipitation that is not intercepted by the canopy:

*Mon*CanThroFall=*Mon*Precip-*Mon*CanInterception

where *Mon*CanThroFall is the amount of precipitation allowed to pass through the canopy each month and *Mon*Precip is the mean monthly precipitation as defined by the Climate.dvr file.

LogIntercept Function.

This function estimates the amount of interception from the deadwood pools including those associated with snags, logs, and stable wood from their mass as calculated in the DECOMPOSE module. It also adjusts the interception as a function of the maximum moisture content of the woody material.

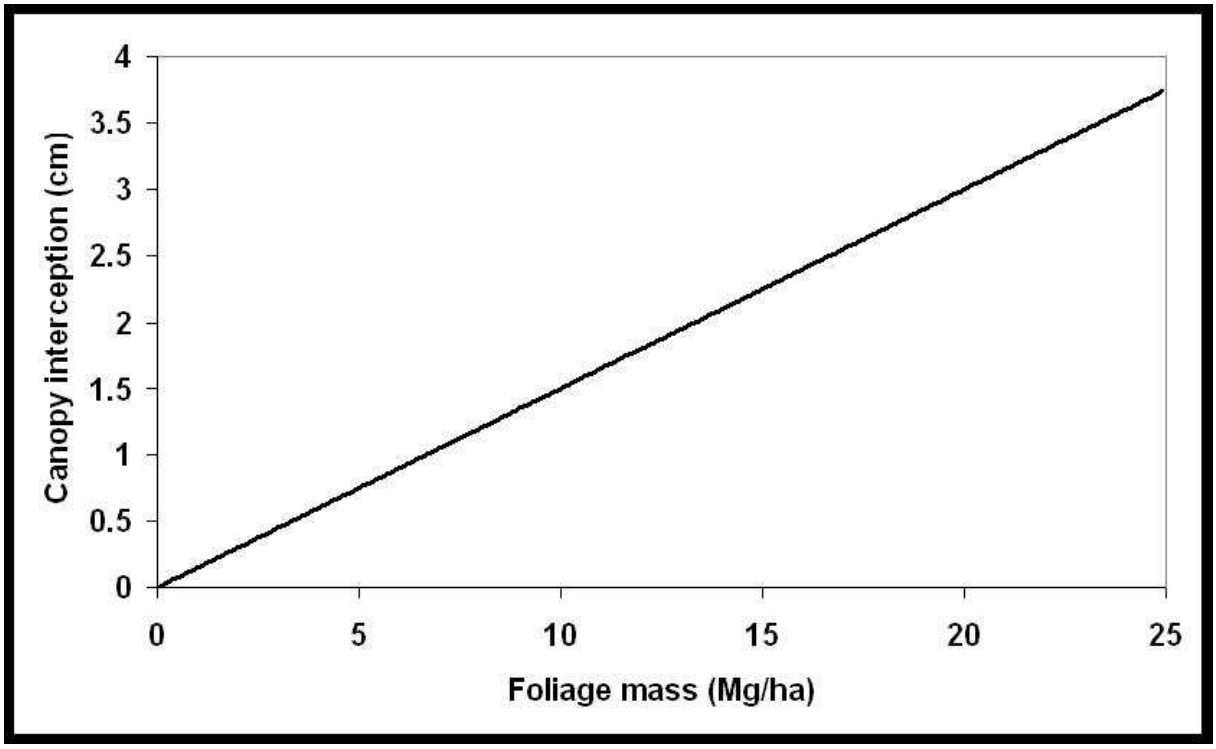

Figure 3-2. Amount of canopy interception as a function of foliage mass in the LANDCARB model.

The first step is to calculate the projected area of each dead wood pool from its mass:

*Pool*ProArea=*Pool*AreaMassRatio**Pool*/100

where *Pool*ProArea is the projected area in percent of the stand grid cell surface area, and *Pool*AreaMassRatio is the ratio of projected area to mass of the dead wood pools. The latter parameters are found in the DecayPool.prm file. Pool can be either SnagSapwood, SnagHeartwood, LogSapwood, LogHeartwood, DeadBranches, or StableWood.

The next step is to calculate the amount of *Mon*CanThroFall intercepted by the woody detrital pools. The amount intercepted is a function of the maximum potential interception based on area (MaxPotInterceptionArea) and the maximum potential based on the storage capacity (MaxPotInterceptionCap). The maximum potential interception based on area is:

MaxPotInterceptionArea=*Pool*ProArea**Mon*CanThroFall

where *Pool*ProArea is the projected area of a pool and *Mon*CanThroFall is the monthly canopy throughfall as calculated above. The maximum potential based on the storage capacity (MaxPotInterceptionCap) is calculated first by calculating the maximum storage capacity of the dead wood pool:

MaxStoresCap= StoresMax- StoresAct

where the maximum possible water store is:

StoresMax=*Pool***Pool*MoistStoreMax/100

where *Pool*MoistStoreMax is set in the DecayParm.prm file. The actual current water stores is:

StoresAct=*Pool***Mon*PoolMoist/100

where *Mon*PoolMoist is calculated by the CLIMATE WaterStore function. The maximum potential interception based on storage capacity is:

MaxPotInterceptionCap=MaxStoresCap/100

assuming there are 100 Mg/ha of water in 1 cm of precipitation.

The amount of canopy throughfall (MonCanThroFall) intercepted by the dead wood pools depends on the relationship of the maximum potentials based on area and storage capacity. If MaxPotInterceptionArea is less than or equal to MaxPotInteceptionCap then:

*MonPool*Inter=MaxPotInterceptionArea

If, on the other hand, MaxPotInterceptionArea is greater than MaxPotInterceptionCap, then:

*MonPool*Inter=MaxPotInterceptionCap

This relationship assures that the dead pool can not absorb more water than the dead pool can store. The last step is to calculate the amount of canopy throughfall that is passed on to the forest floor. The amount added to the forest floor each month as log throughfall is:

*Mon*LogThroFall= *Mon*CanThroFall- Σ *MonPool*Inter

The total interception by the dead woody pools is:

*Mon*LogInteception = Σ *MonPool*Inter

ForFloorInterception Function.

This function estimates the amount of interception by the dead foliage pool (DeadFoliage) as well as a stable carbon pool (StableFoliage) derived from dead foliage. Water is first removed by the DeadFoliage pool; whatever is passed through this layer is partially removed by the StableFoliage pool. The amount intercepted by each pool is a function of the mass as calculated in the DECOMPOSE module and the maximum moisture content of the detrital pool.

We assume that until a certain mass, these pools do not cover the stand grid cell surface completely. If these pools are less than or equal to 3 Mg/ha, the projected area is calculated as:

*Pool*ProArea=*Pool*AreaMassRatio**Pool*/100

where *Pool*ProArea is the projected area of either the dead foliage or stable pool and *Pool*AreaMassRatio defines the relationship between mass and projected area as defined in the DecayPool.prm file. If on the other hand the mass exceeds 3 Mg/ha then

*Pool*ProArea=1.0

The amount intercepted by the dead foliage and stable foliage pool each month is a function of the maximum potential interception based on area (MaxPotInterceptionArea) and the maximum potential based on the storage capacity (MaxPotInterceptionCap). The maximum potential interception based on area is:

MaxPotInterceptionArea=*Pool*ProArea**Mon*LogThroFall

where *Pool*ProArea is the projected area of a pool and *Mon*LogThroFall is the monthly log throughfall as calculated above. The maximum potential based on the storage capacity (MaxPotInterceptionCap) is calculated first by calculating the maximum storage capacity of the dead wood pool:

MaxStoresCap= StoresMax- StoresAct

where the maximum possible water store is:

StoresMax=*Pool***Pool*MoistStoreMax/100

where *Pool*MoistStoreMax is set in the DecayPool.prm file. The actual current water stores is:

StoresAct=*Pool***MonPool*Moist/100

where *MonPool*Moist is calculated by the CLIMATE WaterStore function. The maximum potential interception based on storage capacity is:

MaxPotInterceptionCap=MaxStoresCap/100

assuming there are 100 Mg/ha of water in 1 cm of precipitation.

The amount of log throughfall (MonLogThroFall) intercepted by these dead foliage related pools depends on the relationship of the maximum potentials based on area and storage capacity. If MaxPotInterceptionArea is less than or equal to MaxPotInteceptionCap then:

*Mon*PoolInterception=MaxPotInterceptionArea

If, on the other hand, MaxPotInterceptionArea is greater than MaxPotInterceptionCap, then:

*Mon*PoolInterception=MaxPotInterceptionCap

This relationship assures that these dead foliage related pools can not absorb more water than it can store. The amount added to the soil as throughfall from the dead foliage pool each month is:

*Mon*ForestFloorThroFall=*Mon*LogThroFall-*Mon*DeadFoliageInterception
-*Mon*StableFoliageInterception

Total Interception Function.

This function calculates the total amount of precipitation intercepted by the canopy, above-ground dead wood pools, and the dead foliage pool (Figure 3-3, 3-4). The monthly total interception is:

*Mon*TotalInterception=*Mon*StableFoliageInterception +
*Mon*DeadFoliageInterception+*Mon*LogInteception+*Mon*CanopyInterception

TotalAnnualInterception is the sum of all the monthly interception values.

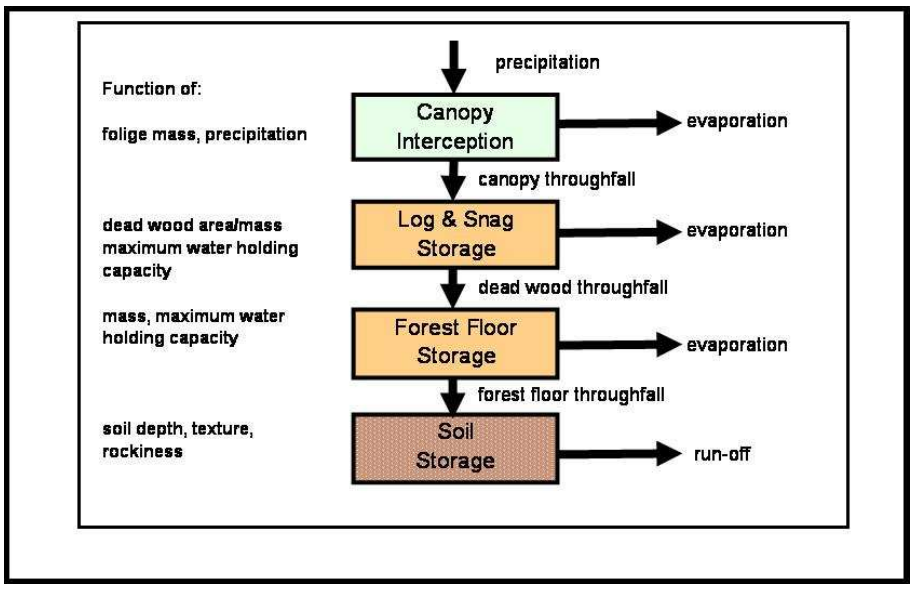

Figure 3-3. Pools and processes leading to interception in the LANDCARB model.

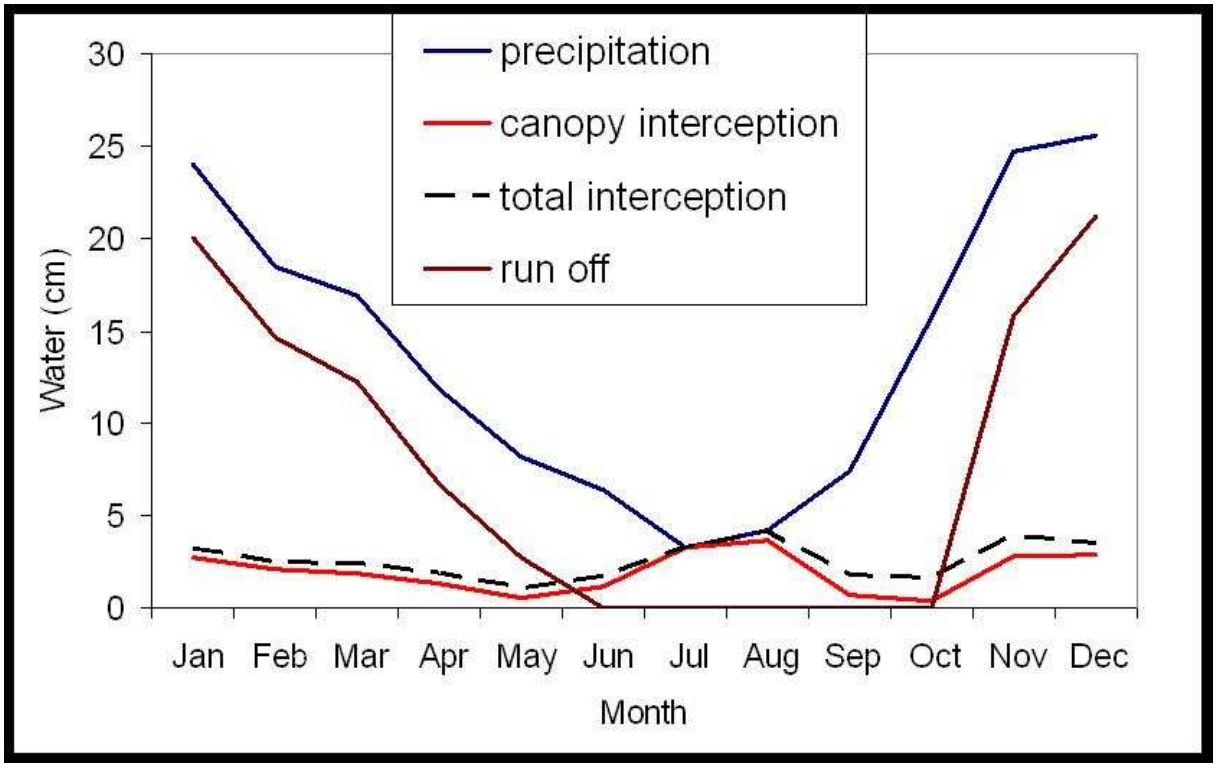

Figure 3-4. Examples of seasonal trends in water flow predicted by the LANDCARB model.

PET & Transpire Function.

This function calculates the monthly total potential evapotranspiration (in cm) of the site using a modification of the Priestly-Taylor method (Bonan 1989, Jensen 1973, Campbell 1977). Total potential evapotranspiration for a month (*Mon*PETTotal) is assumed to be proportional to the estimated solar radiation (*Mon*SolRad), the monthly mean air temperature (*Mon*Temp24 in C), and number of days in a month (MonthDay).

*Mon*PETTotal=CT*(*Mon*Temp24+TX)**Mon*SolRad*MonthDay/*Mon*LatHeatVapor

The constants CT and TX are empirically derived and calculated after Jensen and Haise (1963) and Jensen (1973):

CT = 1/[38-(2*Elev/305) + 380/SatVapPresMax-SatVapPresMin)]

TX = 2.5+(0.14*(SatVapPresMax-SatVapPresMin))+Elev/550

where Elev is the elevation in meters, SatVapPresMin and SapVapPresMax are the saturation vapor pressures in mbars for the mean minimum (TempMeanMin) and mean maximum (TempMeanMax) daily temperatures for the warmest month of the year. The vapor saturation

pressures are calculated from the appropriate air temperatures using Bosen's (1960) approximation. For example for the SatVapPresMin:

$$SatVapPresMin=33.8639*[((0.00738*TempMeanMin+0.8072)^8 - 0.000019*(1.8*TempMeanMin + 48) +0.001316)]$$

*Mon*LatHeatVapor is the latent heat of vaporization (cal) for each month and is calculated as follows:

*Mon*LatHeatVapor= 597.-0.568**Mon*Temp24

This means that each month can have its own value of latent heat of vaporization.

To estimate the potential amount of transpiration by plants (*Mon*PotenTrans), the total potential evapotranspiration (*Mon*PETTotal) is reduced by the amount of evaporation from canopy interception and dead pools:

*Mon*PotenTrans=*Mon*PETTotal-*Mon*CanInterception-*Mon*DeadEvaporation

MonPotenTrans is set so it can not go below zero. If the interception and evaporation terms are larger than PET total then set PotenTrans to zero. This yields a monthly potential transpiration loss, assuming that leaf mass and soil water stores are at a maximum (Figure 3-5). The actual transpiration losses each month (*Mon*Transpiration) are controlled by the soil water stores and the foliage mass:

*Mon*Transpiration=*Mon*PotenTrans*(*Mon-1)*MoistProdIndex*(Foliage/FoliageMax)

where (*Mon-1)*MoistProdIndex is calculated in the MoistProdIndex function and is the value of the previous month, and Foliage is the total foliage mass for all layers and FoliageMax is the maximum total foliage mass possible in a stand grid cell. This is calculated for each layer from the light compensation point (*Layer*LightCompPoint) and the light extinction coefficient (*Layer*LightExtCoeff) for each layer in a stand grid cell. The first step is to calculate the maximum light a layer (*Layer*MaxLightAbsorb) can absorb assuming that the overlying layers are also at their maximum:

*Layer*MaxLightAbsorb=*Layer*LightIn-(*Layer*LightCompPoint/100)

where *Layer*LightIn is the light not absorbed by the overlying layers and *Layer*LightCompPoint is the light compensation point of the layer as defined in the Growth.prm file. In cases where the *Layer*LightIn is less than *Layer*LightCompPoint (when the overlying layers reduce light below the light compensation point of the layer in question), *Layer*MaxLightAbsorb is set equal to zero.

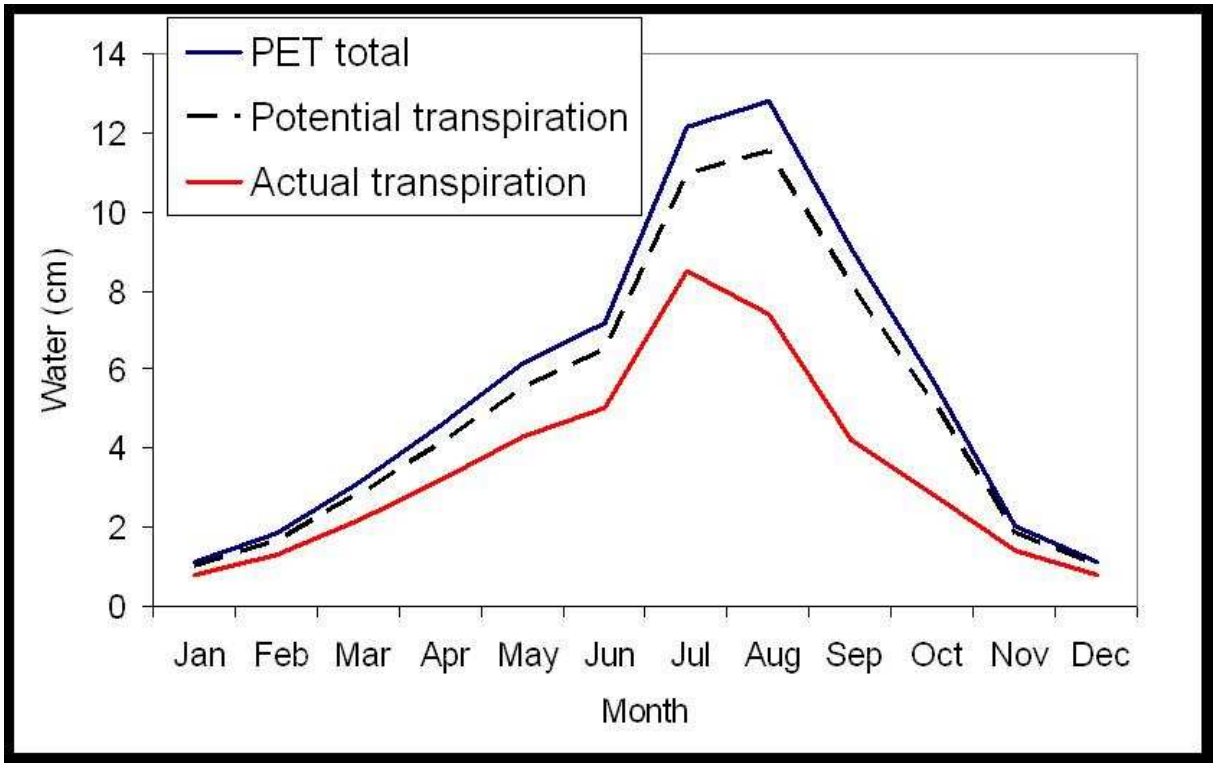

Figure 3-5. Examples of seasonal changes in transpiration predicted by LANDCARB model.

The maximum mass of foliage a layer can have, adjusted for the amount of light it can absorb is:

*Layer*FoliageMax= -Ln [*Layer*LightRemoved/*Layer*LightExtCoeff]

where *Layer*LightExtCoeff is the light extinction coefficient (defined in Growth.prm) and *Layer*LightRemoved is the amount of light removed by a layer:

*Layer*LightRemoved= (*Layer*LightIn-MaxLightAbsorbed)/*Layer*LightIn

The maximum foliage mass of all the plant layers is calculated as the sum of maximum foliage mass for all four layers in a stand grid cell:

FoliageMax=Σ(*Layer*FoliageMax).

WaterStore Function.

This set of functions determines the monthly moisture content of eight dead pools, two surface stable pools (i.e., StableFoliage and StableWood) and the mineral soil. For all pools, the moisture content is computed monthly and represents the balance of inputs through precipitation and outputs via evaporation and/or transpiration (Figure 3-6).

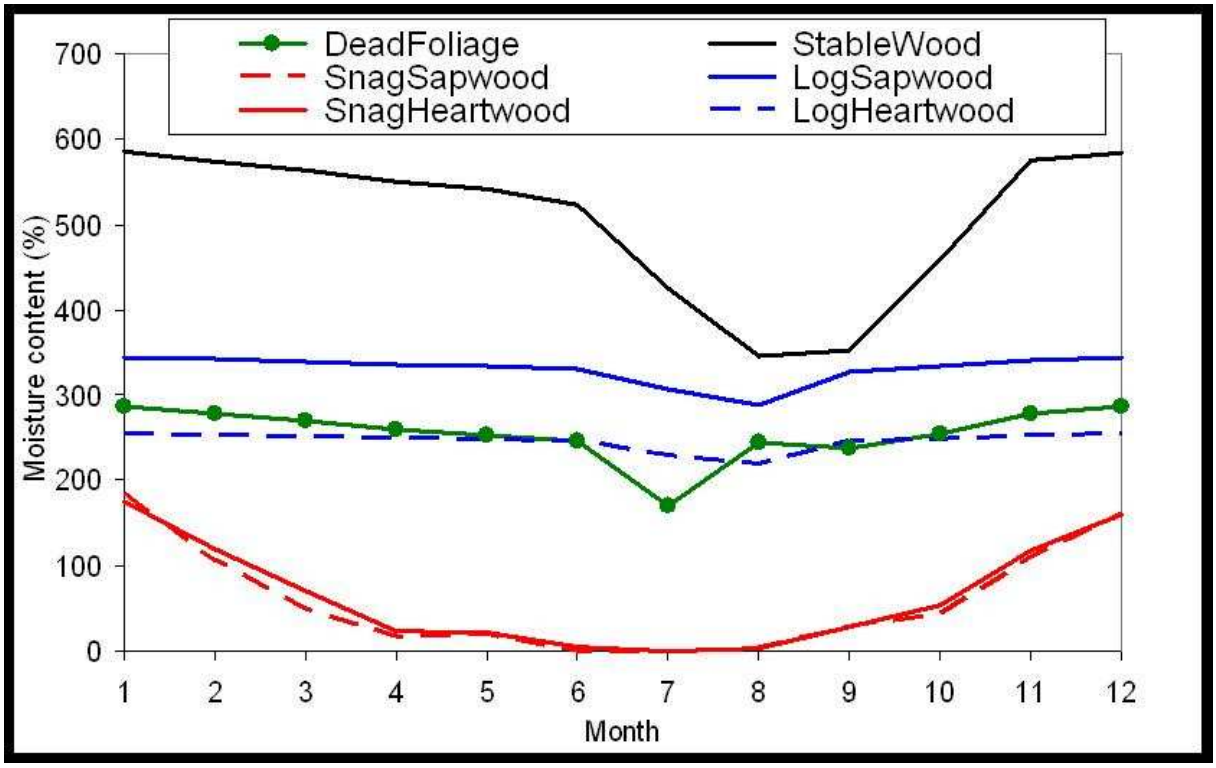

Figure 3-6. Example predictions of seasonal changes in moisture content (mass based) of selected dead carbon pools in the LANDCARB model.

Mineral Soil Subfunction.

This function computes the water stores in the mineral soil. Input to the mineral soil is whatever water has not been intercepted by the canopy, dead wood, surface stable pools, and the dead foliage pools.

*Mon*SoilWaterIn=*Mon*ForestFloorThroFall

where *Mon*SoilWaterIn is the amount of water added to the mineral soil layer in cm. The loss of water from the mineral soil will be controlled solely by the transpiration from plants, this assumes that there is always plant cover or forest floor cover. The overall balance of mineral soil water stores is therefore:

*Mon*DeltaSoilWat=*Mon*SoilWaterIn-*Mon*Transpiration

The water stored in mineral soil for each month would be:

*Mon*SoilWat= *Mon*SoilWatOld +*Mon*DeltaSoilWat.

where *Mon*SoilWatOld is the water store in the soil the previous month. To keep the water potential and other indices from becoming undefined, the minimum value that *Mon*SoilWat is allowed to have is 0.01.

To compute an overall water balance, runoff occurs when *Mon*SoilWat exceeds the SoilWaterMax (the maximum storage capacity of the soil based on its texture, rockiness, and depth as calculated by the SoilTexture module). The monthly runoff is therefore:

*Mon*Runoff=*Mon*SoilWat-SoilWaterMax

When runoff occurs the *Mon*SoilWat is set to equal SoilWaterMax. If the monthly water store is less than or equal to the maximum then *Mon*Runoff is set to zero.

The annual Runoff is the sum of all the monthly values (TotalAnnualRunoff). While this variable is not directly used in carbon budgets, it is a useful variable for model calibration.

The moisture content of the soil is calculated on a volumetric basis relative to the maximum water storage of the particular site being examined:

*Mon*SoilMoist= 100**Mon*SoilWat/SoilWaterMax

where SoilWaterMax is the maximum amount of water (cm) a soil can hold as calculated in SOILTEXTURE.

Dead Pool Water Stores Subfunction.

This function calculates the balance of water stores for four dead pools and two stable pools. The input of water into the DeadFoliage, DeadSapwood, DeadHeartwood, DeadBranch, StableFoliage and StableWood pools is equal to the amount intercepted:

*MonPool*WaterIn=*MonPool*Interception

DeadSapwood and Dead Heartwood are subdivided into snags and logs for water balance purposes. The loss of water each month from these pools is dependent upon the temperature and the amount of solar radiation received. The amount of solar radiation recieved each month is a function of the amount of light that passes through the foliage layer just above the dead pool. Therefore for Deadfoliage, DeadBranch, StableFoliage, StableWood, and DeadSapwood and DeadHeartwood in logs the amount of radiation recieved is:

*MonPool*RadiationInput=*Mon*SolRad*HerbLightOut

where *MonPool*RadiationInput is the amount of radiation recieved by a detrital pool each month, *Mon*SolRad is the total amount of solar radiation recieved by a site each month (see Radiate.dvr file) and HerbLight is the fraction of light passing through the foliage of all plant

layers each month. In the case of DeadSapwood and DeadHeartwood in snags the amount of light recieved is higher given the greater height distribution of this material. Therefore:

*Mon*Snag*Pool*RadiationInput=*Mon*SolRad*UpperTreeLightOut

where *Mon*Snag*Pool*RadiationInput is the amount of radiation recieved by DeadSapwood and DeadHeartwood in snags each month and UpperTreeLightOut is the amount of radiation passing through the upper tree layer. The rate of drying is dependent on the evaporative demand for each dead pool:

*MonPool*EvapDemand=*Mon*Temp24**MonPool*RadiationInput.

When MonTemp24 is negative, *MonPool*EvapDemand is set to 0 so that dead pools do not directly gain water from the atmosphere. The rate that water is lost from each detrital pool is:

*MonPool*RateWaterLoss=*MonPool*EvapDemand**Pool*DryingConstant

where *Pool*DryingConstant is found in the DecayPool.prm file. This parameter represents the rate of drying in a month when the temperature is 1C and the radiation input is 1 cal m^{-2} day^{-1}.

The overall rate of change for each of the detrital layers and surface stable pools is a function of the inputs versus loss through evaporation:

*Mon*Delta*Pool*Water=*MonPool*WaterIn-*MonPool*RateWaterLoss

The store of water in a pool for a given month is:

*MonPool*Water=*MonPool*WaterOld+*Mon*Delta*Pool*Water

with the restriction that *MonPool*Water can not be less than 0.

To calculate the effect of water stores in these detrital layers on the Moisture Decay Index functions, the values of water depth have to be converted to moisture content based on mass. The mass of water per hectare in 1 cm of depth is 100 Mg. Therefore each 1 cm of water stored in a detrital layer is:

*MonPool*WaterMass=*MonPool*Water*100

where *Pool* is any of the four detrital layers we are considering (DeadSapwood, DeadHeartwood, DeadBranches, and DeadFoliage). The moisture content for these pools would therefore be:

*MonPool*Moist=100**MonPool*WaterMass/*Pool*.

where *Pool* is the mass of each dead pool during the year being considered.

In this version of the model there are two layers in which the moisture content is not modeled using inputs and outputs. For the dead fine root pool (DeadFineRoot), the moisture content changes rapidly with the surrounding soil and humus. It is therefore assumed to be the same as for the StableFoliage pool. In the case of dead coarse roots (DeadCoarseRoot) the water balance is controlled by the moisture of the surrounding mineral soil (Chen 1999). We assume that the response-lag over a monthly time step is minimal. When the soil is saturated we assume that the dead coarse roots reach their maximum moisture content. Therefore:

If *Mon*SoilMoist = 100 then

*Mon*DeadCRootMoist=DeadCRootMoistStoreMax

as defined in the DecayPool.prm file. When the moisture content of the mineral soil is less than saturated we assume the dead coarse roots and mineral soil are in equilibrium. However, since mineral soil moisture is expressed in volumetric terms and dead coarse roots in mass terms we must convert units:

If *Mon*SoilMoist<100 then

*Mon*DeadCRootMoist=2**Mon*SoilMoist

Finally, to calculate the potential transpiration losses from the mineral soil it is necessary to calculate the amount of water lost via evaporation from dead and the surface stable pools (*Mon*DeadEvaporation):

*Mon*DeadEvaporation= Σ *MonPool*RateWaterLoss.

WaterPot Function.

This function converts the volumetric moisture content of soils to a xylem water potential (Figure 3-7). This relationship is represented by a reciprocal function modified by an asymptote:

*Mon*WaterPot=CorrTerm*WaterPotAsym + (WaterPot1*(SoilWaterMax/*Mon*SoilWat))

where *Mon*WaterPot is the predawn xylem water potential in MPa for a given month, *Mon*SoilWat is the monthly water store in soil, and SoilWaterMax is the maximum water stores in cm. The later variable is dependent upon the soil depth, rockiness, and texture and is calculated by SOILTEXTURE. The parameter WaterPotAsym simulates the behavior of coarse textured soils that can yield considerable water without changing their water potential. WaterPot1 is the fraction of the water stores when WaterPot is equal to 1 MPa. When this water potential is reached moisture becomes limiting to transpiration and production. The values of WaterPotAsym and WaterPot1 are defined in the Soil.prm file. Finally, the term,

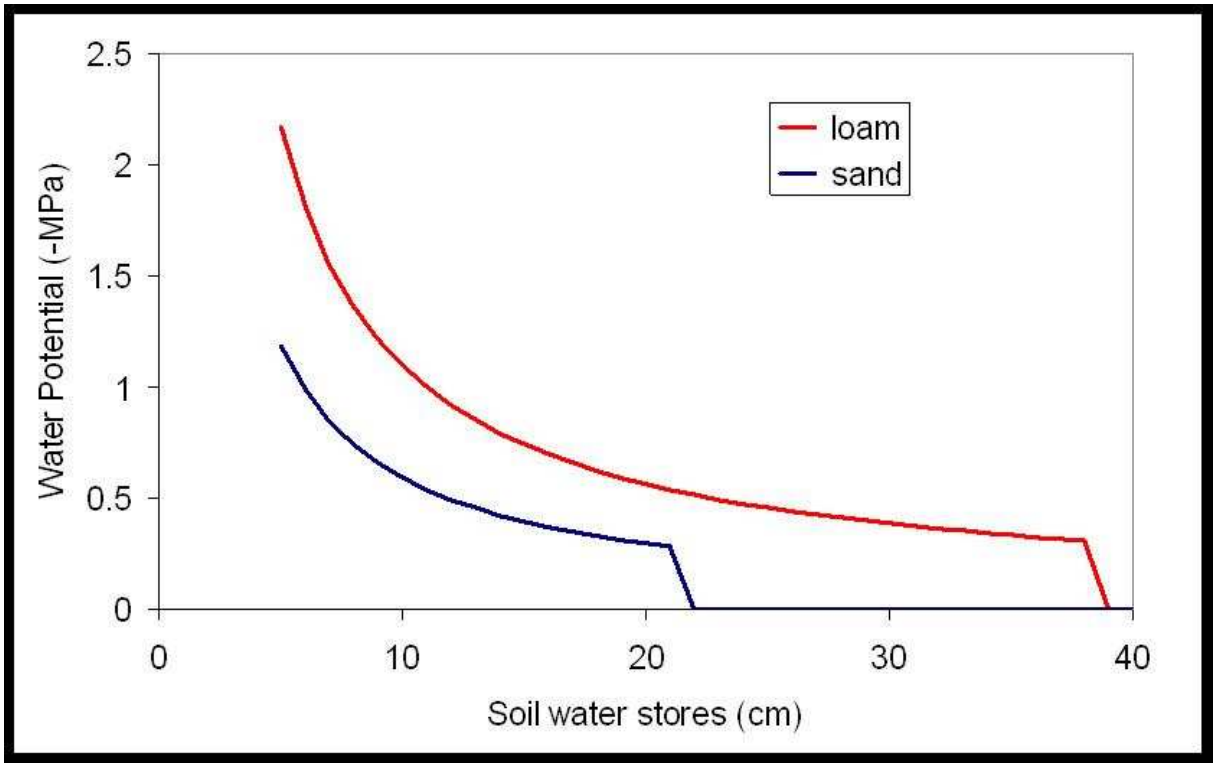

Figure 3-7. relationship between soil water stores and soil water potential used by the LANDCARB model.

CorrTerm, is used to correct for the fact that water potential does not increase appreciably from 0 when the soil is near saturation. CorrTerm is set equal to 0 if the ratio of *Mon*SoilWat/SoilWaterMax is greater to or equal to 0.9. Otherwise CorrTerm is set to 1.

MoistDecayIndex Function.

This function determines the way the moisture content (*MonPool*Moist) of each pool influences the decomposition rate of the detrital layers for each month (Figure 3-8). For all layers we assume that moisture controls decomposition in two ways. The first is through matric potential which makes water unavailable for decomposers. For most detrital forms, decomposition ceases when moisture content reaches the fiber saturation point. The second effect is caused by poor oxygen diffusion when the moisture content is too high. For most detrital layers this is not a problem, however, coarse wood respiration is often limited by this factor. We model the matric potential and diffusion limitation portions separately. For all dead pools except the stable soil pool, the percent moisture content used is based on mass of water divided by dry mass of the substrate. For the stable soil pool, the percent moisture content is based on volume of water divided by volume of soil.

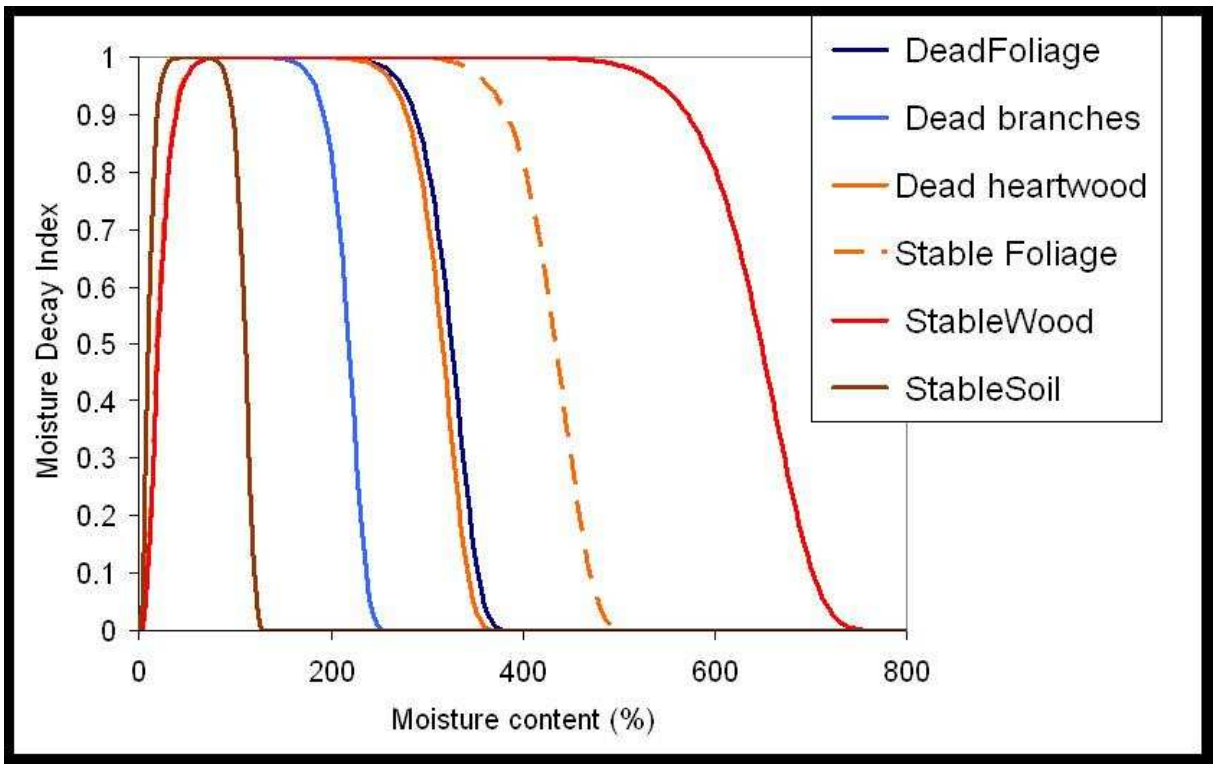

Figure 3-8. Relationship between moisture content and relative effect on decomposition rates of selected pools in the LANDCARB model.

The equation for the matric potential limitation (*Mon*MatricLimit) of each detrital pool or the stable soil pool for each month is:

$$MonPoolMatricLimit=(1-\exp[-IncreaseRate*(MonPoolMoist+MatricLag)])^{MatricShape}$$

where MatricShape is a dimensionless number that determines when the Matric limit is reduced to the point that decay can begin to occur. The MatricLag parameter is used to offset

the curve to the left or right. The IncreaseRate is the parameter determining the point at which the matric limitation ends. This parameter is determined from the minimum moisture content at which decay can occur:

IncreaseRate=3/MoistMin

The diffusion limitation (*Mon*DiffuseLimit) is designed to mimic the reduction in decomposition caused when the substrate becomes water saturated. Water saturation causes a reduction in oxygen diffusion reducing decomposition. This function remains at 1 until the maximum moisture content without diffusion limitations is reached. The function decreases to 0 when moisture content exceeds the maximum for decomposition to occur. This function is calculated for each detrital pool for each month:

$$MonPool\text{DiffuseLimit}=\exp[-(MonPool\text{Moist}/(\text{MoistMax}+\text{DiffuseLag}))^{\text{DiffuseShape}}]$$

where MoistMax is the maximum moisture content without diffusion limitations, DiffuseShape is a dimensionless number that determines the range of moisture contents where diffusion is not limiting, and DiffuseLag is a parameter used to shift the point when moisture begins to limit diffusion. These parameters are stored in the DcayParm.prm file.

The combined effect of matric and diffusion limitations for each dead pool or for the stable soil pool for each month is:

*MonPool*MoistDecayIndex=*MonPool*MatricLimit**MonPool*DiffuseLimit

TempDecayIndex Function.

This function determines the effect of temperature on the decomposition rate of the detrital pools (Figure 3-9). The response to temperature has two components. The first part is an increase in respiration rate with temperature following a Q10 type curve. For each dead pool and each month the value of the following equation will be solved

$$MonPool\text{TempIncrease}=(Pool\text{Q10}^{((Mon\text{Temp24}-10)/10)})$$

where the respiration rate of the layer at 10 C is assumed to be 1.0, and *Pool*Q10 is the rate respiration increases with a 10 C increase in temperature (see the DcayParm.prm file) and *Mon*Temp24 is the temperature of a given month.

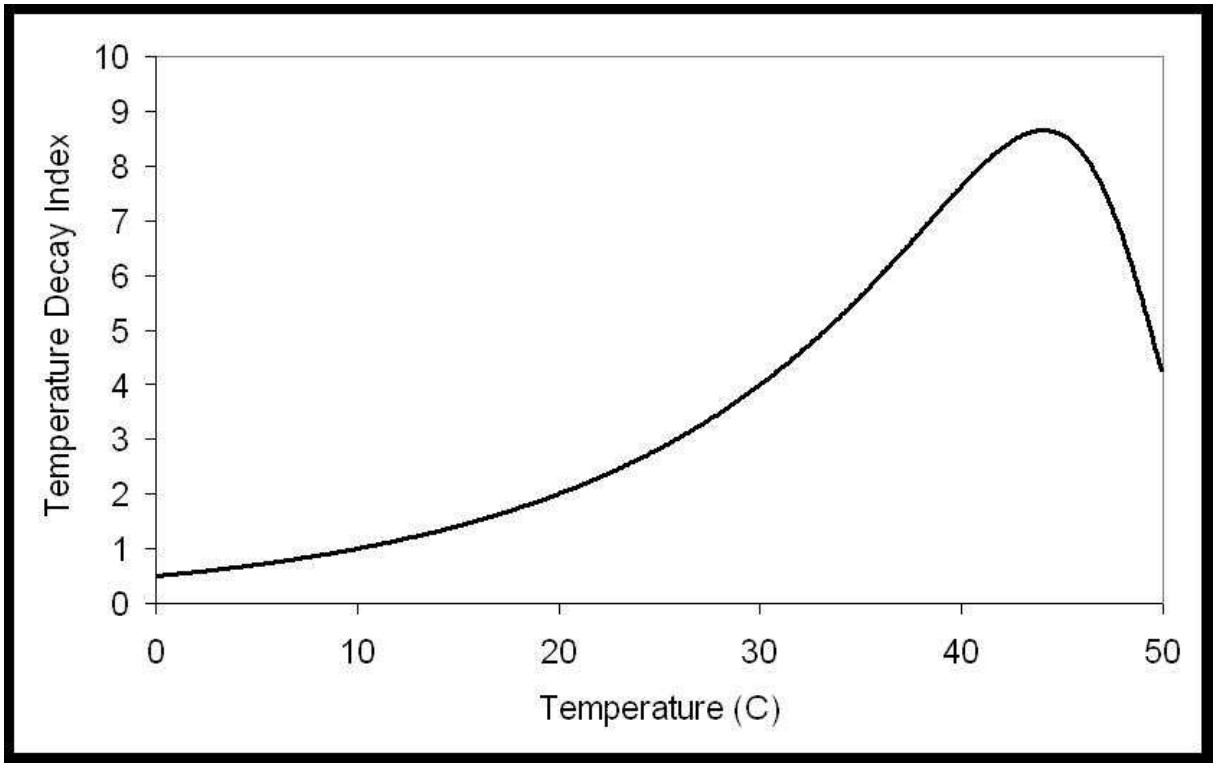

Figure 3-9. Relationship between temperature and relative effect on decomposition used in the LANDCARB model.

The second part of the temperature response simulates the effect of a lethal temperature limit that arrests decomposer activity. This equation is given by:

$$MonPool\text{TempLimit}=\text{Exp}[-(Mon\text{Temp24}/(Pool\text{TempOpt}+Pool\text{TempLag}))^{Pool\text{TempShape}}]$$

where *Pool*TempOpt is the optimum temperature for decomposition of a dead pool and *Pool*TempLag and TempShape are parameters that determine the shape of the response curve as determined from the DecayPool.prm file.

The combined effects of these effects for each dead pool for each month is given by *MonPool*TempDecayIndex:

$$MonPool\text{TempDecayIndex}=MonPool\text{TempIncrease}*MonPool\text{TempLimit}$$

AbioticDecayIndex Function.

This function calculates the combined effects of temperature and moisture on the decomposition rate of each detrital pool and the stable soil pool for each month.

For each dead pool or the stable soil pool the monthly abiotic decomposition index (*MonPool*AbioticIndex) is

*MonPool*AbioticIndex=*MonPool*MoistDecayIndex**MonPool*TempDecayIndex

The mean annual AbioticIndex (*Pool*AnnualAbioticIndex) for each dead pool or the stable soil pool is then used to control the decomposition rates in DECOMPOSE.

TempProdIndex Function.

This function determines the effect of temperature on net photosynthesis of each layer (Figure 3-10). The curve used to simulate this relationship is taken from Running and Coughlan (1988) and defines the mean daytime temperature (*Mon*TempDay; see TempConvert function above) response according to a minimum and maximum temperature compensation point (*Layer*TempMin and *Layer*TempMax) for each layer as defined in the Growth.prm file. If the mean daytime temperature exceeds either the minimum or maximum temperature compensation points of a layer, then the temperature production index (*Layer*TempProd) for a layer is set to zero. If the daytime temperature is within those limits then:

*MonLayer*TempProdIndex=(*Layer*TempMax-*Mon*TempDay)*(*Mon*TempDay-*Layer*TempMin)/(*Layer*TempMax-*Layer*TempOpt)*(*Layer*TempOpt-*Layer*TempMin)

where *MonLayer*TempProdIndex is a relative index of the response of each layer to a monthly daytime temperature (*Mon*TempDay).

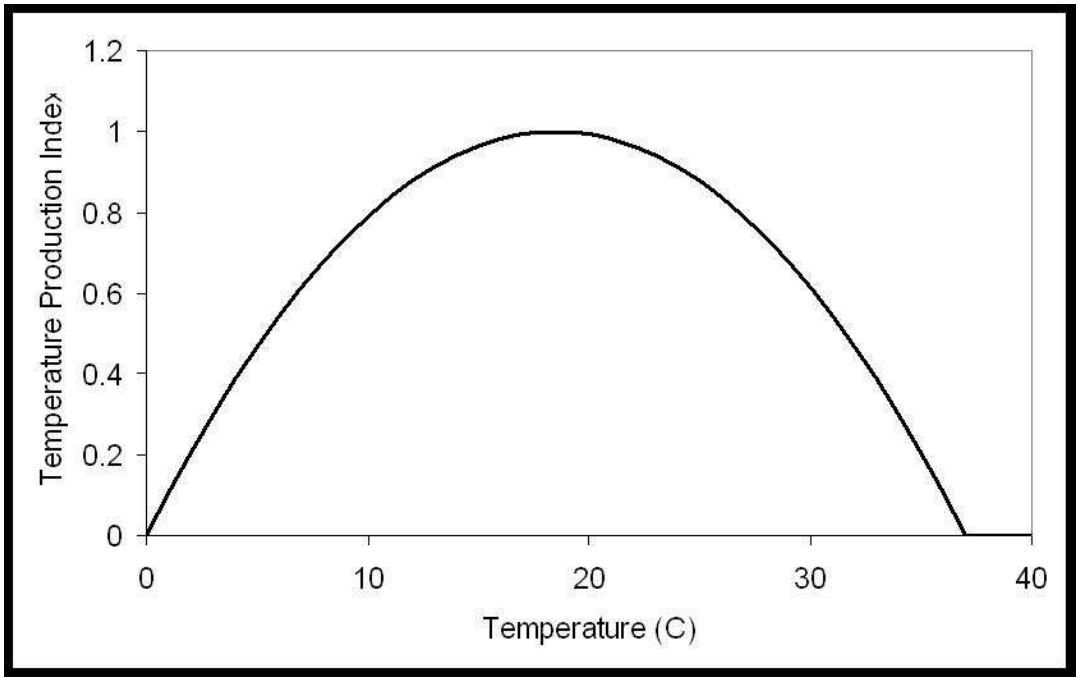

Figure 3-10. Relationship between temperature and relative photosynthetic rate used in the LANDCARB model.

The optimum temperature (*Layer*TempOpt) for a layer is defined as:

*Layer*TempOpt= (*Layer*TempMax-*Layer*TempMin)/2

MoistProdIndex Function.

This function determines the effect of soil moisture on the production of live biomass (Figure 3-11). We assume there is no effect on production of live biomass when either waterlogging occurs. In a future version we hope to implement a reduction in production for soils in which the water potential is less than -0.1 MP. This would account for a more realistic assumption that poor drainage and waterlogging reduces production. The current equation giving the waterlogging response is:

*Mon*WaterLoggingIndex=1

where *Mon*WaterLoggingIndex is the reduction of waterlogging on production.

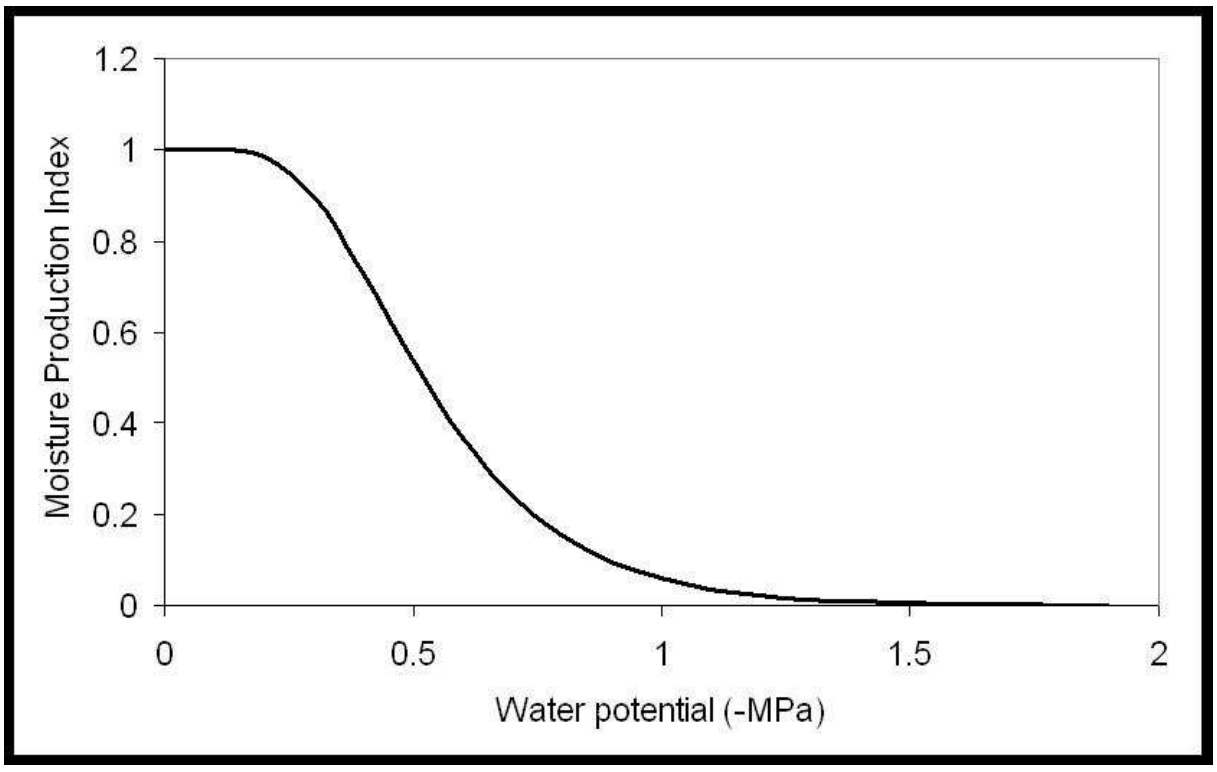

Figure 3-11. Relationship between soil water potential and relative photosynthetic rate used by the LANDCARB model.

When soil water potential (WaterPot) is below -0.3 MPa, the production rate decreases exponentially as a function of drought (Emmingham and Waring 1977). The equation describing this response is:

*Mon*DroughtIndex=1-(1-exp[-5**Mon*WaterPot])^9

where *Mon*DroughtIndex is the reduction in production caused by drought. The overall effect of water potential on production (*Mon*MoistProdIndex) is:

*Mon*MoistProdIndex=*Mon*WaterLoggingIndex**Mon*DroughtIndex

ProdIndex Function.

This function combines the monthly effects of moisture and temperature on the production rate of living biomass for each plant layer. In addition to indicating the response of production, it is also used to control the transpiration of the layers in the PET & Transpire functions. For each month the product of these two indices is computed:

*MonLayer*ProdIndex=*MonLayer*TempProdIndex**Mon*MoistProdIndex

The mean annual production index (Annual*Layer*ProdIndex) is then computed and used by the GROWTH module.

4-SOIL TEXTURE

This module is used to estimate the maximum amount of water that can be stored in a soil profile for the stand grid cells in a soil zone It is invoked once at the beginning of each simulation. This estimate is based on the soil texture class, the depth of the soil in cm, and the percentage of the soil with fragments greater than 2 mm in size that is specified in the SoilZones.prm file. Landscapes are divided into different soil types, the location of which is described in the SoilZones.asc grid file (Figure 3-1).

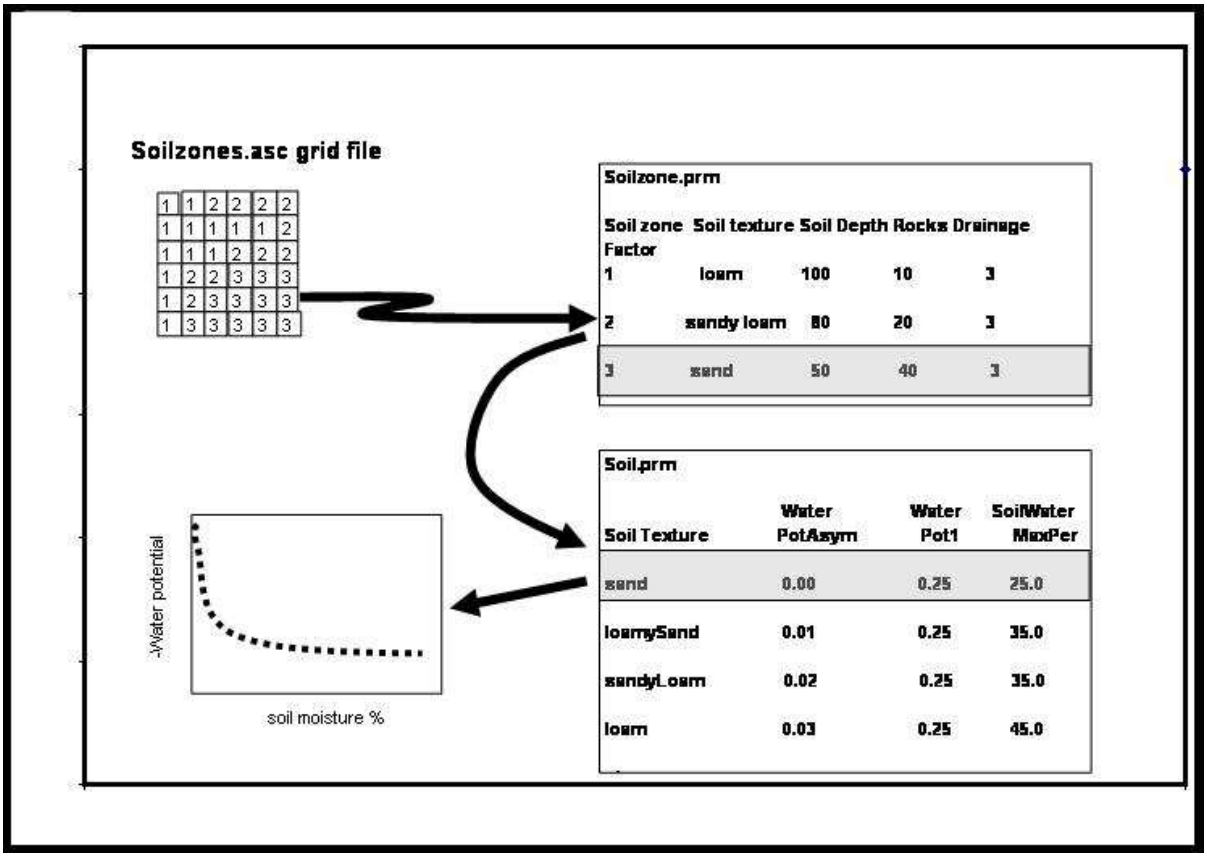

Figure 3-1. Description of soil properties used by the LANDCARB model.

SoilTexture Function.

This function determines the maximum amount of water storage in a soil based upon the soil texture, depth, and rockiness. The output of this function is sent to the CLIMATE module and used in the WaterPot and WaterStores functions.

This function first determines the fraction of the soil that can store water between field capacity and the wilting point (SoilWaterMaxPer) based on the soil texture class specified by the SoilZones.prm file for each soil type. SoilWaterMaxPer is set for each soil texture class contained within the Soil.prm file.

The volume of rocks is used to decrease the overall water holding capacity. The fraction of the soil profile with fine soil (FineSoil) is calculated:

FineSoil=(100-Rocks)/100

where Rocks is the percentage of the soil with fragments greater than 2 mm diameter as specified in the SoilZones.prm file for each soil type.

Finally, the depth of soil that can store water (SoilWaterMax) in cm is calculated from the soil texture, rock content, and soil depth (in cm):

SoilWaterMax=SoilWaterMaxPer*SoilDepth*FineSoil

where SoilWaterMaxPer is the percent of fine soil that can store water from field capacity to wilting point, SoilDepth is the depth of the soil (in cm) as defined in the SoilZones.prm file for each soil type, and FineSoil is the fraction of the soil that is fragments less than 2 mm diameter.

5-NEIGHBOR

The purpose of this module is to simulate the interaction between trees in the cohorts and stand grid cells regarding light. The main spatial interaction between trees is therefore one of shading. The degree of shading is determined by the relative heights of trees in cohorts or stand grid cells and the distance between these entities. No attempt is made to realistically model the height distribution or profile of foliage in canopies in NEIGHBOR. Rather, it is assumed that foliage extends from the ground to the tree top and that foliage mass is evenly distributed over this height. While real foliage profiles are more complex, this simplifying assumption represents many common situations such as well stocked stands. It is most likely to have problems in situations where tree stocking is very low or variation in tree heights is high.

Neighbor shading effects are considered at three levels: 1) within cohorts, 2) between cohorts, and 3) between stand grid cells. **(the latter has not yet been implemented)**. For each the differences in tree height and distance determine the degree of shading. Since trees within cohorts are closer to each other than trees between cohorts and stand grid cells, the effects of shading decreases as one proceeds from within cohorts to between cohorts to between stand grid cells. There no attempt to divide light into indirect or diffuse radiation and direct radiation as is the case in STANDCARB. Light is only reduced for an entity that is shorter than the others.

The degree of blocking of solar radiation (i.e., light) inputs is based on the height difference and distances between the level in question and the adjacent entity in the same level. The height is estimated from the age of the tree layers in each cohort. When light is blocked, then the amount of light entering the given entity (tree layer, cohort, stand grid cell) is reduced. The term entities is used to refer to the locations at various levels of spatial detail (trees in cohorts, between cohorts, between stand grid cells).

Within Cohort Shading. The distance between trees within a cohort is set by the maximum width of tree crowns as specified in the simul.drv file. The difference in tree heights is a function of the age of upper and lower trees in cohorts. Until the TimeClose of the upper tree is reached, it is assumed that variation in tree heights is a function of the age structure of the upper trees. The height difference is that of the oldest versus the youngest upper tree divided by 4 to approximate a standard deviation. This assumes that the oldest and youngest trees approximate the 95% confidence interval. Once TimeClose is exceeded, the difference in tree heights is gradually increased to match that expected for an all-aged stand. It is assumed that the dying out of the upper tree leads to an all-aged structure in the lower trees. TimeClose and the extinction rate of the lower tree are used to create an age-class distribution. These ages are used to estimate the various heights present and a weighted standard deviation is then used as an index of tree height variation. A natural growth function is used to mimic this transistion, with the time required to make the transition determined by the TimeCloseWindow parameter of the upper trees.

Between Cohort Shading. The distance between cohorts is determined by the stand grid cell width and the number of patches that disturbances create. The greater the number of patches, then the shorter the distance between cohorts. The number of disturbance patches is defined in the HarvestClasses.prm and FireClasses.prm files. The stand grid cell width divided by the number of disturbance patches gives the distance between cohort patches. Note that the exact location of cohorts is not tracked in LANDCARB. While there is no information on arrangement, this approach does allow one to predict more shading as the number of disturbance patches increases. The height difference between cohort patches is calculated as the standard deviation of the mean cohort patch heights. This provides a population estimate of the differences in heights.

Between Stand Grid Cell Shading. The stand grid cell arrangement is a rectangular. This means that each stand grid cell has four immediate neighbors and four additional ones on the corners. The distance across a stand grid cell (GridCellWidth) represents the horizontal distance. The slope corrected area represented by each cell is therefore fixed. All stand grid cells are referenced using two numbers to indicate their position on the X and Y coordinates. Height differences between stand grid cells are dependent on the differences in mean tree height in the stand grid cells. Differences in height are calculated for adjacent stand grid cells.

TreeHeight Function. The height of the tree layer is determined by parameters described in the Growth.prm file and the age of the tree layer. The height of the crown in a cohort will be determined from the age of the tree layer (Figure 5-1). For upper tree the age reflects the age of the age-class of the cohort. For the lower tree, the age also reflects the age-class structure produced by knowing the TimeClose and the extinction rate of lower trees.

A Chapman-Richards equation is used to predict the height from the age since planting of the tree layer:

$$TreeHeight = HeightMax * (1 - exp [-HeightRate * Age])^{HeightShape}$$

where HeightMax is the maximum height of a species, HeightRate is the rate at which the maximum is reached, and HeightShape is the parameter that reduces height growth early in the life of a tree. These parameters are stored in the Growth.prm file.

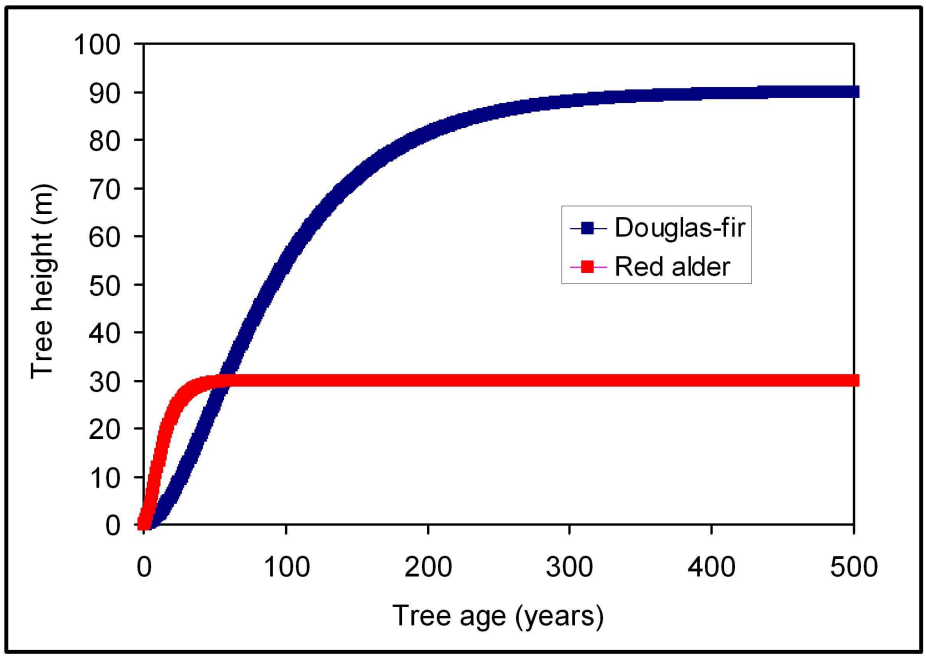

Figure 5-1. Examples of tree height as a function of cohort age.

Shading Function.

The purpose of this function is to determine the fraction of the sky that is obscured by surrounding entities (i.e., trees in cohorts, cohorts, stand grid cells). The angle (in radians) between the top of the tree layer in the entity of question and the adjacent entities is used to adjust the amount of light an entity receives (Figure 5-2). The term entities is used to refer to the locations at various levels of spatial detail (trees in cohorts, between cohorts, between stand grid cells). The angle is computed as:

ShadeAngle=arctan [(TotalHeightCell2-TotalHeightCell1)/CellDist]

where TotalHeightCell2 is the height of the tree layer in an adjacent entity, TotalHeightCell1 is the height of the tree layer in the entity in question and CellDist the distance from the center of one entity and another.

The angle of the sky open to light would therefore be:

OpenAngle=1.57 - ShadeAngle

For the within cohort level shading OpenAngle is rescaled to a fraction of the sky:

OpenFraction = OpenAngle /1.57

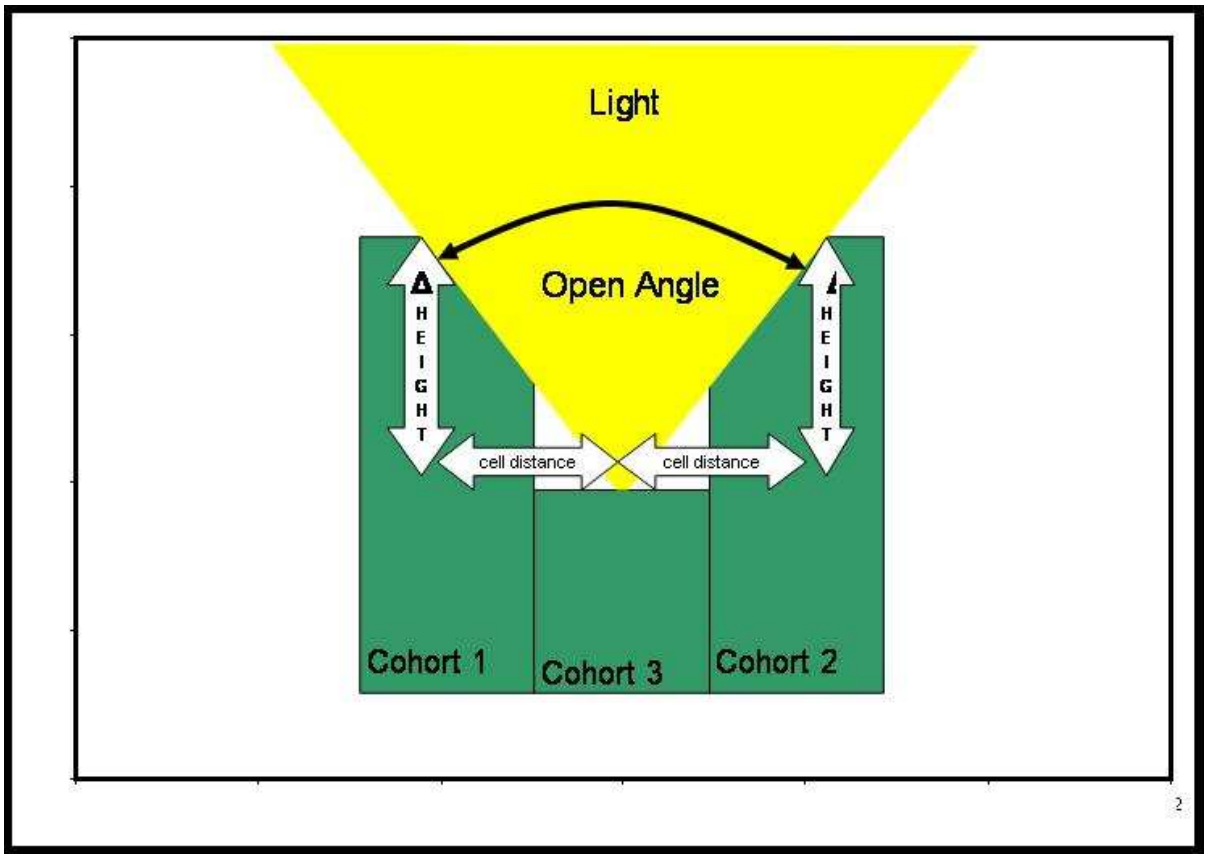

Figure 5-2. Relationship among cohort heights and amount of light available for growth.

For the between cohort level shading OpenAngle is similarly rescaled to a fraction of the sky:

OpenFraction = OpenAngle /1.57

For between stand grid cell shading the OpenAngle for the adjacent 8 stand grid cells are summed and divided by 12.5664 to estimate the fraction of the sky that is obscured by adjacent stand grid cells:

OpenFraction = Σ (OpenAngle /12.5664)

This number is then used to reduce the light that can enter an entity (i.e., tree in a cohort, cohort, or stand grid cell).

6-PLANT

This module plants the vegetation layers and selects the species of tree to be planted in a cohort. Once a cohort is created by a disturbance, each of four vegetation layers is planted and an age-structure is created as determined by the colonization rate of that vegetation (i.e., plant) layer. The species within the tree layers is also determined by PLANT and depends on the light requirements, climate (temperature and moisture), and local seed abundance.

The files directly required by this module are Estab.prm and TreeReg.prm.

PlantChort Function.

The purpose of this function is to plant the vegetation layers once a cohort has been formed. Cohorts are created by disturbances that kill all the vegetation in part or all of a stand grid cell. The proportion of standgrid cell area allocated to a cohort depends on the amount of a stand grid cell that is killed. If the entire stand grid cell is killed, then the next cohort potentially occupies 100% of the stand grid cell area. If half the stand grid cell is killed, then the next cohort occupies 50% of the stand grid cell area, and so on. When a life form or tree species is planted in a cohort, then the life form and species specific growth, mortality, and decomposition parameters are imported into the GROWTH, MORTALITY and DECOMPOSE modules for that cohort. This starts the processes of growth, mortality, and decomposition within the cohort.

Plant layers (i.e., herb, shrub, upper trees and lower trees) have unequal chances of colonizing a cohort. The probability a layer will be planted is defined in the Estab.prm file by a mean rate and a standard deviation of colonization. Upper trees can be colonized either naturally or artificially. It is assumed that herbs, shrubs, and lower trees colonize naturally. In addition, lower trees have a lag before they establish in part because upper trees must establish first, and in part because of dispersal. The establishment rate is used to create an age-class structure based on the upper layer in a cohort assuming the area left for colonization is a negative exponential function (Figure 6-1):

Remaining Area for Colonization= exp[-ColonizationRate* TimeSinceCohortFormation]

The area of each age-class is solved by substracting the remaining area for colonization of the current age-class from that of the preceeding age-class. Note that with this function the different age-classes occupy different proportion of the cohort space. We assume that herb and shrub layers will colonize faster than tree layers and therefore use the upper tree layer ability to colonize to determine if additional cohorts will be required. Therefore the upper layer has a limited period in which to colonize the space allocated to a cohort; if the colonization rate is too slow to colonize the entire cohort area in the allotted time, then a new cohort is established on the remaining area. The assumption is that age-classes within cohorts are roughly similar in terms size and that if too much time elapses this would no longer be the case. If a additional cohorts are required to fill the space created by the disturbance, then age-class distributions for all the plant layers are created. However, the tree layers are not

actually initialized until the lags associated with their establishment are exceeded. This allows the non-tree layers to grow unimpeded until trees actually establish in the cohort. For lower trees the age-class structure is created at the same time as the upper trees, but the lower tree species is not determined until this establishment lag time is exceeded.

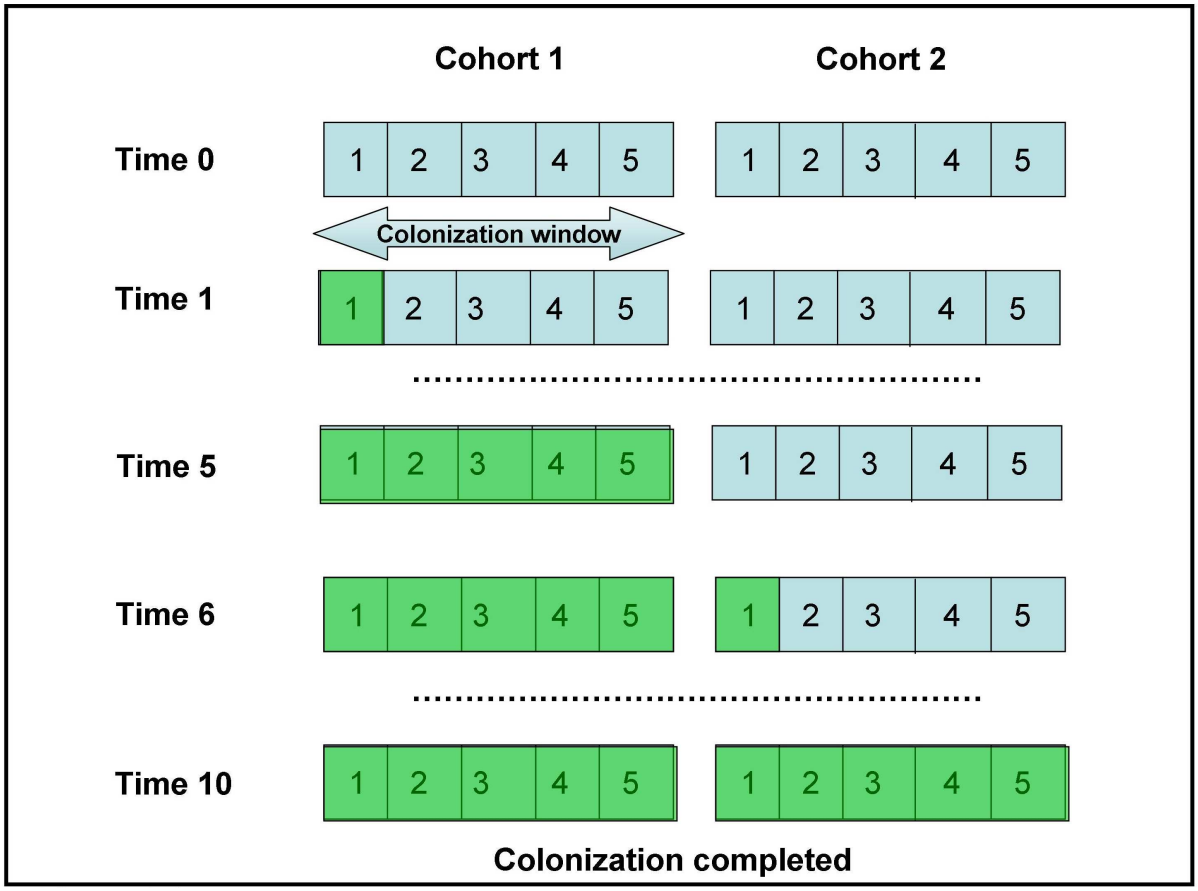

Figure 6-1. Hypothetical example of how cohorts are colonized by trees. The colonization window is the time limit allowed for a cohort to fill. If it is exceeded then a new cohort is established. This keeps trees within a cohort relatively similar in size.

When a layer planted, then a message is sent to GROWTH module to make the mass of foliage for that layer a small positive number. Once the foliage mass is added the layer begins to grow and add biomass. Adjusting InitialFoliageMass (the value is specified in the GrowLayer.prm file) determines the length of time lag required for the layer to begin growing at a significant rate. Setting this parameter low increases the lag time, whereas increasing it decreases the lag time. Note that for lower trees and upper trees in additional cohorts, the initial foliage mass is not changed to a small positive number until their establishment lag times are exceeded.

PlantTree Function.

In LANDCARB trees are treated on the species level. The PlantTree function selects which species can be planted. LANDCARB does not consider all the factors that control tree

establishment. Rather the intent of the PlantTree function is to plant trees in proportion to their abundance and so as to mimic the presence of an early and late successional tree species.

The first step is to determine the local abundance of tree species. This is a function of the tree species present in cohorts within the stand grid cell being examined as well as those present in the adjacent stand grid cells. The user defines a seed source area to be considered which determines how many stand grid cells are considered. The user also defines the degree a species can produce seeds in the SeedSource.prm file in various source zones. Species are randomly selected for planting based on their abundance, however, to be planted a tree species must fall within limits of light, temperature, and soil moisture as defined in the TreeReg.prm file.

To select a tree species the PlantTree function first eliminates all the species that do not fall within the light, temperature, and moisture limits (Figure 6-2). It then determines the proportion of seeds available based on the results of the Seed Dispersal Module. Tree species are then selected in proportion to their abundance.

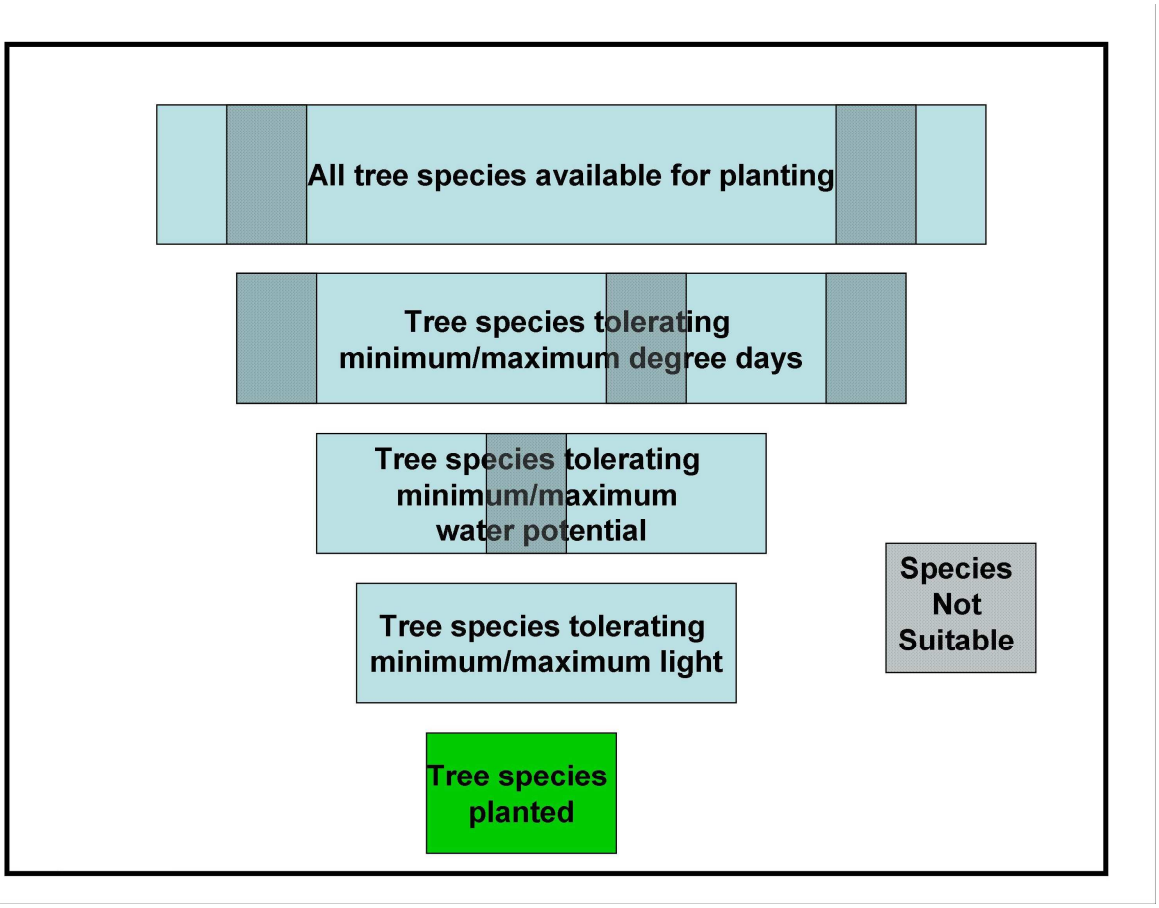

Figure 6-2. Conceptual diagram of how all available tree species are filtered to those actually planted in a cohort.

Temperature Limits. A species may not be planted if the thermal environment of the site exceeds the limits of a tree species. If the degree day values (DDays) of a site determined from the DegreeDays function of the CLIMATE module are equal to or below the degree day minimum (DDayMin), or equal to or exceed the degree day maximum (DDayMax) for a species, then its probability of establishment is set to zero (DDayLimit). If the DDays are within the limits then DDayLimit is set to one.

Soil Moisture Limits. A species of tree may also not be planted if the site is too dry or too wet. For these calculation we use the absolute value of soil water potential as a indicator of soil moisture. **(Note: when using soil water potential the wettest soil has the lowest water potential and the driest soil has the highest water potential).** If the yearly maximum soil water potential (MaxMonWaterPot) calculated for a stand grid cell in the WaterPot function of the CLIMATE module is equal to or lower than the species minimum (TreeSoilMax), then the value of SoilLimit is set to zero. If the soil water potential exceeds the species minimum (TreeSoilMin) for more than a specified number of months, then SoilLimit is also set to zero. If the site soil water potential is within these limits, then SoilLimit is set to one.

Light Limits. A species of tree may not be able to establish within a cell if there is too much or too little light. In the former case, light *per se* may not be the limiting factor. Excessive heating or drying may be the actual mechanism involved. These problems are highly correlated, however, to high light levels. Minimum light levels are related to the species shade tolerance and light compensation point.

The light value used to determine whether a tree species can establish depends upon whether it is being planted as an upper tree or as a lower tree layer. For upper trees the value of light that is considered is the amount entering a cell (UpperTreeLightIn). This may fall below full sunlight if adjacent cohorts and stand grid cells contain taller trees that shade the cohort being considered (see NEIGHBOR module). For lower trees the value of light used is the amount of light leaving the upper tree layer (LowerTreeLightIn) at the time the lower tree establishment lags are exceeded. If the light value considered in either case is equal to or within the limits during the time step a species is being selected, then the value of *Layer*LightLimit is set to 1. If the light exceeds the maximum and minimum light limits, then the value of *Layer*LightLimit is set to zero.

7-DIEOUT

This module determines when and the rate a upper canopy layer in a cohort will die out. The rational is that the upper canopy layer has a finite life-span and has a limited ability to expand horizontally after a certain age. Therefore after a certain age mortality and disturbance will decrease the area the upper tree layer can occupy. When the area occupied by the upper tree layer decreases, the light reaching the lower tree layer increases (Figure 7-1). The parameters used to define when trees die out (AgeMax, TimeClose, TimeCloseWindow) are contained within the Mort.prm file.

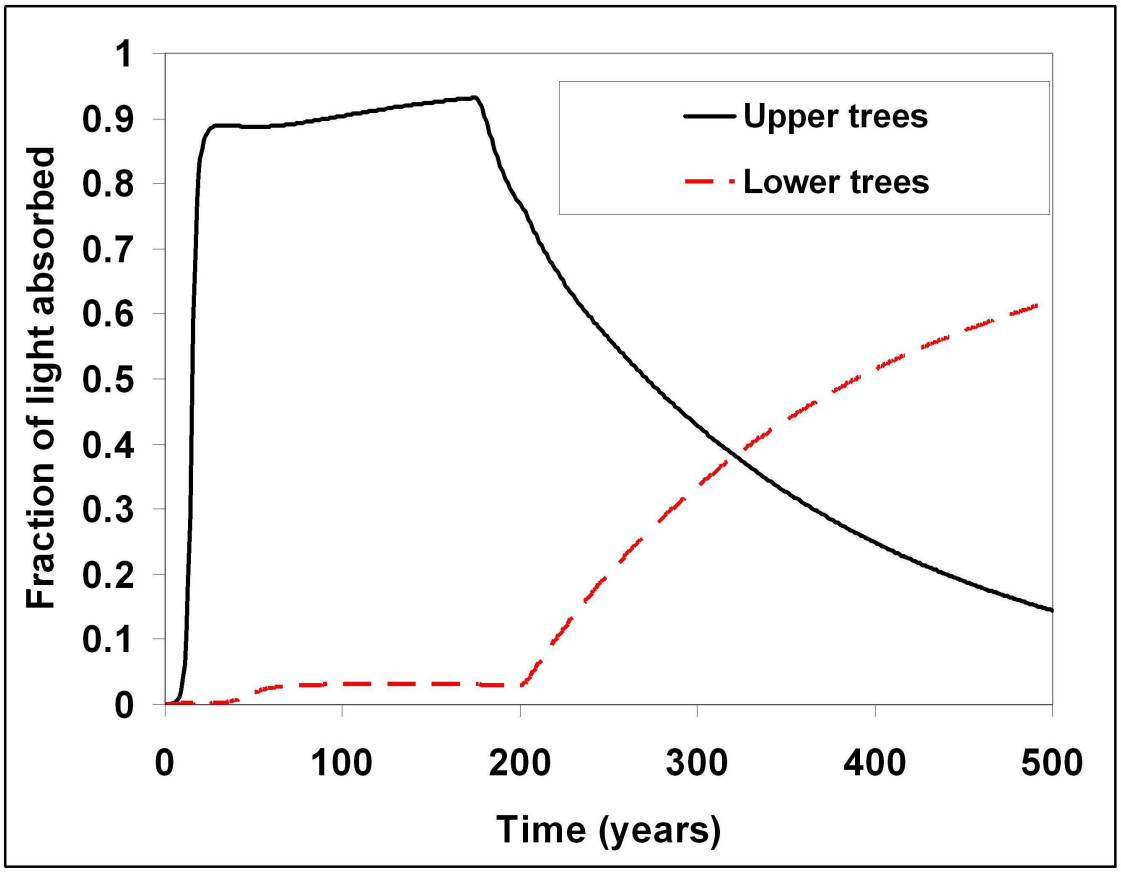

Figure 7-1. Example of how the DIEOUT model removes upper trees once TimeClose has been reached for a species. At TimeClose the horizontal spread of individual trees reaches a maximum and therefore mortality leaves openings for lower trees to receive more light.

TimeThere Function.

This function calculates the time the upper tree layer has occupied a cohort. Because not all the trees are planted in simulation year 0, there is a difference between the simulation time and the time a tree species occupies a cohort. To determine if an upper canopy layer has a chance of dying out, the time the species is present in the cohort must be calculated:

TimeThere=Time-TimePlant

where time is the simulation Time in years, TimePlant is the simulation time that an upper tree species was planted in a cohort, and TimeThere is the number of years an upper species has occupied the upper canopy layer of a cohort.

DieOut Function.

This function calculates the rate an upper tree layer dies out, freeing up space and light for the lower tree layer. The area occupied by an upper tree layer is a function of the time the species has occupied a cell. If the time since a species was planted in a cell (TimeThere) is less than TimeClose (defined in the Mort.prm file) then :

UpperTreeArea= 1.0

If the time since a species was planted in a cell (TimeThere) is greater than or equal to TimeClose, then the area occupied by the upper tree layer decreases as follows:

UpperTreeArea= exp[-ExtRate*(Time-TimeClose-TimeCloseWindow)]

where ExtRate is the annual probability that a species will die, and TimeClose Window is the period of time it takes for all upper trees to reach their maximum horizontal extent once TimeClose is reached. The rate upper trees decline is calculated from the maximum age of the species (AgeMax) and the time required for a single tree to dominate a plot (TimeClose):

ExtRate=4.61/(AgeMax-TimeClose)

As upper trees lose area, they receive less light and this limits their growth. Light not used by the upper tree layers is passed to the lower tree layer and as that light increases the upper trees become more and more dominant.

It should be noted that once upper tree pass the time they can not expand horizontally, other forms of mortality such as harvest and fire will also cause the upper tree layer to lose area.

8-GROWTH

The purpose of this module is to calculate the mass of seven live pools of carbon: 1) foliage, 2) fine roots, 3) branches, 4) sapwood, 5) heartwood, 6) heartrot, and 7) coarse roots. To avoid confusion with the corresponding detrital pools, we have referred to these live pools as **parts**. The functions in GROWTH are invoked each year for each cohort before any disturbance occurs.

This module is divided into eleven functions which perform specific tasks. These include: 1) Light Absorption and Foliage, 2) Foliage Age-class Adjustment 3) Hydraulics, 4) Allocation, 5) Respiration, 6) Heartwood Formation, 7) Heartrot, 8) Mortality, 9) Prune, 10) Live Stores, and 11) Volume (note not yet implemented). Each of these functions is invoked for each plant layer present in a cohort.

The parts present depend on the plant layers present in a cohort. Herbs are assumed to have leaves and fine roots only. Shrubs have leaves, fine roots, branches, sapwood, and coarse roots. Trees have leaves, fine roots, branches, sapwood, heartwood, heartrot, and coarse roots. Boles are divided into three pools: sapwood, heartwood, and heartrot. Sapwood and heart-rots represents respiring tissue, whereas heartwood represents non-respiring tissue. Heartrot represents a part of the bole that is decomposing. Splitting the bole into these parts also allows one to have the decomposition of dead wood as a function of decay-resistance of the heartwood of the tree species growing in a plot.

The mass of parts calculated in this module are in turn used by CLIMATE, MORTALITY, DECOMPOSE, HARVEST, PERSCRIBED FIRE, and WILDFIRE.

The files directly used by the GROWTH module to define parameters includes Growth.prm and GrowLayer.prm.

Light Absorption and Foliage Function

This function determines the growth of the foliage layers and the amount of light absorbed by them (Figure 8-1). Light is expressed in relative terms as a percentage of full sunlight. We assume that taller layers have a competitive advantage over shorter stature layers; if taller layers are present they will absorb light before underlying layers. Therefore, in this model smaller stature layers do not have the ability to exclude potentially taller layers such as trees. This behavior can, however, be simulated by reducing the colonization rate of the upper and lower tree layers in the PLANT module (see Estab.prm file). In addition to being reduced by taller layers by shading from above, light coming into a plant layer can be reduced by shading from trees within cohorts, as well as surrounding cohorts and stand grid cells (see NEIGHBOR).

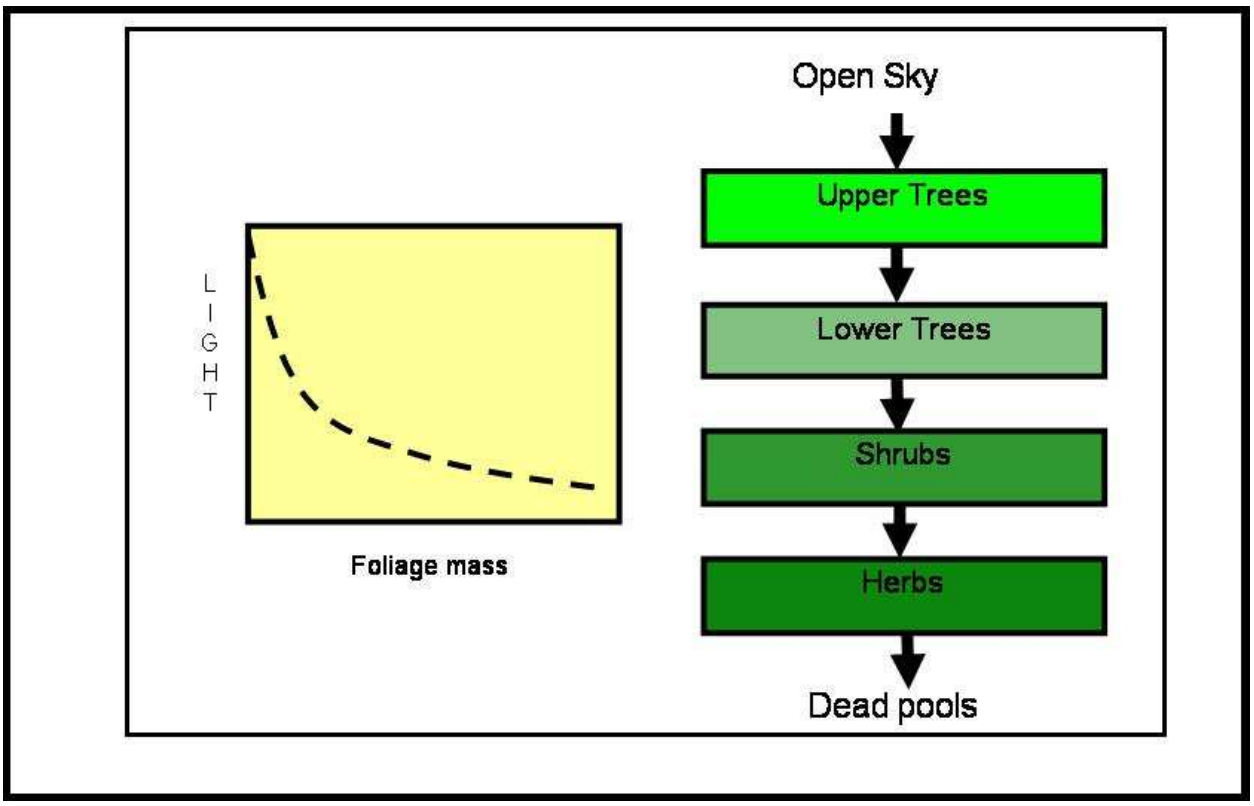

Figure 8-1. Movement of light through the LANDCARB model and removal of light by foliage.

The foliage mass of a layer (*Layer*Foliage*)* will not begin to grow until a small mass of foliage is added to the foliage part of a layer in a cohort by PLANT. This allows the mass of foliage to increase at a rate dependent upon the amount of remaining light. Before a layer is planted the following condition is present:

The *Layer*FoliageProdRate=0

and

*Layer*Foliage=0

As soon as a layer is planted these variables are set:

FoliageProdEffic=1

and

Foliage=InitialFoliageMass

where InitialFoliageMass is the initial mass of foliage that is planted. InitialFoliageMass can be varied (See the GrowParm.prm file) to introduce a lag in the time required for a layer to grow significantly. By reducing this parameter one increases the lag in the growth of a layer. Foliage production rate is an index that indicates the relative ability of foliage to produce more foliage. When this variable is 1, foliage increases at the maximum rate. When this variable is 0, foliage does not increase and when it is negative (i.e., an overlying layer establishes and absorbs too much light) the foliage mass decreases.

Layers are able to increase their foliage mass until the light compensation point (LightCompPoint) for that layer or species of tree is reached. As overlying layers can absorb light, the foliage growth of underlying layers can be far below that expected for full sunlight. This approach also allows the foliage of the underlying layers to change in response to an overlying layer dying out or to an overlying layer establishing.

The first step to calculate the rate that foliage increases is to convert LightCompPoint from a percentage to a proportion:

*Layer*LightCompPoint=*Layer*LightCompPoint/100

The next step is to calculate the potential maximum of light (*Layer*MaxLightAbsorb) that can be removed by each layer as a function of the amount of light that comes into the top of each layer and the light compensation point:

*Layer*MaxLightAbsorb=*Layer*LightIn-*Layer*LightCompPoint

where *Layer*LightIn is the relative fraction of full sunlight that reaches the top of a given layer. *Layer*MaxLightAbsorb is a dynamic variable and is calculated each year because the amount of light removed by overlying layers or adjacent trees, cohorts, and stand grid cells changes over time. To avoid possible cases where *Layer*MaxLightAbsorb could go negative (which happens if the light compensation point is larger than the light remaining from an overlying layer) *Layer*MaxLightAbsorb is restricted to be greater than 0.

The amount of light (Light) remaining at base of the foliage of each layer is a function of the mass of foliage of that layer:

*Layer*LightOut=*Layer*LightIn*exp[-*Layer*LightExtCoeff**Layer*Foliage]

where *Layer*LightIn is the light passed to an underlying layer and *Layer*LightExtCoeff is the light extinction coefficient for a layer. For trees, the latter parameter is a function of the species present in a cell. The light coming into an underlying layer equals the light passing through the overlying layer. The layers are set up so that the upper tree layer absorbs light first, what is left over is "passed" along to the lower tree layer, and that "passes" along what is left over to the shrub layer, and finally the shrub layer passes along what ever is left over to the herb layer. If the foliage mass of a layer is zero then no light is absorbed.

The next step is to calculate the light absorbed (*Layer*LightAbsorbed) by a layer:

*Layer*LightAbsorbed=*Layer*LightIn-*Layer*LightOut

The foliage production efficiency (*Layer*FoliageProdEffic) of a layer is assumed to decrease as the amount of light removed increases:

$$Layer\text{FoliageProdEffic} = 1-(Layer\text{LightAbsorbed}/Layer\text{MaxLightAbsorb})^2$$

This function implies that as the amount of light removed by a layer increases, its ability to increase its foliage mass decreases. When the light absorbed equals the maximum that can be absorbed then the foliage production efficiency equals 0. If the light absorbed exceeds the maximum then the leaves die, that is FoliageProdEffic is negative.

The absolute foliage production rate (*Layer*FoliageProdRate) of a layer is a function of the foliage production efficiency (*Layer*FoliageProdEffic) and the maximum absolute rate of foliage increase (*Layer*FoliageProdRateMax) in full sunlight (as defined in the Growth.prm file):

*Layer*FoliageProdRate=*Layer*FoliageProdEffic***Layer*FoliageProdRateMax

The rate that foliage mass of each layer increases (*Layer*FoliageAlloc) is:

*Layer*FoliageAlloc=*Layer*FoliageProdRate***Layer*Foliage

where *Layer*Foliage is the mass of foliage in a plant layer.

Foliage Age-class Adjustment.

In LANDCARB a plant layer need not colonize a cohort within one year. This means that some parts of the cohort are older (or younger) than others. To account for this fact the foliage mass of the layer in a cohort is adjusted to reflect all the age-classes that are present. Rather than simulate the foliage mass of each age-class, LANDCARB assumes that foliage of each age-class goes through a similar progression. That is the foliage of the second age-class is similar to that of the first age-class except that it is lagged by one year. The foliage of the third age-class is similar to that of the second, but lagged by one year, and so on. The foliage mass of each age-class is weighted by its area to estimate the total foliage mass of all the age-classes of a layer in a cohort. This approximation speeds up computations because only the mass of one age-class in a cohort is actually simulated. The assumption that the foliage of one age-class is similar to another is only valid when the age-classes represented do not span too large a time period. This is why cohorts are given a fixed time in which to establish. If that time is exceeded, then a new cohort is formed.

Hydraulics Function.

The hydraulics function accounts for the fact that as plants increase in height the efficiency of leaves to photosynthesize decreases (Ryan and Yoder 1994, Figure 8-2). This leads to a decrease

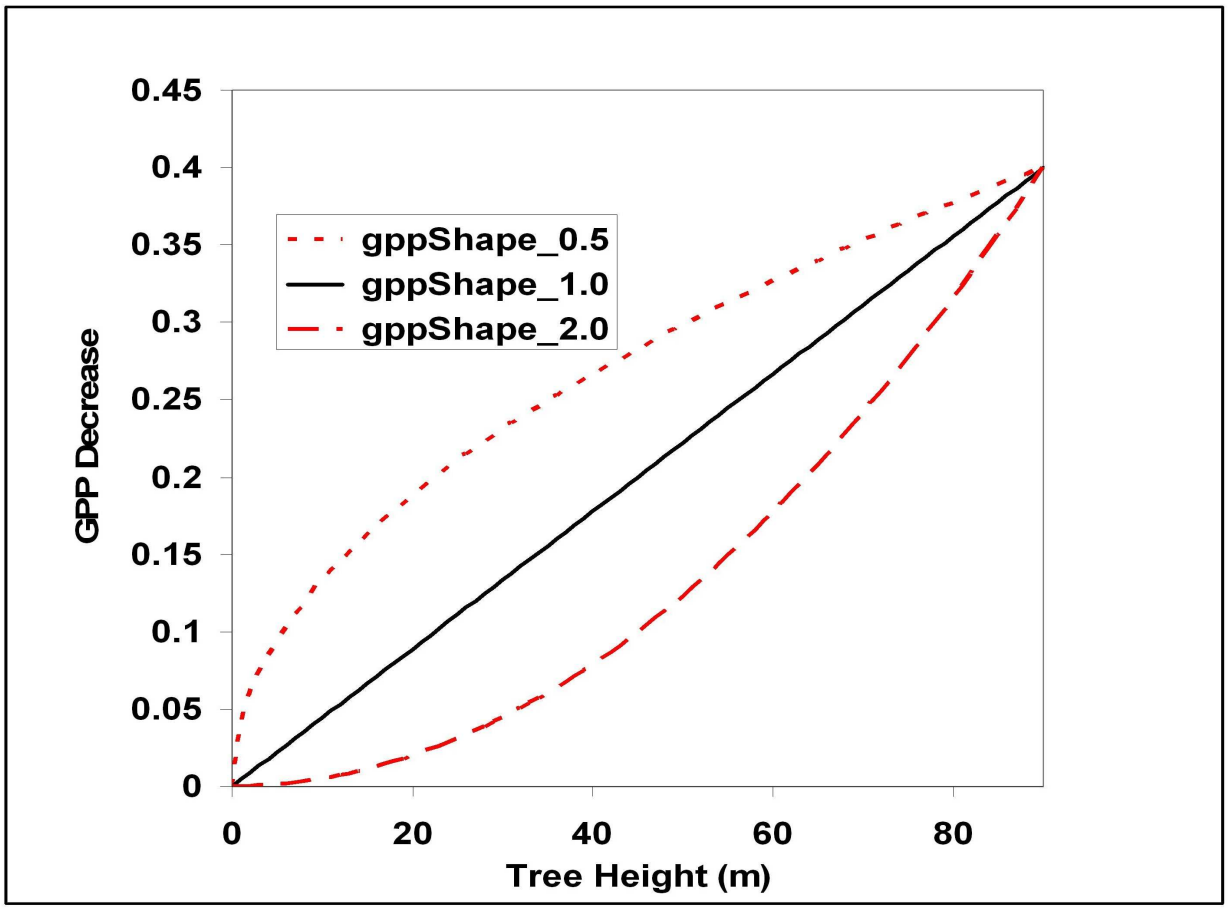

Figure 8-2. Reduction in GPP associated with tree height for different values of the GPPShape parameter. The maximum reduction in this example was assumed to be 40%.

in Gross Primary Production (GPP) as trees age. Although this is still a working hypothesis, we have included it in this version of the model to test its possible effects on carbon stores. This effect is implemented for upper and lower trees. The first step is to determine the coefficient (GPPDecreaseRate) describing the decline in GPP as height increases. This based on the maximum decrease in GPP (GPPDecreaseMax) that occurs at maximum tree height (HeightMax):

$$GPPDecreaseRate = GPPDecreaseMax/100 * HeightMax^{GPPShape}$$

where GPPShape is a constant that determines if the decrease in GPP with height is linear, or quadratic in form. The predicted decrease for a given tree height (GPPDecrease) is given by:

$$GPPDecrease = GPPDecreaseRate * Height^{GPPShape}$$

where Height is the current height (m) of the upper or lower tree in a cohort as calculated in the NEIGHBOR module.

Allocation Function.

This function allocates production by the foliage to the following plant parts: fine roots, sapwood, branches, and coarse roots. There are several assumptions used in the calculation of fine roots, sapwood, branch, fine root, and coarse root production. The first is that production of these parts is proportional to the mass of foliage of each layer. The second is that the proportions of allocation to bole (i.e., sapwood and heartwood), branches, and coarse roots are fixed. The latter assumption is based on the idea that these structural elements need to be balanced to function properly. Although the allocation of production to these parts is fixed, this does not mean that the portions of parts is constant. This is because loss of parts, as calculated by MORTALITY, is a function of the amount of light absorbed by a cell. Therefore, cohorts with less light absorbed (and therefore less competition) will have proportionally more branches, and coarse roots than those where the maximum amount of light has been absorbed.

We assume that there is a fixed ratio (*Layer*FineRootAllocationRatio) between the allocation to fine roots and foliage mass. This ratio is life-form specific (herbs, shrubs and trees) and defined in the Growth.prm file. This assumption implies that the energy and nutrient gathering portions of plants are in balance. This ratio is assumed to be highest for herbs, intermediate for shrubs, and lowest for trees giving the highest allocation of biomass below ground for herbaceous plants and lowest for trees. The allocation to fine roots is calculated as:

*Layer*FineRootAlloc=*Layer*FineRootAllocationRatio**Layer*Foliage

The allocation to woody plant parts can be determined by two methods. The first is based on a calibration to yield curves. This will set the growth rate so that the wood volume matches that of a specified level of productivity or site index for a selected species. If this option is specified in the Simul.drv file, then the variable *Layer*GrowthRate is set to value referred to in the SiteIndex.prm file. *Layer*GrowthRate can be thought of as the ability of foliage to form other tissues.

The second method used to determine the allocation to woody parts is to base the growth rate on climatic indices calculated in CLIMATE. If this option is specified in Simul.drv, then:

*Layer*GrowthRate=*Layer*AnnualProdIndex**Layer*GrowthEffic

where *Layer*AnnualProdIndex is the effect of temperature and moisture on growth for a layer and *Layer*GrowthEffic is the growth efficiency for a layer as specified in the GrowParm.prm file.

Regardless of the method used to determine the allocation to woody parts, the mass of production allocated from foliage to sapwood is:

*Layer*SapWoodAlloc=(1-GPPDecrease)**Layer*GrowthRate**Layer*Foliage

where GPPDecrease is the decrease in GPP due to hydraulic limitations (set equal to 0 for all non-tree layers), *Layer*SapWoodAlloc is the mass of sapwood produced by a layer, *Layer*Foliage is the mass of foliage of a layer, and *Layer*GrowthRate is the ratio of wood mass produced to foliage mass. This relationship makes sapwood production reach a maximum when foliage mass is at a maximum. If foliage mass is reduced by thinning or shading then the rate of sapwood production will also be reduced until foliage is regrown.

The amount of production allocated to branches from foliage for trees and shrubs (*Layer*BranchAlloc) is equal to a fixed proportion of the rate of sapwood production for that layer. The parameter *Layer*BranchBoleRatio defines the ratio of branch to sapwood production. This parameter is set to give the proportions of a tree greater than 50 cm diameter at breast height as solved by biomass equations (Means et al. 1994). The mass of branches produced for a layer is therefore:

*Layer*BranchAlloc=*Layer*BranchBoleRatio**Layer*SapWoodAlloc

The mass of production allocated to coarse roots from foliage (*Layer*CoarseRootAlloc) for the tree and shrub layers is calculated in manner similar to branches:

*Layer*CoarseRootAlloc=*Layer*CoarseRootBoleRatio**Layer*SapWoodAlloc

where *Layer*CoarseRootBoleRatio is the ratio of coarse root to sapwood production of a layer as defined in the Growth.prm file.

Respiration Function.

The purpose of this function is to estimate the respiration of the plant parts for each layer. Foliage, fine roots, branches, sapwood, heart-rot, and coarse roots all are capable of respiring. Heartwood is not capable of respiring. For fine roots, branches, sapwood, heart-rot, and coarse roots losses from respiration are used to calculate the net change of those parts in the Live Stores function (see below). The respiration losses of all plant parts except heartwood are estimated from their mass:

*LayerPart*Resp=*LayerPart*RespRate**Layer*Part.

where *LayerPart*Resp is the mass of production that is respired by a plant part, *LayerPart*RespRate is the rate as a proportion of each part, and *Layer*Part is the mass of each part for a layer the previous time step. *LayerPart*RespRate is calculated as a function of the species and the mean annual ambient temperature (MeanAnnualTemp) for the site (see MeanAnnualTemp function in the CLIMATE module). For all plant parts the proportion of mass respired each year is:

*LayerPart*RespRate=*Layer*Resp*Part*10 *Q10*Part*^((MeanAnnualTemp-10)/10)

where *Layer*Resp*Part*10 is the respiration rate of the plant part at 10 C, Q10*Part* is the rate respiration increases with a 10 C increase in temperature and MeanAnnualTemp is the mean annual temperature (Figure 8-3).

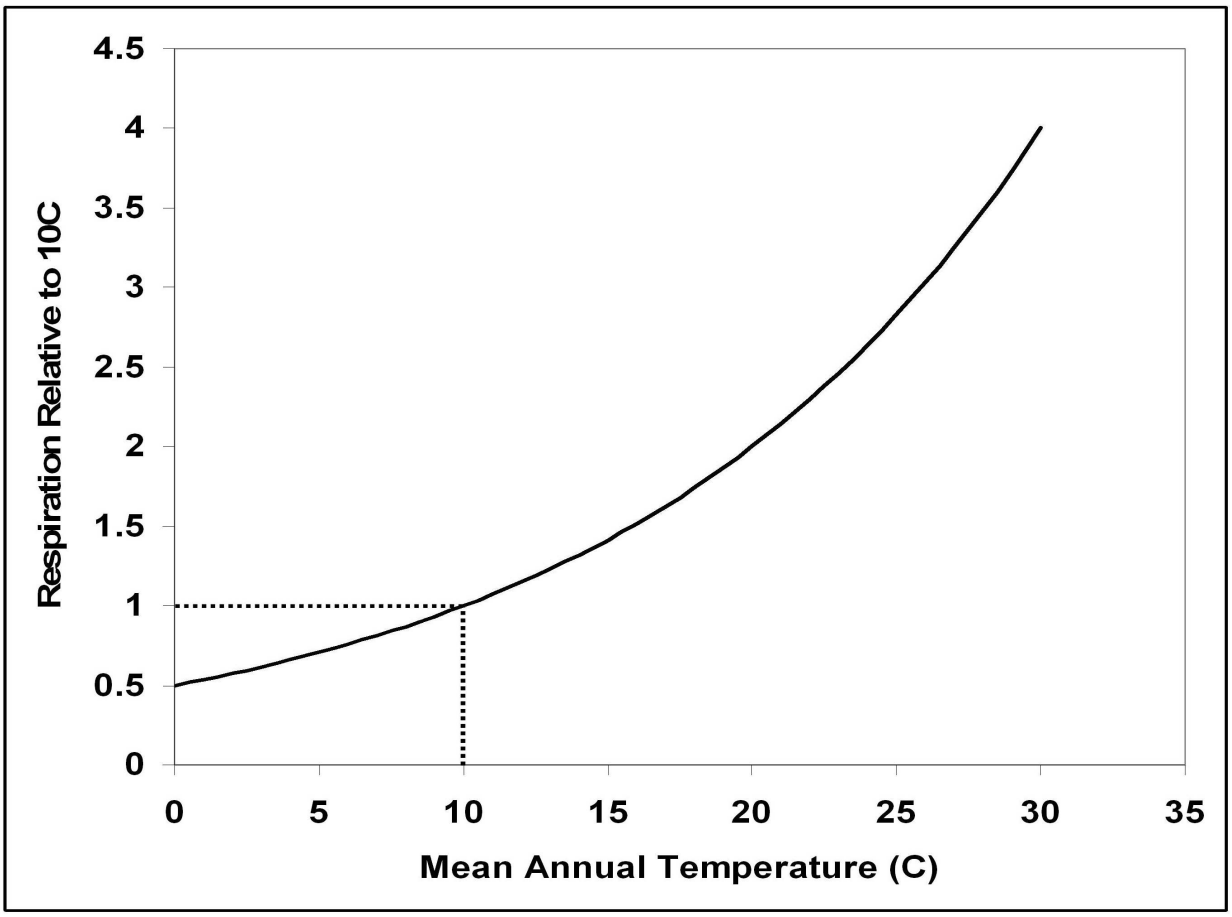

Figure 8-3. Response of live plant part respiration to mean annual temperature used in the LANDCARB model.

For foliage, fine roots, branches, and coarse roots the fraction that is alive is constant among species and layers. In the case of sapwood, adjustments are made to *Layer*RespSapwood10 to reflect the fact that tree species have differing proportions of the sapwood that is alive. The respiration rate for sapwood contained in the GrowLayer.prm file is based on a living sapwood fraction of 5%. The rates used are based on respiration of lodgepole pine and Engelmann spruce (Ryan 1990). This base rate is adjusted by:

*Layer*RespSapwood10=*Layer*RespSapwood10*(*Layer*SapLive/5)

where *Layer*SapLive is the percentage of the sapwood of a layer that is alive. This parameter is stored in the Growth.prm file and is tree species specific based on the relative differences in sapwood core respiration.

Heartwood Formation Function.

This function calculates the rate that heartwood is formed from sapwood for the tree layers (Figure 8-4). The mass transferred from sapwood to heartwood (*Layer*HeartWoodAlloc) for each tree layer is determined by the rate of heartwood formation (*Layer*RateHeartWoodForm) and the mass of sapwood (*Layer*SapWood) for the previous time step:

*Layer*HeartWoodAlloc=*Layer*RateHeartWoodForm***Layer*SapWood

*Layer*RateHeartWoodForm is parameterized so that the proportion of boles in sapwood matches the values in mature trees of the various tree species.

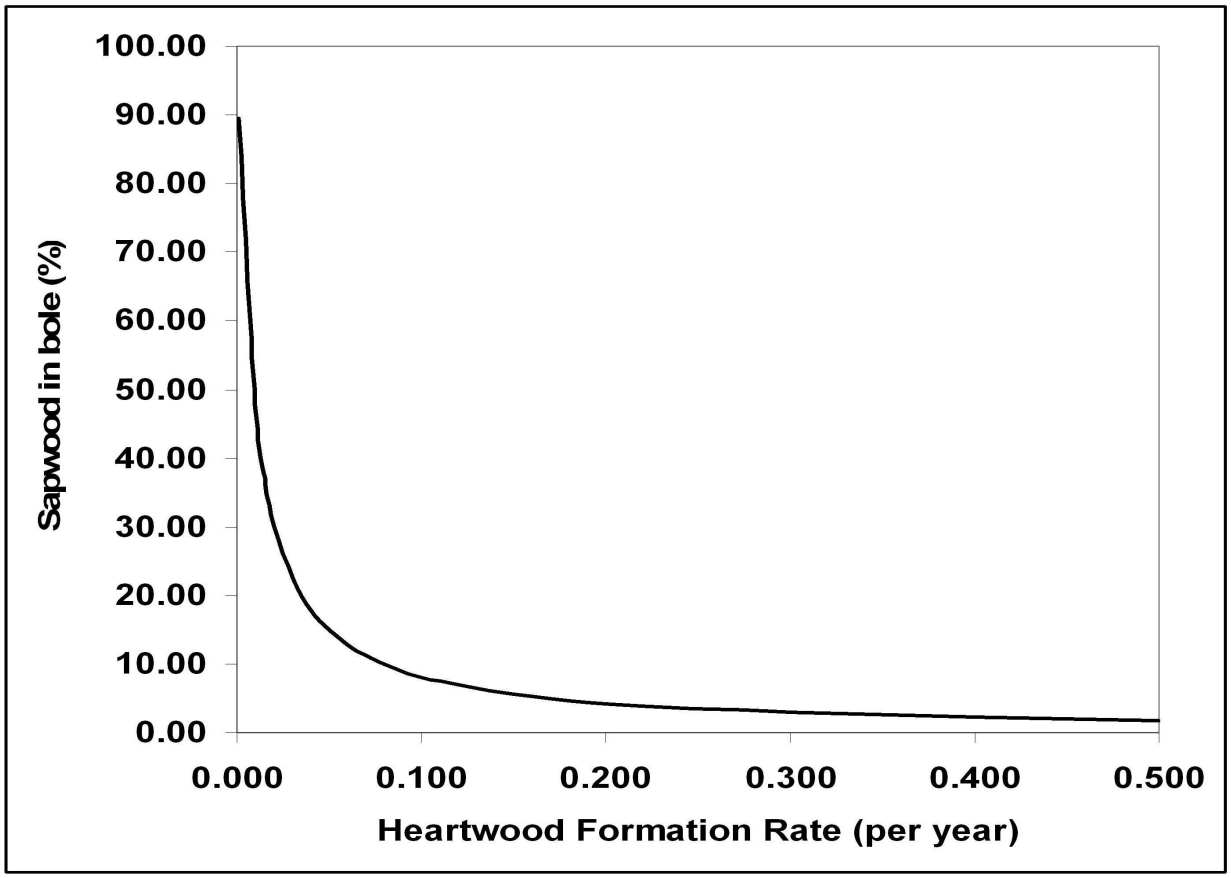

Figure 8-4. Influence of the Heartwood Formation Rate parameter on the fraction of sapwood in the bole of trees without heart-rot in the LANDCARB model.

Heartrot Function.

This function calculates the rate that heartrot is formed from heartwood for the tree layers (Figure 8-5). For each species there is a characteristic time lag before heart-rots become significant (Harmon et al. 1996). This time lag (HeartRotLag) is set in the Growth.prm file for each tree

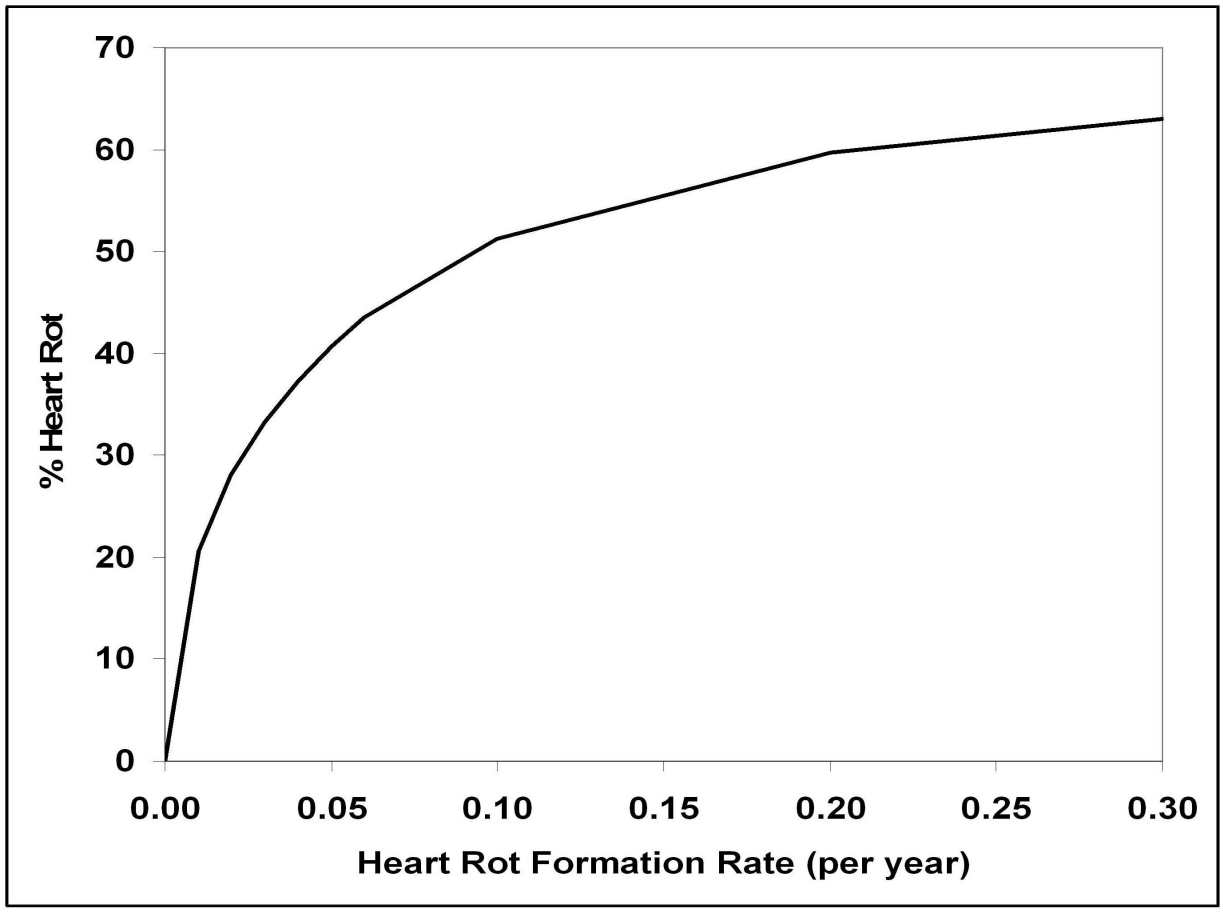

Figure 8-5. Influence of Heart-Rot Formation rate parameter on the fraction of wood that has heart-rot in the LANDCARB model.

species. If the time a layer has occupied a cohort is less than the time lag, then no heartwood is allocated to heart-rot. If the time a layer has occupied a cell equals or exceeds this lag, then the rate mass is transferred from heartwood to heart-rot *(Layer*HeartRotAlloc).

While only two tree layers are explicitly modeled (the upper and lower trees), conceptually we need to consider three because the lower tree eventually is replaced by a new generation of lower trees many of which do not contain heart-rot. We vary the rate mass is transferred from heartwood to heart-rot *(Layer*HeartRotAlloc) so as to reflect effect of the age-class structure of the layer and the fact not all trees will be infected by heart-rot at the same age. The latter is based on the assumption that once the trees in an age-class pass the time lag associated with heart-rot formation, then there is a window in which they all become infected by heart-rots. This window

is defined by the variable ProbHeartRotMax in the herb line of the Growth.rpm file in the rate heart-rot formation column.

The lag time to heart-rot formation differs for the three tree layers (recall that one, the second generation of lower trees is a virtual tree layer that exists in concept). For the upper tree, lag time to heart-rot formation is the value in the Growth.prm for the species occupying the upper tree layer (HeartRotLag). The value used for first generation of lower trees is the sum of the TimeClose and TimeCloseWindow for upper trees plus the heart-rot formation lag (HeartRotLag) for lower trees noted in the Growth.prm file. This means that the lower trees cannot have heart-rot until they are released by the dieout of the upper trees and the lower tree heart-rot formation lag is exceeded. The value used for the second generation of lower tree is the the sum of the TimeClose and TimeCloseWindow for the lower tree and upper tree layer. The latter term is used not to indicate when the heart-rot begins for this virtual layer, but to indicate when the transition from the first and second generation of the lower tree begins. This is because the second generation of lower trees has an all aged-structure, some of these trees have heart-rot, while others do not.

The age-class distribution to be used also varies with the tree layer being considered. For the upper trees, the age class structure is determined by the rates of upper tree colonization as set by the Estab.prm file. For the first generation of the lower trees, the age-class structure is determined by the rate that upper trees die-out, a distribution described by the upper tree ExtRate parameter (see Die-out Module). This accounts for the fact lower trees may be present, but they only become large trees susceptible to heart-rot when upper trees start to die out. For the second generation of the lower trees we do not have an age-class distribution per se, but we do track the relative importance of the first and second generation of lower trees. We assume that the age-class distribution of the second generation of lower trees is stable and does not change in time and that it is created by the dying out of the first generation as described by the ExtRate for lower trees.

For the upper trees and the first generation of lower trees a joint probability distribution derived from the age-class structure of trees in a cohort and the ProbHeartRotMax parameter is used to transition from no heart-rot formation to the maximum rate of heart-rot formation for each species as defined by *Layer*RateHeartRotForm on the herb line in the Growth.prm file.

The cumulative probability of heart-rot once a tree passes the age of the heart-rot lag is varied non-linearly over time:

$$ProbHeartRot=(1-exp[-ProbHeartRotMax*(TimeThere-HeartRotLag)])^{HRShape}$$

Where HRShape is the value stored in the shrub line of the Growth.rpm file and the rate heart-rot formation column. The probability of becoming infected with a heart-rot for any year once the heart-rot lag is passed is the difference in the cumulative probability from one year to the next.

For the second generation of lower trees, we assume that once the upper tree layer has died out and first generation of the lower trees has started to die out, that an all-aged structure developes.

This means that some trees are too young to have heart-rot, while others are of sufficient age to have heart-rot. This effectively lowers the average rate of heart-rot formation. The amount of decrease depends on the proportion of the second lower tree layer that is too young to have heart-rot.

Once the time lag for the second generation of lower trees is passed the proportion of the second generation of lower trees is:

$$PropSecondGenLT= (1- \exp[-LTExtRate*TimeSecondGen)^{2}$$

Where LTExtRate is the extinction rate of the lower trees and TimeSecondGen is the time since the second generation started to form which is equal to:

TimeSecondGen=Time-TimeLagSecondGen

Where TimeLagSecondGen is the sum of the upper tree and lower tree TimeClose plus TimeCloseWindow values. Note that before TimeLag2ndGen the proportion of second generation lower trees is assumed to be zero.

The relative rate of heart-rot formation once the second generation of lower trees has become completely established is a function of the proportion of second generation lower trees that are old enough to have heart-rot. This proportion is determined as:

$$RelHRForm2ndGen=(1-\exp[-LTExtRate*TimeWithHR])^{2}$$

Where

TimeWithHR = (3/LTExtRate)-LT HeartRotLag-(3/ProbHeartRotMax)

The last equation computes the number of years of the second generation has heart-rot.

The relative adjustment for heart-rot formation once the first generation of lower trees begins to die out (or conversely once the time lag for the second generation of lower tree is exceeded) is:

SecondGenHRAdjust= RelHRForm2ndGen+ (1- RelHRForm2ndGen)*(1- PropSecondGenLT)

Heart-rotFor each tree layer the mass of heartwood formed in a year is determined by the rate of heartrot formation (*Layer*RateHeartRotForm) and the mass of heartwood (*Layer*HeartWood) from the previous time step:

*Layer*HeartRotAlloc=*Layer*RateHeartRotForm**Layer*HeartWood

*Layer*RateHeartRotForm is parameterized so that the proportion of boles in heartrot matches the values in mature trees of the various tree species (Harmon et al. 1996b).

Mortality Function.

This function calculates the mass of all plant parts lost by normal mortality processes. Even without harvest or wildfire some of the trees in a cohort are subject to mortality caused by competition (self thinning), wind, insects, and pathogens. It is assumed that when trees are subject to natural mortality, all the plant parts for that tree are added to detrital pools. The equation describing these losses is:

*LayerPart*Mort=*Layer*MortalityRate**LayerPart*

*Layer*MortalityRate is the proportion of trees dying and is calculated in the MORTALITY module and *LayerPar*t is the mass of a layer from the previous time step . This parameter does not remain constant, but increases as the amount of light absorbed increases so that when the maximum amount of light is absorbed, the maximum mortality rate is reached. This mimics the increased competition among individuals as the canopy closes.

Prune Function.

This function calculates the mass of foliage, fine roots, branches, and coarse roots that are lost to litterfall, fine root turnover, or pruning.

The mass of these non-woody plant parts for each layer lost to these processes is:

*LayerPart*Turnover=*LayerPart*TurnoverRate*Layer*Part*

where Layer*Part*Turnover is the mass of non-woody plant parts of a layer lost from normal foliage fall and fine root turnover, *LayerPart* is the mass from the previous time step of the live part being considered, and *LayerPart*TurnoverRate is the fraction of the part for a layer that is dying in a given year. *LayerPart*TurnoverRate has different names depending upon the plant part considered. FoliageTurnoverRate and FRootTurnoverRate are used for foliage and fine roots respectively

The mass of these woody plant parts for each layer lost to these processes is:

*LayerPart*Prune=*LayerPart*PruneRate*Layer*Part*

where Layer*Part*Prune is the mass of plant parts of a layer lost from pruning, *LayerPart* is the mass from the previous time step of the live part being considered, and *LayerPart*PruneRate is the fraction of the part for a layer that is pruned in a given year. *LayerPart*PruneRate has different names depending upon the plant part considered. BranchPruneRate and CRootPruneRate are used for branches and coarse roots, respectively.

Live Stores Function.

This function calculates the mass of the plant parts after normal growth and mortality. The change in mass for each plant part caused by harvest and fire are calculated by HARVEST, PERSCRIBED FIRE and WILDFIRE, respectively. The balance of foliage is calculated as follows:

LayerPart = *LayerPart*Old + *LayerPart*Alloc

Where *LayerPart*Old is the mass of the part from the previous year, and *LayerPart*Alloc is the mass of the plant part produced

For fine roots the mass in any year is:

LayerPart = *Layer*FineRootOld + *Layer*FineRootAlloc –*Layer*FineRootResp
- *Layer*FineRootTurnover-*Layer*FineRootMort

Where *Layer*FineRootOld is the mass of the fine roots from the previous year, *Layer*FineRootAlloc is the mass of fine roots produced , *Layer*FineRootResp is the mass respired, *Layer*FineRootTurnover is the mass of fine roots lost to turnover, and *Layer*FineRootMort is the mass dying with boles because of normal mortality

For branches and coarse roots the mass in any year is:

LayerPart = *LayerPart*Old + *LayerPart*Alloc -*LayerPart*Resp- *LayerPart*Prune
-*LayerPart*Mort

Where *LayerPart*Old is the mass of the part from the previous year, *LayerPart*Alloc is the mass of the plant part produced (for branches this is *Layer*BranchAlloc and for coarse roots this is *Layer*CoarseRootAlloc), *Layer*Part is the mass of plant part for that year, *LayerPart*Resp is the mass respired, *LayerPart*Prune is the mass pruned, and *LayerPart*Mort is the mass dying with boles because of normal mortality

The mass of sapwood for a layer is calculated as:

*Layer*Part = *Layer*PartOld + *LayerPart*Alloc -*LayerPart*Resp
-*Layer*SapWoodHeartWoodAlloc-*LayerPart*Mort

where all the variables are the same as for branches and coarse roots except that *Layer*SapWoodHeartWoodAlloc is the mass of sapwood allocated to heartwood.

The mass of heartwood for a layer is calculated as:

*Layer*Part = *Layer*PartOld + *LayerPart*Alloc -*LayerPart*Mort-*Layer*HeartRotAlloc

where all the variables, except *Layer*HeartRotAlloc, the allocation to heart-rot, are defined as for the other plant parts. In the case of heartwood *LayerPart*Alloc is *Layer*SapWoodHeartWoodAlloc, the mass of sapwood allocated to heartwood formation.

The mass of heart-rot for a layer is calculated as:

*Layer*Heartrot= *Layer*HeartrotOld + *Layer*HeartrotAlloc -*Layer*HeartrotResp
-*Layer*HeartrotMort

where all the variables are the same as above.

In addition to these plant parts the mass in boles for the upper or lower tree layers is calculated as

*Layer*Bole=*Layer*Sapwood+*Layer*Heartwood.

The total live mass is the sum of all the live parts:

*Layer*TotalLive=Σ *Layer*Part

where LayerPart is the mass of the part for each layer. In addition to these totals, the total mass of each part is also summed across all the layers.

9-MORTALITY

The purpose of this module is to calculate the mortality, pruning, and turnover rates of plant parts (foliage, fine roots, branches, sapwood, heartwood, heartrot, and coarse roots) associated with normal growth processes. The rate that is calculated depends on the layer and the plant part being considered. These variables are used by the GROWTH and DECOMPOSE modules to adjust the live mass of parts or to calculate inputs to dead pools. Dead inputs associated with either harvest or fire are calculated by the HARVEST, PERSCRIBED FIRE or WILDFIRE modules.

During the course of normal growth, the mortality of sapwood, heartwood, and heart-rot occurs when a tree dies. In contrast, the mortality of foliage, fine roots, branches, and coarse roots occurs when a tree dies or when parts are pruned. That is, when the sapwood, heartwood, and heartrot of tree dies, we assume that all the associated plant parts will also die. There is, however, some mortality of these non-bole plant parts even when a tree does not die caused by normal pruning and replacement.

The parameters required by this function are stored in the Mort.prm file.

TreeMort Function

This function determines the rate upper and lower trees die (Figure 9-1). The rate of tree mortality is dependent upon the age of a layer in a cohort. Trees are divided into those capable of expanding horizontally, and those that can not. If the time a species has occupied cohort (TimeThere) is less than the age required for trees to reach its maximum horizontal extent (TimeClose), then mortality rates are dependent on the amount of light absorbed in a cohort. This mimics the effects of density dependent thinning. If the time a species occupies a cohort is greater than TimeClose, then tree mortality is influenced by the maximum age a tree species can attain. This mimics the effects of density independent mortality.

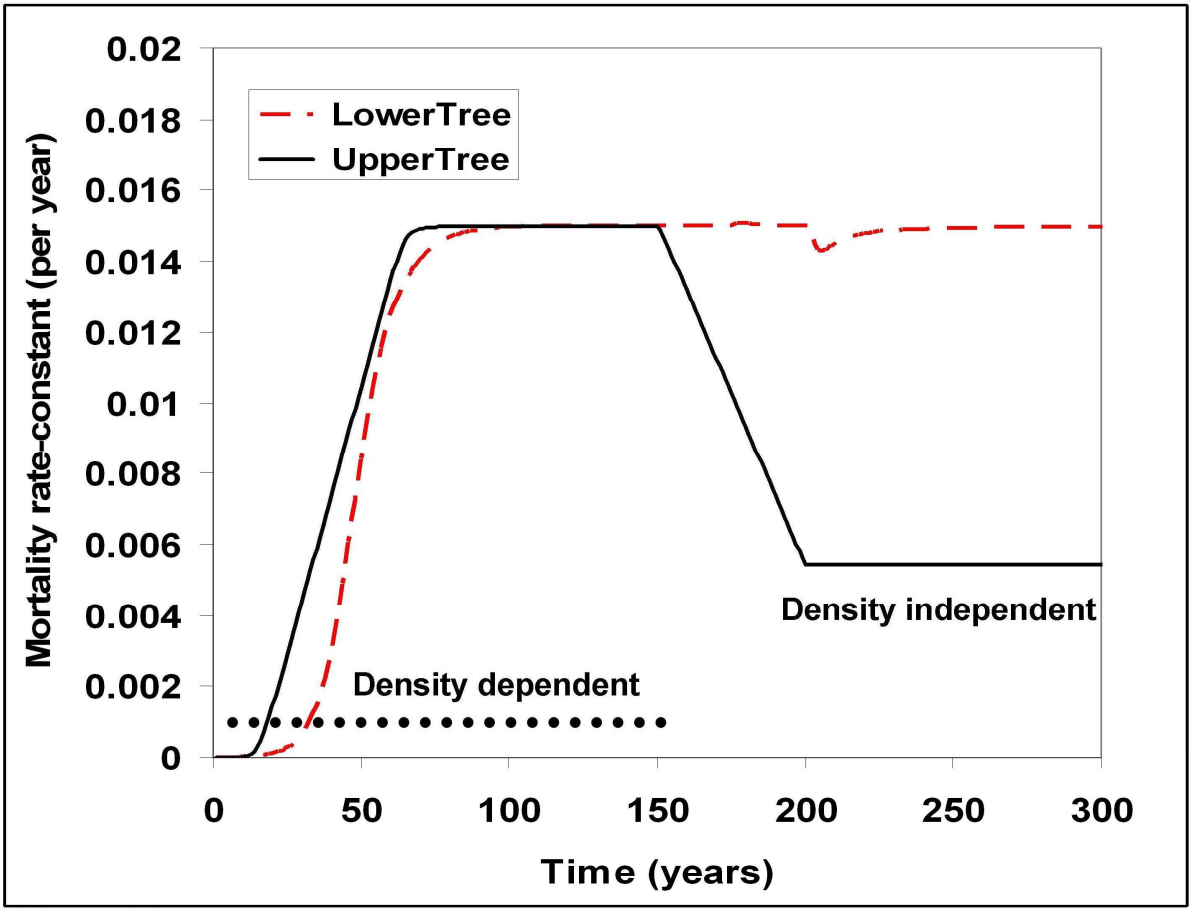

Figure 9-1. Changes in mortality rate-constant predicted over time in the LANDCARB model. Upper trees establish in a pulse, whereas lower trees occur in a wide range of ages.

Since upper trees establish in a pulse and lower trees establish gradually their age structure differs and this influences how mortality changes once TimeClose is reached. The transition between these different mortality rates for upper trees is controlled by the age-class structure of a tree layer in a cohort. Since not upper trees will reach their maximum horizontal extent at the exact same age, the transition is also modified by a probability describing this variation in timing from tree to tree. For upper trees the age-class structure and the probability trees reach their maximum horizontal extent are combined to produce a joint probability distribution that is used to weight the relative contribution of density dependent and independent forms of mortality. For lower trees, the gradual establishment of trees results in an all-aged structure. This means that in old stands that mortality is an average of both density dependent and density independent rates.

A major assumption used to calculate density dependent mortality rates is that as the amount of light absorbed by the stand increases, the mortality rate for trees increases. The amount of light absorbed is taken from the GROWTH module. When TimeThere is less than TimeClose:

*Layer*MortRateDensityDependent=MortMax*(LightAbsorbed/MaxLightAbsorb)

where *Layer*MortRateDensityDependent is the rate trees die due to density dependent causes, LightAbsorbed is the amount of light absorbed by the layer, and MaxLightAbsorb is the maximum amount of light that can be absorbed by the tree layer. The later two variables are calculated by the GROWTH module. This function increases mortality as a positive function of the total amount of light removed, mimicking the phase of self thinning during the middle stages of stand development.

For upper trees when TimeThere is greater than or equal to TimeClose, then the tree mortality rate is:

TreeMortRateDensityIndependent= ExtRate

As defined in the DIEOUT module.

The overall upper tree mortality rate represents a weighted average between density dependent and density independent causes. The weighting factor is the cumulative proportion of trees that have exceeded TimeClose and the additional temporal variation (CumProbGTTimeClose).

TreeMortRate= (1- CumProbGTTimeClose)* *Layer*MortRateDensityDependent
+ CumProbGTTimeClose* *Layer*MortRateDensityIndependent

To estimate this average the age-structure of lower trees is approximated by assuming a weighting value of 1 for each year below TimeCLose for this species and then a value determined by a negative exponential function described by the extinction coefficient. For lower trees younger than TimeClose, the density dependent mortality rate is used, whereas for those older than TimeClose, the density independent mortality rate is used. The product of the weighting value and the mortality rate is then used to estimate a weighted average mortality rate. The transition time between the earlier density dependent stage of mortality for lower trees and the all-aged mortality rate is determined by the TimeCloseWindow for lower trees.

BranchPrune Function

This function determines the rate that branches of upper and lower canopy trees and shrubs are lost via pruning. The rate branches are pruned (BranchPruneRate) is positive function of the total amount of light removed and the maximum rate of pruning for an intact stand (BranchPruneMax).

BranchPruneRate=BranchPruneMax*(LightAbsorbed/MaxLightAbsorb)

where LightAbsorbed is the amount of light that is absorbed by a tree or shrub layer and MaxLightAbsorb is the maximum light available for the layer to absorb, as calculated in the GROWTH module.

CRootPrune Function

The rate coarse roots of upper and lower tress and shrubs are pruned is calculated by this function. The rate coarse roots are pruned (CRootPruneRate) is positive function of the total amount of light removed and the maximum rate of pruning for an intact stand (CRootPruneMax).

CRootPruneRate=CRootPruneMax*(LightAbsorbed/MaxLightAbsorb)

where LightAbsorbed is the amount of light that is absorbed by a tree layer and MaxLightAbsorb is the maximum light available for the layer to absorb as calculated in the GROWTH module.

Foliage Function.

This function determines which rate of foliage turnover used in DECOMPOSE. For all the plant layers the rate foliage is added to the DeadFoliage pool is defined by LeafTurnoverRate as stored in the Mort.prm file.

FineRoot Function.

This function determines which rate of fine root turnover will be used by the DECOMPOSE module. The rate fine roots turnover (FineRootTurnoverRate) is positive function of the total amount of light removed and the maximum rate of fine root turnover for an intact stand (FineRootTurnoverMax) as stored in the Mort.prm file.

FineRootTurnoverRate=FineRootTurnoverMax*(LightAbsorbed/MaxLightAbsorb)

where LightAbsorbed is the amount of light that is absorbed by a tree, shrub, or herb layer and MaxLightAbsorb is the maximum light available for the layer to absorb, as calculated in the GROWTH module.

10-DECOMPOSE

This module is used to simulate the input, decomposition, and storage of carbon in dead and stable pools within cohorts. It also determines the rate charcoal is buried in soil.

All dead **pools** are named after the corresponding live plant parts with the prefix Dead added. Six pools of dead carbon are considered: 1) the **dead foliage** derived from foliage, 2) **dead fine roots** which can be either in the organic or mineral soil 3) **dead branches** (fine woody debris) derived from branches, 4) **dead sapwood** (one form of coarse woody debris), 5) **dead heartwood** (another form of coarse woody debris), and 6) **dead coarse roots**. For computational purposes dead sapwood and dead heartwood are further subdivided into salvageable versus non-salvagable and snag versus log pools (Figure 10-1).

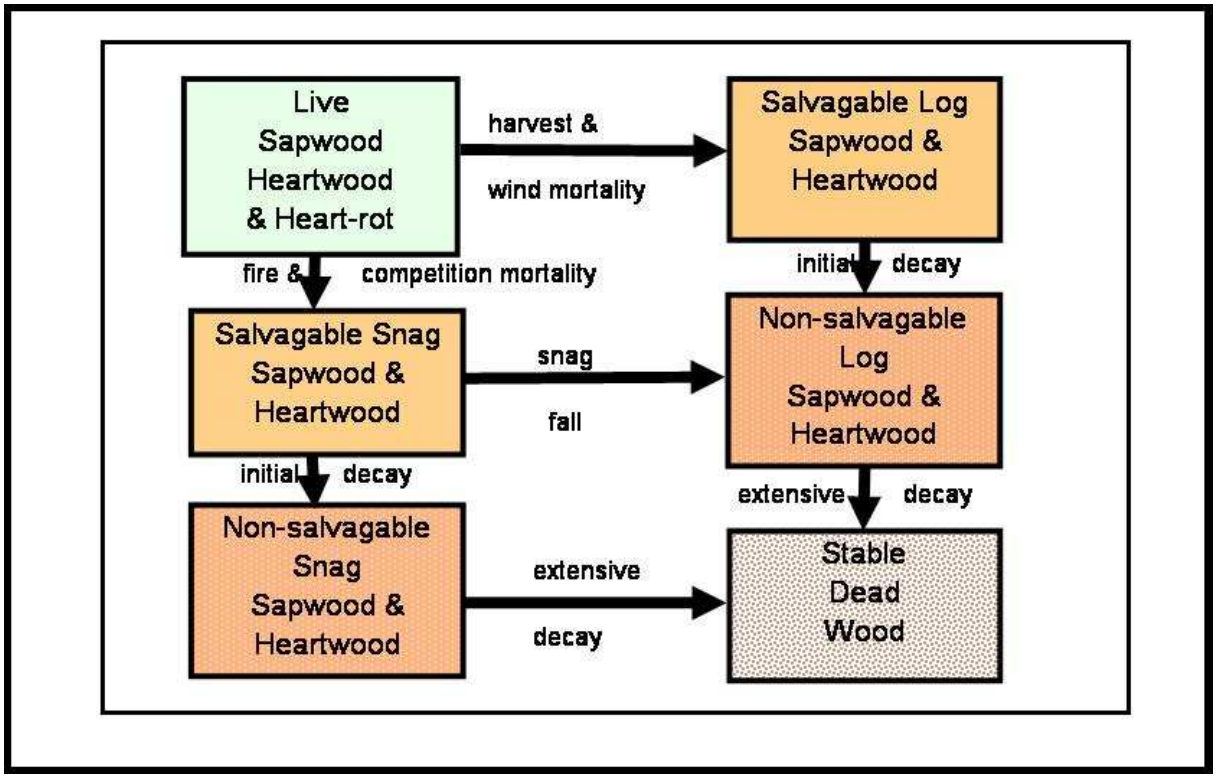

Figure 10-1. Coarse dead wood pools and flows modeled by LANDCARB.

In addition to these dead pools, the model simulates the dynamics of three stable pools (**stable foliage**, **stable wood**, and **stable soil)** that represent extremely decomposed organic matter. Despite the prefix stable, these pools do lose organic matter via decomposition, albeit at a very slow rate.

The burial of charcoal is also tracked by this module, although its formation is computed in PRESCRIBED FIRE and WILDFIRE.

All of the plant layers can input dead parts into the DECOMPOSE module, however, some life forms do not have certain woody parts and therefore do not contribute to the woody dead pools. Herbs are assumed to contribute to the dead foliage and dead fine roots only. Shrubs contribute to the dead foliage, dead fine roots, dead branches, dead sapwood, and dead coarse roots. Finally, trees contribute to all the dead pools including dead foliage, dead fine roots, dead branches, dead sapwood, dead heartwood, and dead coarse roots.

The inputs of material to the dead pools comes from three potential sources: 1) normal litterfall and mortality, 2) harvesting, and 3) fire killed plants. The first input is calculated by MORTALITY, the second by HARVEST, and the third by either PRESCRIBED FIRE or WILDFIRE. Although pools receive inputs from the four plant layers, the input mass is aggregated and the substrate quality is averaged for each pool. The input of material into the stable pools is influenced by the lag required to form this material.

The rate that each dead pool decomposes and the time lag assocatiated with transfers to other pools is determined by the species (as parameterized by the Decomp.prm file), and the climate as calculated by the CLIMATE module. Decomposition rates of the stable pools are determined by the values stored in the DecayPool.prm file as modified by climatic factors calculated by the CLIMATE module.

The stores of dead carbon calculated in the DECOMPOSE module are used by the CLIMATE module to calculate the interception of water by the dead foliage, dead wood, and stable pools. Dead fine root, dead coarse root, and stable soil pools are assumed to not intercept water for water balance purposes.

The files directly used by this module are Decomp.prm and DecayPool.prm.

Dead Pool Input Function.

This function is used to calculate the total input into each of the dead pools. The inputs of material to the dead pools comes from three potential sources: 1) normal litterfall and mortality, 2) harvest and 3) fire. For any given year, the input can come from several of these sources. Each year the inputs from normal litterfall and mortality are calculated first, and then additional inputs from harvest or fire are added. The inputs from harvest and fire are calculated in separate time steps that represent a fraction of a year. This method is used to avoid possible conflicts in calculating the mass of pools.

Inputs to each of the dead pools are calculated as follows where *Pool* represents a specific dead pool, *Layer* represents a plant layer, and *Part* represents a plant part:

For the parts foliage, fine roots, branches, and coarse roots in cells not harvested or burned, the input is :

*LayerPool*Input=*Pool*TurnoverRate**Part* + MortRate**Part*

where *LayerPool*Input is the input mass from normal leaf fall, fine root turnover, and pruning for a particular layer and part, *Part* is the mass of the live part being considered, *Pool*TurnoverRate is the fraction of the part for a layer that is pruned or replaced in a given year, and MortRate is the mortality rate of trees. *Pool*TurnoverRate has different names depending upon the plant part considered; FoliageTurnoverRate, FRootTurnoverRate, BranchPruneRate, and CRootPruneRate are used for foliage, fine roots, branches and coarse roots, respectively. All these variables are calculated by the MORTALITY module. MortRate it is also calculated in the MORTALITY module and accounts for the input of non-bole parts associated with the mortality of entire trees.

For the parts sapwood, heartwood, and heartrot, in cohorts not harvested or burned, the input is:

*LayerPool*Input=MortRate**Part*

where *LayerPool*Input in this case is the input mass from sapwood, heartwood, or heartrot of dying trees, *Part* is either sapwood, heartwood, or heartrot mass, and MortRate is the mortality rate of trees calculated in the MORTALITY module.

Plant parts from layers can also be added to dead pools when trees are harvested. These inputs are calculated after normal mortality inputs are calculated in a time step representing a fraction of a year (e.g., 12.2). If the trees in a cohort are harvested then the inputs from foliage, fine roots, branches, sapwood, heartwood, and coarse roots are:

*LayerPool*Input=*Pool*Harv

where *Pool*Harv is the amount of material generated from a layer as input to a dead pool by harvest activities from the HARVEST module.

Finally, plant parts from layers can be added to dead pools when plants are killed by prescribed fire or wildfires. These inputs are calculated after normal mortality inputs and harvest inputs are calculated in a separate time step that represents a fraction of a year (e.g., 12.3). If plants in a cell are killed by fire, then the inputs from foliage, fine roots, branches, sapwood, heartwood, and coarse roots are:

*LayerPool*Input=BurnInput*Pool*

where BurnInput*Pool* is the amount of plant parts killed by fire but not consumed as calculated by either PRESCRIBED FIRE or WILDFIRE.

The total input to a pool (*Pool*Input) at any time step is the sum of all the inputs from the layers in a cell.

*Pool*Input=Σ (*LayerPool*Input)

Dead Substrate Effect Function.

This function calculates the effect of the substrate quality of the various inputs on the overall decomposition rate of a dead pool. This function is invoked each time inputs are added to a cohort. Thus is possible to invoke this function three times in one year if normal growth, harvest, and fire occurs in a year.

The decomposition rate of each pool is dependent on the substrate quality of the inputs to that pool and the current substrate quality of the pool. The overall decomposition rate is a weighted average of the input and current stores. This has the dual effect of building in a system memory but allowing the decomposition rate to gradually change if the substrate quality of the inputs change.

The first step is to calculate the weighted average decomposition rates of the inputs of each pool from the herb, shrub, lower tree, and upper tree layers so that the layers with the largest inputs have the greatest impact on the decomposition rate:

InputDecayRate*Pool*=Σ (*LayerPool*Input*DecayRate*LayerPart*)/*Pool*Input

where InputDecayRate*Pool* is the weighted average decomposition rate of the inputs to a dead pool, *LayerPool*Input is the mass input of each plant part from a specific plant layer (e.g., herbs) to a dead pool, *Pool*Input is the total input of all layers to a pool, and DecayRate*LayerPar*t is the decomposition rate of a part for a layer at 10 C and when moisture is not limiting. For herb and shrub layers the latter parameter is fixed for the entire layer. For trees, however, this parameter is a function of the tree species occupying the particular tree layer. The values of DecayRate*LayerPar*t are the values stored in the Decomp.prm file.

The second step is to calculate the weighted average decomposition rate from the average substrate quality of the inputs and the current material within the dead pool. This step builds in a system memory and allows the decomposition rate of a dead pool to change gradually when the substrate quality of the inputs change. Therefore one can not change the decomposition rate of a dead pool unless the change in substrate quality of the inputs is continued. The weighted average decomposition rate of each dead pool is:

*Pool*DecayRateAvg=(InputDecayRate*Pool****Pool*Input
+Old*Pool*DecayRateAvg**Pool*)/(*Pool*Input+*Pool*)

where *Pool*InputDecayRate and *Pool*Input are as above, Old*Pool*DecayRateAvg is the weighted average decomposition of each dead pool from the past year, and *Pool* is the last year's mass of a particular dead pool. An array of the values from the previous year containing Old*Pool*DecayRateAvg is stored and used to calculate the current value of *Pool*DecayRateAvg.

Position Function.

This function determines if dead wood pool inputs from sapwood, heartwood, and heartrot are added in the "position" of a snag or log. This dichotomy is important in that it determines the water balance of the dead wood and hence the decomposition rate and the time it remains in a salvagable condition.

The proportion of sapwood, heartwood, and heartrot that is added to the snag versus log pool depends on the location of the site, cause of mortality, the age of the trees in a cell, and the tree layer considered:

1). In addition, the proportion of snags dying from normal mortality causes is determined from the location as defined in the Ecoregion.prm file. Most trees die as snags early in stand development, however, the proportion of snags decreases as stands approach the old-growth phase. The proportion of snags also varies with location in the Pacific Northwest, with the the lowest proportion of snags along the wind-prone coastal zone, and the highest in the insect-prone interior zone (Franklin et al. 1987).

2) All upper and lower trees dying from fire, associated with either wildfire or site preparation are added to the snag pools.

3) All woody parts of upper and lower trees that result from cutting operations are added to the log pools.

Lag Emulation Function.

For many decomposition processes there are lags before a transfer can occur. Specifically, the rate dead mass is transfered from salvagable to non-salvagable pools, from snag to log pools, and to stable pools all involves lags. That is, snags do not break into log pieces until a period of decomposition occurs. Likewise, dead wood is merchantable until a large proportion of the material is decayed. In STANDCARB a dead pool cohort data structure is used to introduce these lag time effects. This is a memory intensive approach, and is not used in LANDCARB. Rather than directly track cohorts to create lags, LANDCARB emulates the lag process.

The decomposition lag emulator is based on observations of how these lag effects interact with the amount of input to create deviations in the perceived rate-constant of the transfer of one pool to another. For example, as long as the input is constant, the perceived transfer rate-constant appears to be a constant over time. In contrast, when there is a large pulse of input the transfer rate-constant appears to change. Initially it decreases, because the pool is dominated by new material that has not decomposed sufficiently to be transferred. Eventually as the pulse of input is decomposed, the transfer rate-constant appears to increase and then decrease back to a lower value reflecting a more even input of dead mass. These behaviors are emulated by the following:

1. The transfer rate-constant is initially be set to the average that occurs when inputs to a dead pool are constant. This is specified in the DecayPool.prm.

2. Whenever there is a disturbance in a cohort, a counter is started called the preceeding lags counter. The value of this counter is equal to the lag time (adjusted for the AbioticDecay Index) of the preceeding pools that may have received the disturbance related pulse. If the disturbance adds directly to the pool of interest, then this counter is set to zero. Each subsequent year the counter is decreased by 1. This counter is used to account for the fact that lag for a transfer to occur is dependent on how many steps occur between the time of the disturbance and the transfer of interest.
3. When the preceeding lags counter has reached zero another counter called the disturbance input counter is initiated. The value of this counter is equal to the lag time (adjusted for the AbioticDecay Index) of the transfer of interest. Each subsequent year the counter is decreased by 1. During this time the perceived transfer-rate constant will be set to a level lower than the normal average value.
4. When the disturbance input reaches zero, then increased rates of transfer from the disturbance related input can occur. When the disturbance input counter reaches zero two things happen: a) The disturbance input counter is set to zero until the next disturbance and b) a new counter called the disturbance transfer counter is set to a value that is determined from the rate that cohorts are cleared out as defined in the DecayPool.prm file:

Disturbance Transfer Counter=3/CohortClearingRate

Cohort in this case refers to the decomposition related cohorts used in STANDCARB. In each subsequent year the disturbance transfer counter is reduced by 1. When the disturbance transfer counter is zero, then the transfer rate-constant is returned to the normal average value (see 1 above). Moreover, the disturbance transfer counter is also set to zero until the next time the disturbance input counter becomes positive and reaches zero. When the disturbance transfer counter is greater than zero, the transfer rate-constant will be above that of the normal average value.

The degree that the transfer rate-constant decreased by input from a disturbance depends on the size of the input from disturbance relative to the existing pool size:

TransferRateDuringLag=
transfer rate-constant average *(1-exp(-0.69**Pool*/(*Pool*Input)))

This will mean that if the input is equal to the pool, then the transfer rate-constant will be halved. As the imbalance between inputs and stores increases, the transfer rate-constant approaches zero. If the pool is much greater than the input, then the transfer rate-constant is equal to the average. The inputs and pool sizes are specific to the transfer in question and its related pool. *The input and pool mass considered is just for the year the disturbance input counter is initiated.*

The transfer rate-constant during the period that the disturbance-related input is being transferred is calculated in a series of steps:

MaxTransferRateDuringClearing= CohortClearingRate

This assures that the pulse is cleared out in a time compatible with the disturbance transfer counter. This maximum is adjusted so that the losses from decomposition and transfers does not exceed 100% of the pool size. The total losses from the pool are calculated as:

LossTotal= MaxTransferRateDuringClearing+ *Pool*Decay

Where *Pool*Decay is calculated below and represents the proportion decomposing based on the substrate quality and the climate. If LossTotal exceeds 1 then the MaxTransferRateDuringClearing is adjusted downward:

TransferRateDuringClearingAdjusted= 1- *Pool*Decay

If the LossTotal is equal to or less than 1 then

TransferRateDuringClearingAdjusted= MaxTransferRateDuringClearing

The transfer rate is then adjusted to account for the balance of new inputs versus the mass remaining in the pulse from the disturbance:

TransferReduction=exp [- 0.69*(*Pool*Input)/*Pool*]

When the ratio of inputs to the pool size becomes equal, then the transfer is reduced from that indicated by the cohort clearing rate by 50%. As the input for a period equal to the lag time becomes greater than the pool, then the transfer rate-constant will decrease. The inputs and pool sizes are specific to the transfer in question and its related pool. The input and pool mass is specific to each year that the Disturbance Transfer Counter is working.

The actual transfer rate when the pulse from the disturbance is clearing is:

TransferRateDuringClearing= TransferReduction*MaxTransferRateDuringClearingAdjusted

Lags are averaged based on the lag time for a process for each plant layer present in a cohort. This is a weighted average, weighted by the mass of input versus the existing mass of a dead pool. The input lag time is also a weighted average, but based on the inputs from the different plant layers. This is similar to the weighted average procedure used to calcuate the decomposition rate. The lag times are also adjusted for the effects of climate. The lag time in the Decompose.prm file is an optimum. This optimum is divided by the AbioticDecompositionIndex. Therefore as the AbioticDecompositionIndex decreases, the lag time increases. Lags are rounded up to the nearest year. Lags are all adjusted by the AbioticDecayIndex of the relevant pool except for the lags associated with snags. In the case of snags, the lag is adjusted by either the snag or the coarse root AbioticDecayIndex depending upon which is larger. The reason is that in cases where snag decomposition is low, snags fall to the ground because the roots break after decomposing. Conversely, if root decomposition is slow, then snags break along the stem.

To account for the possibility that disturbances may input a new pulse into dead pools before the decomposition-related lag are exceeded, a lag emulator is created each time a disturbance occurs. The effects of multiple disturbances on apparent lags is simulated by averaging the lag emulators for each disturbance. This average is weighted by the mass remaining in each pulse. When the series of three counters in a lag emulator reach their endpoint, the lag emulator is removed and does not impact the transfer rate calculations.

Dead Decomposition Function.

This function calculates the decomposition rate and mass of dead pools lost from decomposition. Decomposition rate is calculated from the substrate effect (see Dead Substrate Effect function above) and the effects of abiotic factors, temperature, solar radiation warming, and moisture as calculated in the CLIMATE module. The rate of decomposition losses from all the dead pools is:

*Pool*Decay=*Pool*DecayRateAvg**Pool*AnnualAbioticDecayIndex

where *Pool*Decay is the realized decomposition rate of a dead pool, *Pool*DecayRateAvg is the substrate quality determined rate when temperature is 10 C and moisture is not limiting, and *Pool*AnnualAbioticDecayIndex is the combined effects of temperature and moisture on decomposition as calculated in the AbioticDecayIndex function of the CLIMATE module.

The mass of dead lost via decomposition in a year from a pool is :

*Pool*DecayLoss= *Pool*Decay*Pool

where *Pool*DecayLoss is the mass lost via decomposition, *Pool*Decay is the realized decomposition rate of a pool, and *Pool* is the mass of a dead pool cohort.

Salvagable Transfer Function.

This function transfers mass from dead salvageable wood pools to dead non-salvageable wood pools:

SalvagableTransferPool=*Pool*SalvageTransfer Rate**Pool*

where SalvagableTransferPool is the mass transferred to the non-salvagable wood pools and *Pool* is the mass of a contributing salvageable dead wood pool.

Snag Transfer Function.

This function transfers mass from dead snag wood pools to dead log wood pools:

SnagTransferPool=*Pool*SnagTransfer Rate**Pool*

where SnagTransferPool is the mass transferred to log pools and *Pool* is the mass of a contributing snag dead wood pool.

Stable Transfer Function.

This function transfers mass from dead pools to stable pools:

StableTransferPool=*Pool*StableTransfer Rate**Pool*

where StableTransferPool is the mass transferred to the stable pool and *Pool* is the mass of a contributing dead pool.

Dead Stores function.

This function is used to calculate the change in the mass of dead pools each year. The balance for each dead pool is the inputs minus the losses from decomposition and transfers to the stable pools. Losses from fire are calculated by the PRESCRIBED FIRE or WILDFIE modules. The overall rate of change for a dead pool is:

Δ*Pool*=*Pool*Input -*Pool*DecayLoss-*Process*Transfer*Pool*

where *Pool*Input is calculated in the Dead Input function, *Pool*DecayLoss is calculated in the Dead Decomposition function, and *Process*Transfer*Pool* can represent salvageable pool, snag, or stable pool-related transfers depending on the dead pool.

The mass in a given dead pool for a given year is therefore:

Pool= Old*Pool* + Δ*Pool*

where Pool is the mass in particular dead pool, Old*Pool* is the value for the previous year, and Δ*Poo*l is as above.

Stable Soil Function.

This function controls the input and decomposition of stable organic matter. These represent three very stable pools in the model and should not change greatly over time unless the forest is removed for extensive periods. These include: 1) StableFoliage, which is derived from dead foliage (similar to O2 horizons), 2) StableWood, which is derived from logs and branches (similar to brown-rotted wood), and StableSoil, which is derived from dead fine and coarse roots (similar to mineral soil organic matter). The intent of this function is to mimic the slow changes in stores in these pools.

Because the decomposition parameters of stable pools are difficult to measure, these parameters may have to be changed for each particular site. We recommend that they be

estimated by running the model and comparing to the values expected for each ecosystem being examined. If the inputs of dead pools to the stable pools are constant and the estimated stable pool approximates the store expected for particular site, then the decomposition rates for these pools are probably correct. To see if inputs to the stable pools are constant one can examine the Mort.Dgn file. If the inputs are constant and the stable pool is increasing, then the decomposition rates are probably too low. If the inputs are constant and the stable pool is decreasing, then the decomposition rates are probably too high. The user should change the value of StableDecayRate in the DecayPool.prm file until it converges on the target level of the stable pool being calibrated.

The equation describing the inputs to the stable pools is the sum of all the various inputs from dead pools:

Stable*Pool*Input= Σ StableTransfer*Pool*

where StableTransfer*Pool* is the transfer rate from each of the dead pools as calculated in the Stable Transfer function.

As with the dead pools, the decomposition rate-constant of each stable pool (StablePoolDecay) is a function of the substrate and abiotic effects:

Stable*Pool*Decay=Stable*Pool*DecayRate*Stable*Pool*AnnualAbioticDecayIndex*OldStable*Pool*

where Stable*Pool*AnnualAbioticDecayIndex represents the combined effects of temperature and moisture calculated by CLIMATE, Stable*Pool*DecayRate is the rate of decomposition of a pool at 10 C when moisture conditions are not limiting, and OldStable*Pool* is the mass for the previous year.

The overall rate of change for each pool is:

Δ Stable*Pool*=Stable*Pool*Input-Stable*Pool*Decay

The stores for this pool in a given year are:

Stable*Pool*=OldStable*Pool*+ Δ Stable*Pool*

where OldStable*Pool* is the mass for the previous year and Δ Stable*Pool* is calculated as above.

Total Dead Stores Function.

This function calculates the total mass of dead pools and total mass of stable carbon. These values are reported in the Total.out file.

Charcoal Function.

This function computes mass of charcoal and the rate that charcoal is buried, becoming protected from fire losses. Charcoal is followed in two pools: 1) a surface pool that is created and consumed by fire and 2) a buried pool that transferred from the surface pool and is not subject to losses via fire or decomposition (Figure 10-2). The rate the surface charcoal pool is transferred is determined in the DecayPool.prm file.

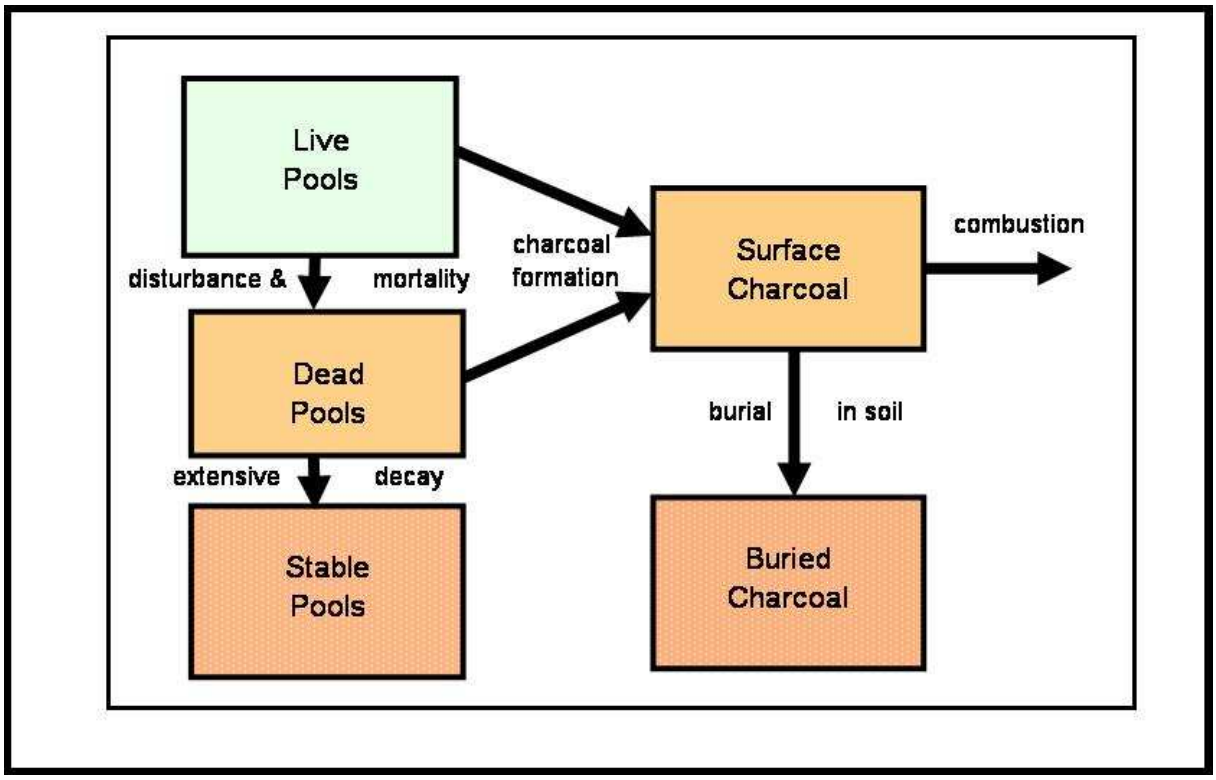

Figure 10-2. Charcoal formation, combustion, and burial used in the LANDCARB model.

The mass of the surface charcoal pool is:

SurfaceCharcoal=SurfaceCharcoalInput+SurfaceCharcoalOld-CharcoalTransfer

Where SurfaceCharcoalInput is the amount of charcoal created from live parts and dead pools in PRESCRIBED FIRE or WILDFIRE, SurfaceCharcoalOld is the pool size the previous year, and CharcoalTransfer is the mass of surface charcoal buried:

CharcoalTransfer= SurfaceCharcoalOld*CharcoalTransferRate

SurfaceCharcoalInput is zero unless a fire occurs, then it is the sum of the charcoal formed from all the live parts and dead pools.

The mass of buried charcoal is:

BuriedCharcoal= CharcoalTransfer+BuriedCharcoalOld

Where BuriedCharcoalOld is the pool size the previous year.

11-HARVEST

The HARVEST module determines what happens to the live, dead, and stable pools when a stand grid cell is harvested (Figure 11-1). When a harvest occurs, all the cohorts are potentially impacted. The user specifies the harvest type, whether the new cohort area created, the utilization level, and the number of patches in a stand grid cell the harvest creates. While it is possible to change the area of new cohort space over time, we encourage the user to be conservative. The output variables of this module are used to modify the state variables in the GROWTH and the DECOMPOSE modules. Harvest is invoked the year a harvest is specified and can be for any number of cells or species. Harvest calculations are made following the calculation of changes associated with normal growth and decomposition.

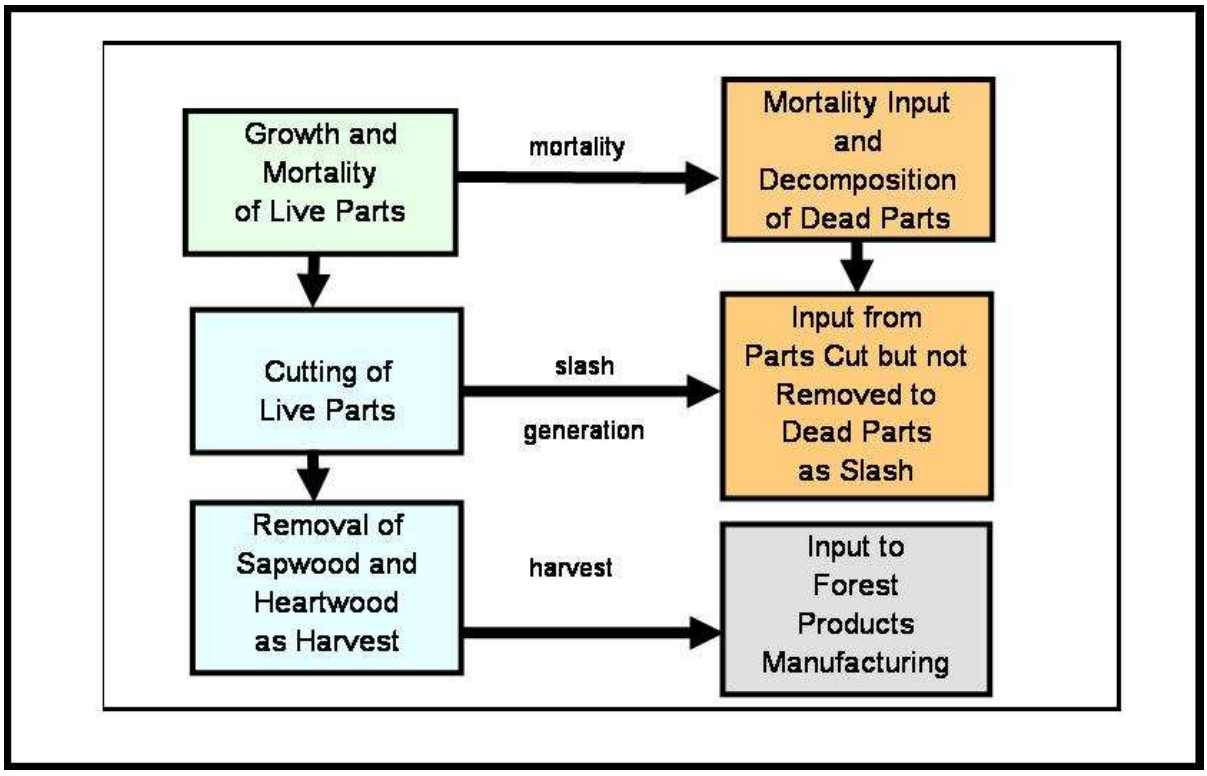

Figure 11-1. Processes occurring when a harvest occurs in the LANDCARB model.

Harvests can either create a new cohort (all the plant layers are killed) or if a new cohort is not created, then how existing cohorts are impacted. Specifically, to mimic partial harvest of a stand grid cell it is possible to use one of two options:

1) if plant layers are not completely killed by the harvest, the new cohort area is set to 0, and the utilization level indicates which plant layers are influenced and what degree. All cohorts would be effected by this kind of harvest;

2) if the plant layers are killed by the harvest, then area for a new cohort is created and plant layers are recolonized. Depending on the area of new cohort space created, some of the cohorts will be effected and other will not.

In the latter case the user needs to specify the number of patches the disturbance creates. There is a minimum number of patches that a harvest creates. If half the stand is harvested, then one needs to have at least two patches. If one-quarter is harvested, then one needs at least four patches. The number of patches can be any multiple of the minimum number of patches. The number of patches determines the mean distance between cohorts and the light levels recieved, but does not influence the number of cohorts that are modeled.

LANDCARB uses the concept of a virtual gridwork of substand cells within stand grid cells (Figure 11-2). In STANDCARB this substand gridwok of cells is explicit. The number of cells in the LANDCARB substand gridwork is set by the minimum number of patches implied by the area of new cohort space created by the harvest. If this disturbance pattern is repeated, then in time the number of cohorts in a stand grid cell will be the same as the minimum number of patches. This prevents a situation where the number of cohorts increases without limit with each disturbance cycle.

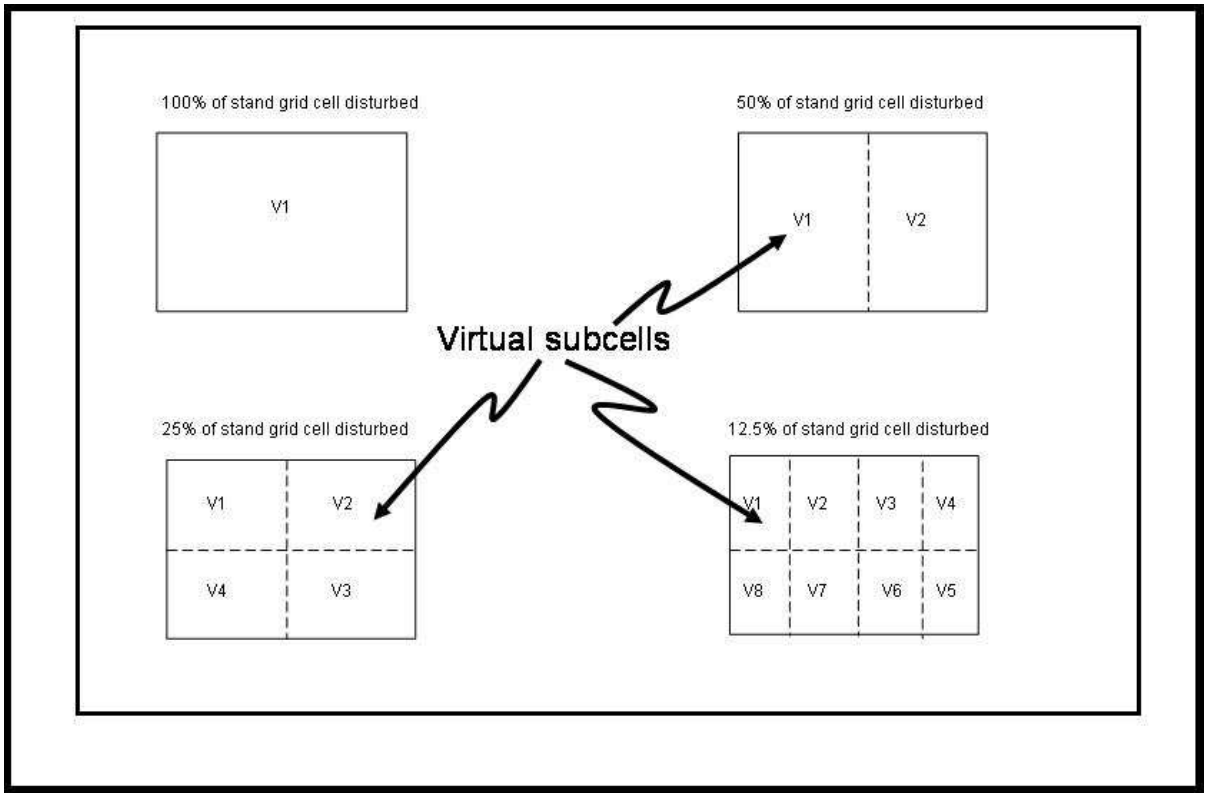

Figure 11-2. Examples of virtual subcells used by the LANDCARB model when a partial harvest occurs in a stant grid cell.

When a harvest occurs certain of the existing “patches” can be harvested. The user sets whether the youngest, oldest, or a random set of patches is harvested. If the oldest is selected, then an area equal to the “patch” size is harvested creating a new cohort. Suppose one starts with one cohort in a stand grid cell and 25% of the stand grid cell will be harvested in 4 patches. The first round of harvest would result in two cohorts; the original cohort would occupy 3 “virtual patches” and the new cohort would occupy 1 “patch”. After the second round of harvest there would be three cohorts, two occupying one “patch” and one occupying two “virtual patches”. With another round of harvest there would be four cohorts, each one occupying one “patch”. Yet another round of harvest and there still would be 4 cohorts each in one “patch”, but the original cohort would have disappeared. In the case where the youngest cohort is selected then there would always be two cohorts, as the youngest is always harvested. The first cohort would always occupy 3 “virtual patches” and the youngest one “patch”.

The utilization standards (i.e., fraction of the bole removed) for each type of harvest is defined by the Harvest.drv file. The schedule of harvests for stand grid cells is given in the HarvIntLC.drv file.

If harvesting occurs on a stand grid cell in a given simulation year, then the HARVEST module determines which activity is to occur as defined by the HarvInt.drv file. Possible activities include: precommercial thinning, commercial thinning, clear-cut harvest and timber salvage. All of these treatments except timber salvage impact the live tree layers and may or may not lead to a new cohort being formed. Precommercial thinning is defined as a thinning of the trees where all the bole material is left on the site as slash. In a commercial thinning, a proportion of the boles in a cohort is removed. In a clear-cut all the bole material for a cohort is cut. Timber salvage impacts the salvageable sapwood and heartwood and never leads to a new cohort being formed. For live trees if a harvest occurs and a new cohort is not formed, then harvest may be performed on the upper or lower tree layer, separately or together. When a new cohort is formed by harvest, then both the upper and lower tree layers are completely cut.

All the harvest activities result in the production of dead plant parts that are removed from the GROWTH module and added to the decomposition pools in the DECOMPOSE module. In addition, some of the bole material is removed as harvested mass.

The files used by this module is Harvest.drv and HarvIntLC.drv.

Convert Function.

This function converts the parameters from the Harvest.prm file from percentages to proportions:

AmtCut=AmtCut/100

AmtTake=AmtTake/100

where AmtCut is the fraction of the tree bole volume that is cut and AmtTake is the fraction of the boles that are cut that is taken as harvested material.

Harvest Function.

Once the type and timing of a harvest treatment has been determined from the HarvIntLC.drv file the Harvest function calculates the amount of sapwood and heartwood mass removed, the mass of sapwood and heartwood left in tops and stumps, the mass of salvageable sapwood and heartwood removed and the mass of dead material left as slash that was created by the harvest. Regardless of the type of harvest specified, the mass plant parts remaining after harvest is calculated as:

*Part*New= (1-AmtCut)**Part*Old

where

*Part*New and *Part*Old are the masses of plant parts after and before the harvest. Whenever a new cohort is to be formed, AmtCut is set to 1.0 (i.e., 100%).

Sapwood and heartwood can be removed from the site during harvest. The mass of sapwood and heartwood that is removed (*Part*Taken) from a cohort is calculated as:

*Part*Taken= AmtCut*AmtTake**Part*

where AmtCut is the proportion of the tree biomass cut and AmtTake is the proportion of the bole mass that is harvested and exported from the site to become forest products, and *Part* is either sapwood or heartwood mass. In most cases, AmtTake for precommercial thinning is set to zero.

We have used the convention that the input of detritus mass associated with harvest is named for the detritus pool with the addition of Harv (e.g., sapwood left to decompose after harvest is called DeadSapwoodHarv). The amount of sapwood and heartwood mass added to the DeadSapwood and DeadHeartwood pools due to harvesting is calculated as:

*Pool*Harv = AmtCut*(1-AmtTake)**Part*

where *Pool* is either DeadSapwood or DeadHeartwood, and *Part* is either Sapwood or Heartwood mass, respectively.

For other plant parts such as branches and coarse roots, the model assumes there is no export from the site. The mass of the non-bole parts transferred to their appropriate dead pool by commercial thinning is calculated as:

*Pool*Harv=AmtCut**Part*

where *Pool* is the detrital pool the material is being added to (i.e., DeadFoliage, DeadFineRoots, DeadBranch, and DeadCoarseRoots) and *Part* is the corresponding plant part mass (i.e., Foliage, FineRoots, Branches, and CoarseRoots).

When there is a salvage of timber the user specifies a proportion of the salvageable dead sapwood and heartwood that can be removed. The mass salvageable dead wood remaining after harvest is calculated as:

*Pool*New= (1-AmtRemoved)**Pool*Old

where

*Part*New and *Part*Old are the masses of salvageable dead wood pools after and before the salvage.

The amount removed in a salvage is:

*Pool*Taken= (AmtRemoved)**Pool*Old

Volume Function.

This function converts the mass of boles harvested from a cohort for a given year to wood volume. The volume of bole wood removed in harvest is estimated from the sapwood and heartwood mass removed and the fraction of boles in wood (WoodPer), and the wood density (WoodDen) for the given species harvested in a cohort. The parameters WoodPer and WoodDen are stored in the Growth.prm file.

The total mass removed in boles for a tree layer in a cell by harvest is:

Harvest=SapwoodTaken + HeartwoodTaken

The mass of wood (Wood) in the harvested boles is:

Wood=Harvest*WoodPer/100

where WoodPer is the percentage of the bole mass that is wood as opposed to bark (Wilson et al.1987). The volume of wood (HarvVol) is calculated by dividing the wood mass (Wood) by the wood density (WoodDen):

HarvVol=Wood/WoodDen

The values of wood density for each species is based on Marglin and Wahlgren (1972) and Wilson et al. (1987).

The total mass removed as salvaged boles in for a cohort by salvage is:

Harvest=DeadSapwoodTaken + DeadHeartwoodTaken

The fraction of wood (Wood) in the salvaged boles is:

WoodSalvaged=Harvest*WoodPer/100

where WoodPer is the percentage of the bole mass that is wood as opposed to bark (Wilson et al.1987). The volume of salvaged wood (SalvageVol) is calculated by dividing the wood mass (Wood) by the wood density (WoodDen):

SalvageVol=Wood/(0.9*WoodDen)

Where the wood density is decreased to account for decay.

12-PRESCRIBED FIRE

This module determines the effect of prescribed fire on plant layers, dead pools, stable pools, and charcoal pools (Figure 12-1). It may be invoked after a timber harvest or independent of timber harvest. For live vegetation it determines the amount that is killed and consumed by a fire and modifies the amount of live carbon in the GROWTH module and transfers some of this material to the DECOMPOSE module as fire-killed dead pool inputs. Not all the live vegetation killed by fire is necessarily transferred to detritus; some is consumed by the fire itself. For dead pools this module reduces the amount of dead material in the DECOMPOSE module to reflect the losses caused by fire. The changes in live parts, dead and stable pools is described in the Prescribedfire.prm file. The PrescribedFire.prm file is set up so that as fire severity increases from light to hot, the fraction each live part or dead pool killed or removed by fire increases. This module also controls the amount of charcoal this formed from live and dead pools and amount of old charcoal consumed by fires. The rate of charcoal formation is controlled by the CharcoalForm.prm file.

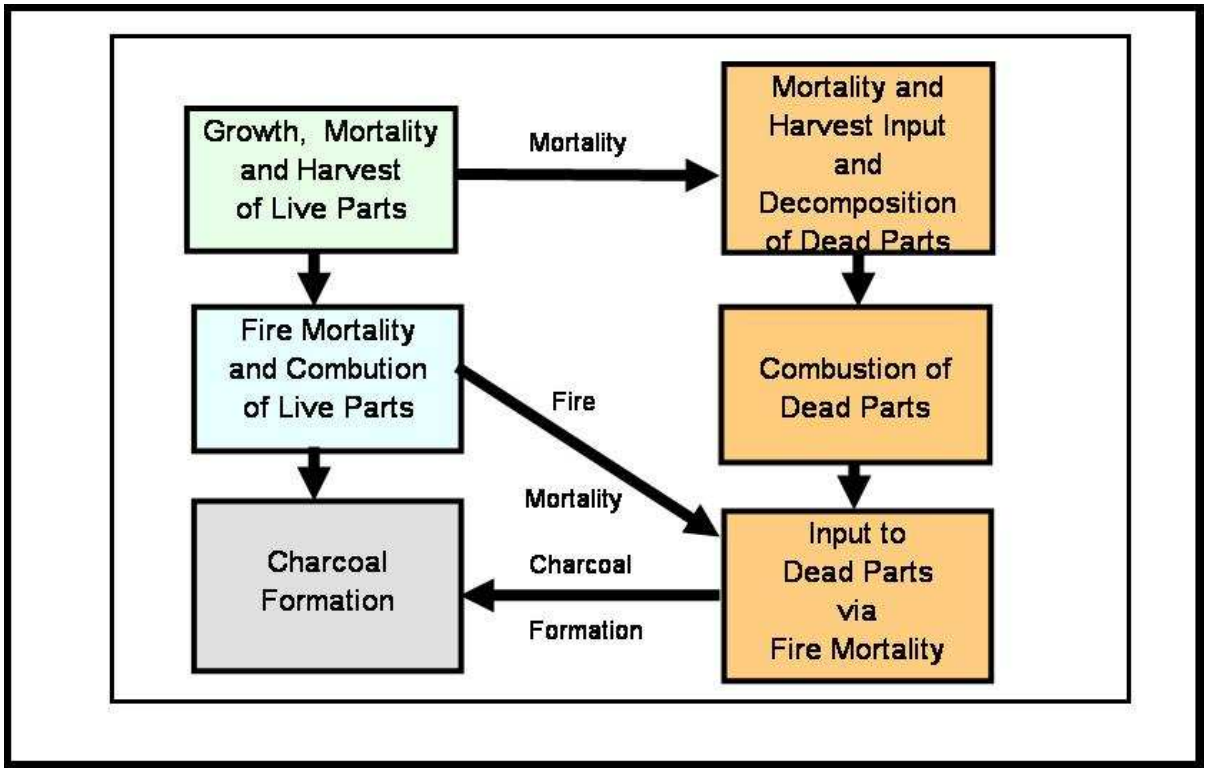

Figure 12-1. Processes occurring in the LANDCARB model when a prescribed fire occurs.

Prescribed fires can either create a new cohort (all the plant layers are killed) or if a new cohort is not created, then how existing cohorts are impacted. Specifically, to mimic a prescribed fire of a stand grid cell it is possible to use one of two options:

1) if plant layers are not completely killed by the prescribed fire, the new cohort area is set to 0, and the fire severity level indicates which plant layers, dead pools, stable pools, and

charcoal pools are influenced and what degree. All cohorts would be effected by this kind of prescribed fire;

2) if all the plant layers are killed by the prescribed fire, then area for a new cohort is created and plant layers are recolonized. Depending on the area of new cohort space created, some of the cohorts will be effected and other will not.

In the latter case the user needs to specify the number of patches the prescribed fire creates. There is a minimum number of patches that a prescribed fire creates. If half the stand is killed, then one needs to have at least two patches. If one-quarter is killed, then one needs at least four patches. The number of patches can be any multiple of the minimum number of patches. The number of patches determines the mean distance between cohorts and the light levels recieved, but does not influence the number of cohorts that are modeled.

LANDCARB uses the concept of a virtual gridwork of substand cells within stand grid cells (Figure 12-2). In STANDCARB this substand gridwok of cells is explicit. The number of cells in the LANDCARB substand gridwork is set by the minimum number of patches implied by the area of new cohort space created by the prescribed fire. If this disturbance pattern is repeated, then in time the number of cohorts in a stand grid cell will be the same as the minimum number of patches. This prevents a situation where the number of cohorts increases without limit with each disturbance cycle.

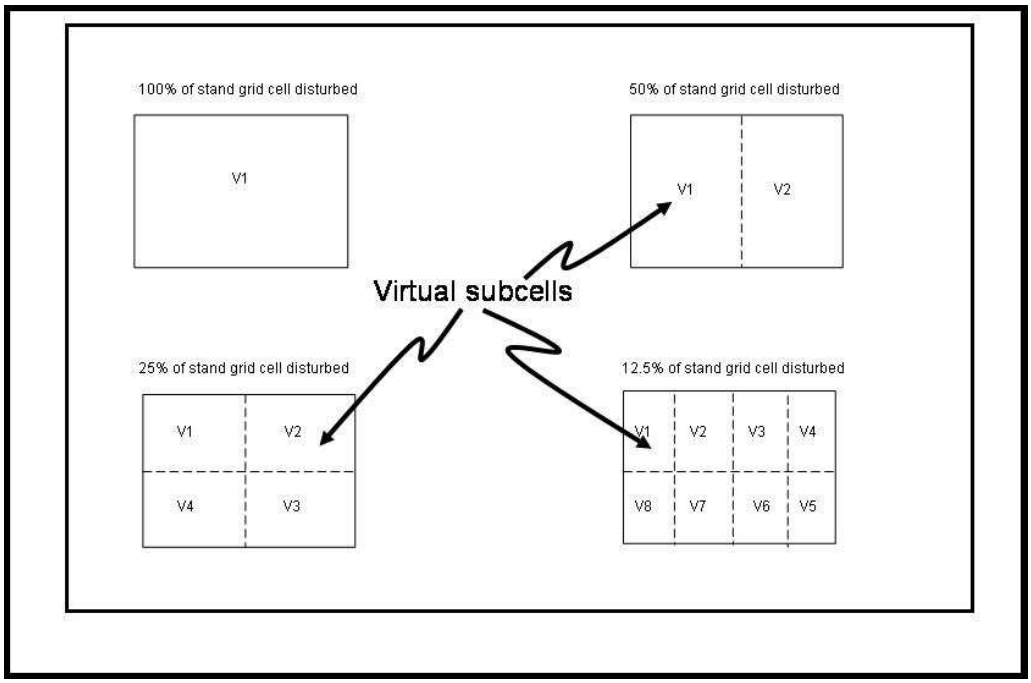

Figure 12-2. Examples of virtual subcells used by the LANDCARB model when a prescribed fire occurs in a stant grid cell.

When a prescribed fires occurs certain of the existing "patches" can be burned. The user sets whether the youngest, oldest, or a random set of patches is burned. If the oldest is selected, then an area equal to the "patch" size is killed by fire creating a new cohort. Suppose one starts with one cohort in a stand grid cell and 25% of the stand grid cell will be burned in 4 patches. The first round of prescribed fires would result in two cohorts; the original cohort would occupy 3 "virtual patches" and the new cohort would occupy 1 "patch". After the second round of prescribed fires there would be three cohorts, two occupying one "patch" and one occupying two "virtual patches". With another round of prescribed fires there would be four cohorts, each one occupying one "patch". Yet another round of prescribed fires and there still would be 4 cohorts each in one "patch", but the original cohort would have disappeared. In the case where the youngest cohort is selected then there would always be two cohorts, as the youngest is always burned. The first cohort would always occupy 3 "virtual patches" and the youngest one "patch".

If a harvest has occurred, a site preparation fire is set to occur after the harvest. If the prescribed fire is independent of the harvest, then it occurs the year it is scheduled. Moreover, the calculations occur after normal growth and decomposition calculations have been made for that year. Only one of three types of fires can occur in a given year: light, medium or hot fires.

The parameters controlling this module are found in the PrescribedFire.prm and CharcoalForm.prm files. The user selects the timing and type of prescribed fire in the PrescribedFireScheduler.drv files.

BurnKill function.

This function determines the proportion of each part of each layer remaining after a fire occurs. The first step is to determine when and what type of fire occurs in a stand grid cell. The fraction of the above-ground live mass of a part of a layer surviving (Surv*Part*) after a fire is:

Surv*Part*=(1-AboveKill/100)**Part*

where AboveKill is the percentage of above-ground parts killed by the fire (determined from the BurnKill.prm file), and *Part* is the mass of the above-ground part (i.e., foliage, branches, sapwood, and heartwood) in question. An exception occurs when AboveKill is 100% and it is determined that the tree in the cell sprouts (see Sprout module). In this case *Layer*Foliage is set to the same value as when a layer is planted. This allows the layer to begin growing again in the cell following the fire. These calculations are performed for each of the layers in a cell.

The fraction of the below-ground live mass of a part surviving (Surv*Part*) after a fire is:

Surv*Part*=(1-BelowKill/100)**Part*

where BelowKill is the fraction of below-ground parts killed by the fire (determined from the Prescribed Fire.prm file), and *Part* is the mass of the below-ground part (i.e., fine roots and coarse roots) in question. These calculations are performed for each of the plant layers.

The mass of above- and below-ground parts killed (Kill*Part*) is calculated as:

Kill*Part*=*Part*-Surv*Part*

where *Par*t refers to a specific plant part of a layer in a cell. This quantity is deducted from the live mass of the parts for each layer of each cell in the GROWTH module.

Live Consume Function.

This function calculates the of mass plant parts that is consumed by fire. Above- and below-ground parts have different portions of parts consumed by fire. For above-ground parts the mass consumed (Consum*Part*) is:

Consum*Part*=AboveBurn*Kill*Part*/100

where AboveBurn is the percentage of the above-ground parts that are killed by fire that are combusted (determined from the Prescribed Fire.prm file), and *Part* is the mass of the above-ground part (i.e., foliage, branches, sapwood, and heartwood) in question. These calculations are performed for each of the layers in a cell.

For below-ground parts the mass consumed (Consum*Part*) is:

Consum*Part*=BelowBurn*Kill*Part*/100

where BelowBurn is the percentage of the below-ground parts that are killed by fire that are combusted (determined from the BurnKill.prm file), and *Part* is the mass of the below-ground part (i.e., foliage, branches, sapwood, and heartwood) in question. These calculations are performed for each of the layers in a cell.

The mass of above- and below-ground parts added to the appropriate detrital pool in DECOMPOSE (BurnInput*Pool*) is calculated as:

BurnInput*Pool*=Kill*Part*-Consum*Part*

where *Pool* refers to a specific detrital pool in a cell. This quantity is added to the appropriate detrital pool of each cell in the DECOMPOSE module.

Dead/Stable/Charcoal Consume Function.

This function determines the proportion of each dead, stable, and charcoal pool remaining after a prescribed fire occurs. The first step is to determine when and what type of prescribed fire occurs in a stand grid cell. To calculate the amount removed in each of the pools after fire (*Pool*FireLoss), the fraction remaining is multiplied by the mass of the pool:

*Pool*FireLoss=(1-*Burn*Remaining/100)**Pool*

where *Pool* is the mass of a given detrital pool (i.e., dead foliage, dead branches, dead sapwood, dead heartwood, dead fine roots, dead coarse roots, or stable soil), *Burn*Remaining is the percent remaining after a fire of type *Burn*. The latter parameter is determined from the PrescribedFire.prm file.

If a fire does not occur in a given stand grid cell on a given year then

*Pool*FireLoss=0.

The pool mass is then reduced to account for these fire losses:

*Pool*New=*Pool*Old-*Pool*FireLoss

where *Pool*New and *Pool*Old are the pool mass after and before the fire, respectively.

Charcoal Formation Function.

This function computes the amount of charcoal that is formed by prescribed fire. Live parts, dead pools, and stable pools are all potentially capable of forming charcoal, although it should be noted that non-woody materials are unlikely to form charcoal. The amount of charcoal formed from different parts and pools is varied by fire severity and is defined in the CharcoalFormation.prm file.

The mass of charcoal created from live parts is:

*Part*SurfaceCharcoalInput=CharcoalFormationRate* Kill*Part*/100

The mass of charcoal created from dead and stable pools is:

*Pool*SurfaceCharcoalInput=(*Burn*Remaining/100)**Pool*

The total mass of charcoal formed is:

SurfaceCharcoalInput= Σ*Part*SurfaceCharcoalInput+ Σ*Pool*SurfaceCharcoalInput

Fuel Load

This module creates an overall fuel loading that includes live and dead carbon pools. Each pool is given a weighting based on the relative contribution of a given pool to killing vegetation and/or causing other pools to burn. The store of each pool is multiplied by the weighting factor (see BurnDead.prm) to calculate the pool fuel load:

PoolFuelLoad=FuelWeightingFactor*Pool

The total fuel load is calculated by summing all the pool fuel loads:

FuelLoad= $\sum$ PoolFuelLoad

The fuel load is a relative index and depends on the weighting factors used. The fuel loads calculated need to be scaled appropriately to the fuel laod limits associated with given levels of fire severity.

Fire Severity-Fuel Load Feedback

Prescribed fire severity may be altered from the target value if the fuel load is lower or higher than the target range appropriate for the target severity (Figure 12-3). When the severity-fuel feedback is switched on in the Simul.drv file, then the fuel load is compared to the range appropriate for the target fire severity. If the fuel load is lower than expected for a given severity, then there is a chance that fire severity will decrease. Likewise, if the fuel load is higher than expected for a given fire severity, then there is a chance the fire severity will increase. The underlying probability of change is set in the BurnDead.prm file. The farther the fuel loading is from the target range, the higher the chance fire severity will change. A similar process is followed for wildfires. However, to account for the fact that weather can be controlled to a higher degree in prescribed fires than wildfires, the probability of changing fire severity should be substantially lower for prescribed fires than wildfires.

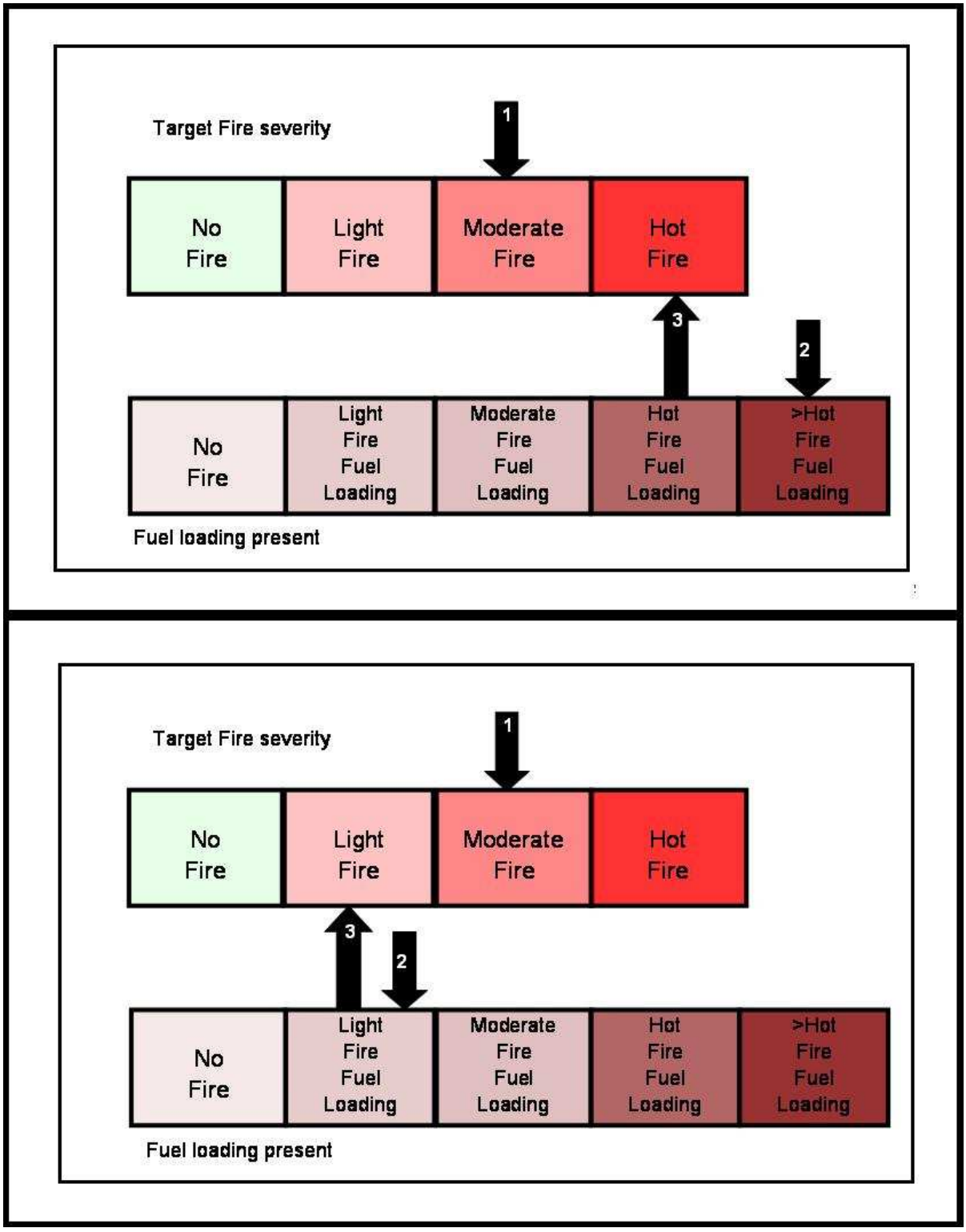

Figure 12-3. Conceptual examples of how the LANDCARB model modifies the realized fire severity from the initial target fire severity. 1- A target fire severity is selected; 2-the fuel loading is compared to the range for the target; and 3- the final fire severity level is adjusted up or down depending on the fuel load.

13-WILDFIRE

This module determines the effect of wildfire on plant layers, dead pools, stable pools, and charcoal pools. For live vegetation it determines the amount that is killed and consumed by a fire and modifies the amount of live carbon in the GROWTH module and transfers some of this material to the DECOMPOSE module as fire-killed dead pool inputs (Figure 13-1). Not all the live vegetation killed by fire is necessarily transferred to dead pools; some is consumed by the fire itself. For dead pools this module reduces the amount of dead material in the DECOMPOSE module to reflect the losses caused by fire. The changes in live parts, dead and stable pools is described in the Wildfire.prm file. The WildFire.prm file is set up so that as fire severity increases from light to hot, the fraction each live part or dead pool killed or removed by fire increases. This module also controls the amount of charcoal this formed from live and dead pools and amount of old charcoal consumed by fires. The rate of charcoal formation is controlled by the CharcoalForm.prm file.

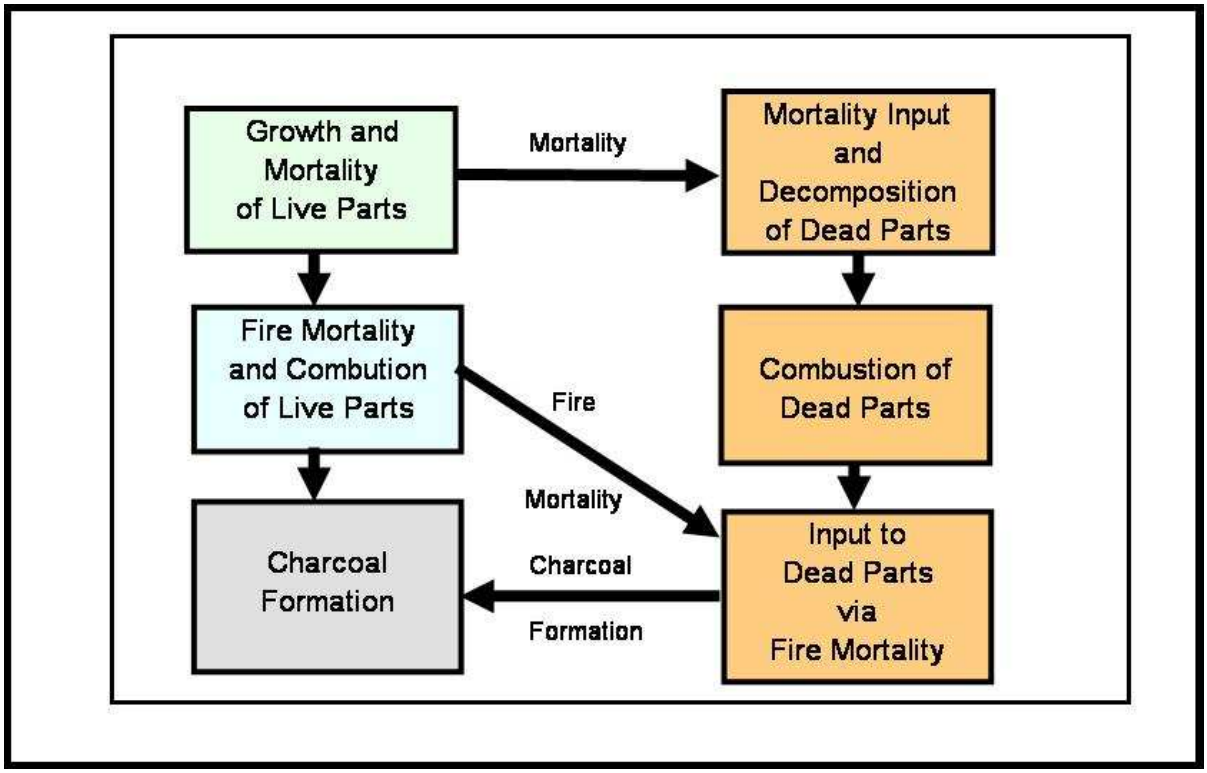

Figure 13-1. Processes occurring in the LANDCARB model when a wildfire occurs.

Wild fires can either create a new cohort (all the plant layers are killed) or if a new cohort is not created, then how existing cohorts are impacted. Specifically, to mimic a wild fire in a stand grid cell it is possible to use one of two options:

1) if plant layers are not completely killed by the wild fire, the new cohort area is set to 0, and the fire severity level indicates which plant layers, dead pools, stable pools, and charcoal pools are influenced and what degree. All cohorts would be effected by this kind of wild fire;

2) if all the plant layers are killed by the wild fire, then area for a new cohort is created and plant layers are recolonized. Depending on the area of new cohort space created, some of the cohorts will be effected and other will not.

In the latter case the user needs to specify the number of patches the wild fire creates. There is a minimum number of patches that a wild fire creates. If half the stand is killed, then one needs to have at least two patches. If one-quarter is killed, then one needs at least four patches. The number of patches can be any multiple of the minimum number of patches. The number of patches determines the mean distance between cohorts and the light levels recieved, but does not influence the number of cohorts that are modeled.

LANDCARB uses the concept of a virtual gridwork of substand cells within stand grid cells (Figure 13-2). In STANDCARB this substand gridwork of cells is explicit. The number of cells in the LANDCARB substand gridwork is set by the minimum number of patches implied by the area of new cohort space created by the wild fire. If this disturbance pattern is repeated, then in time the number of cohorts in a stand grid cell will be the same as the minimum number of patches. This prevents a situation where the number of cohorts increases without limit with each disturbance cycle.

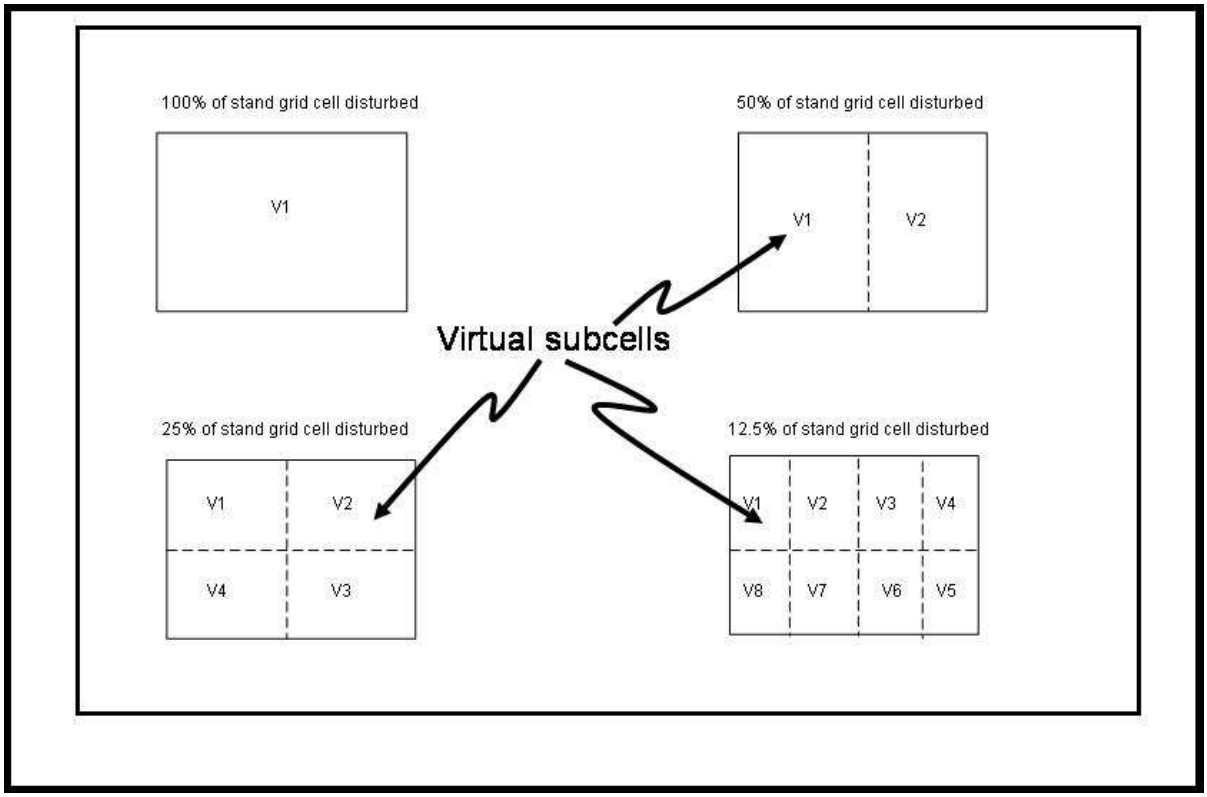

Figure 12-3. Examples of virtual subcells used by the LANDCARB model when a wildfire occurs in a stant grid cell.

When a wild fires occurs certain of the existing “patches” can be burned. The user sets whether the youngest, oldest, or a random set of patches is burned. If the oldest is selected, then an area equal to the “patch” size is killed by fire creating a new cohort. Suppose one starts with one cohort in a stand grid cell and 25% of the stand grid cell will be burned in 4 patches. The first round of wild fires would result in two cohorts; the original cohort would occupy 3 “virtual patches” and the new cohort would occupy 1 “patch”. After the second round of wild fires there would be three cohorts, two occupying one “patch” and one occupying two “virtual patches”. With another round of wild fires there would be four cohorts, each one occupying one “patch”. Yet another round of wild fires and there still would be 4 cohorts each in one “patch”, but the original cohort would have disappeared. In the case where the youngest cohort is selected then there would always be two cohorts, as the youngest is always burned. The first cohort would always occupy 3 “virtual patches” and the youngest one “patch”.

The calculations for this module occur after normal growth and decomposition calculations have been made for that year. Only one of three types of fires can occur in a given year: light, medium or hot fires.

The parameters controlling this module are found in the WildFire.prm and CharcoalForm.prm files. The user selects the timing and type of wild fire in the WildFireSpread.drv files.

BurnKill function.

This function determines the proportion of each part of each layer remaining after a fire occurs. The first step is to determine when and what type of fire occurs in a stand grid cell. The fraction of the above-ground live mass of a part of a layer surviving (Surv*Part*) after a fire is:

Surv*Part*=(1-AboveKill/100)**Part*

where AboveKill is the percentage of above-ground parts killed by the fire (determined from the BurnKill.prm file), and *Part* is the mass of the above-ground part (i.e., foliage, branches, sapwood, and heartwood) in question. An exception occurs when AboveKill is 100% and it is determined that the tree in the cell sprouts (see Sprout module). In this case *Layer*Foliage is set to the same value as when a layer is planted. This allows the layer to begin growing again in the cell following the fire. These calculations are performed for each of the layers in a cell.

The fraction of the below-ground live mass of a part surviving (Surv*Part*) after a fire is:

Surv*Part*=(1-BelowKill/100)**Part*

where BelowKill is the fraction of below-ground parts killed by the fire (determined from the Wild Fire.prm file), and *Part* is the mass of the below-ground part (i.e., fine roots and coarse roots) in question. These calculations are performed for each of the plant layers.

The mass of above- and below-ground parts killed (Kill*Part*) is calculated as:

Kill*Part*=*Part*-Surv*Part*

where *Par*t refers to a specific plant part of a layer in a cell. This quantity is deducted from the live mass of the parts for each layer of each cell in the GROWTH module.

Live Consume function.

This function calculates the of mass plant parts that is consumed by fire. Above- and below-ground parts have different portions of parts consumed by fire. For above-ground parts the mass consumed (Consum*Part*) is:

Consum*Part*=AboveBurn*Kill*Part*/100

where AboveBurn is the percentage of the above-ground parts that are killed by fire that are combusted (determined from the Wild Fire.prm file), and *Part* is the mass of the above-ground part (i.e., foliage, branches, sapwood, and heartwood) in question. These calculations are performed for each of the layers in a cell.

For below-ground parts the mass consumed (Consum*Part*) is:

Consum*Part*=BelowBurn*Kill*Part*/100

where BelowBurn is the percentage of the below-ground parts that are killed by fire that are combusted (determined from the BurnKill.prm file), and *Part* is the mass of the below-ground part (i.e., foliage, branches, sapwood, and heartwood) in question. These calculations are performed for each of the layers in a cell.

The mass of above- and below-ground parts added to the appropriate detrital pool in DECOMPOSE (BurnInput*Pool*) is calculated as:

BurnInput*Pool*=Kill*Part*-Consum*Part*

where *Pool* refers to a specific detrital pool in a cell. This quantity is added to the appropriate detrital pool of each cell in the DECOMPOSE module.

Dead/Stable/Charcoal Consume Function.

This function determines the proportion of each dead, stable, and charcoal pool remaining after a wildfire occurs. The first step is to determine when and what type of wildfire occurs in a stand grid cell. To calculate the amount removed in each of the pools after fire (*Pool*FireLoss), the fraction remaining is multiplied by the mass of the pool:

*Pool*FireLoss=(1-*Burn*Remaining/100)**Pool*

where *Pool* is the mass of a given detrital pool (i.e., dead foliage, dead branches, dead sapwood, dead heartwood, dead fine roots, dead coarse roots, or stable soil), *Burn*Remaining is the percent remaining after a fire of type *Burn*. The latter parameter is determined from the PrescribedFire.prm file.

If a fire does not occur in a given stand grid cell on a given year then

*Pool*FireLoss=0.

The pool mass is then reduced to account for these fire losses:

*Pool*New=*Pool*Old-*Pool*FireLoss

where *Pool*New and *Pool*Old are the pool mass after and before the fire, respectively.

Charcoal Formation Function.

This function computes the amount of charcoal that is formed by wildfire. Live parts, dead pools, and stable pools are all potentially capable of forming charcoal, although it should be noted that non-woody materials are unlikely to form charcoal. The amount of charcoal formed from different parts and pools is varied by fire severity and is defined in the CharcoalFormation.prm file.

The mass of charcoal created from live parts is:

*Part*SurfaceCharcoalInput=CharcoalFormationRate* Kill*Part*/100

The mass of charcoal created from dead and stable pools is:

*Pool*SurfaceCharcoalInput=(*Burn*Remaining/100)**Pool*

The total mass of charcoal formed is:

SurfaceCharcoalInput= Σ*Part*SurfaceCharcoalInput+ Σ*Pool*SurfaceCharcoalInput

Fuel Load

This module creates an overall fuel loading that includes live and dead carbon pools. Each pool is given a weighting based on the relative contribution of a given pool to killing vegetation and/or causing other pools to burn. The store of each pool is multiplied by the weighting factor (see BurnDead.prm) to calculate the pool fuel load:

PoolFuelLoad=FuelWeightingFactor*Pool

The total fuel load is calculated by summing all the pool fuel loads:

FuelLoad= $\sum$ PoolFuelLoad

The fuel load is a relative index and depends on the weighting factors used. The fuel loads calculated need to be scaled appropriately to the fuel laod limits associated with given levels of fire severity

Fire Severity-Fuel Load Feedback

Prescribed fire severity may be altered from the target value if the fuel load is lower or higher than the target range appropriate for the target severity (Figure 13-3). When the severity-fuel feedback is switched on in the Simul.drv file, then the fuel load is compared to the range appropriate for the target fire severity. If the fuel load is lower than expected for a given severity, then there is a chance that fire severity will decrease. Likewise, if the fuel load is higher than expected for a given fire severity, then there is a chance the fire severity will increase. The underlying probability of change is set in the BurnDead.prm file. The farther the fuel loading is from the target range, the higher the chance fire severity will change. A similar process is followed for wildfires. However, to account for the fact that weather can be controlled to a higher degree in prescribed fires than wildfires, the probability of changing fire severity should be substantially higher for wildfires than prescribed fires.

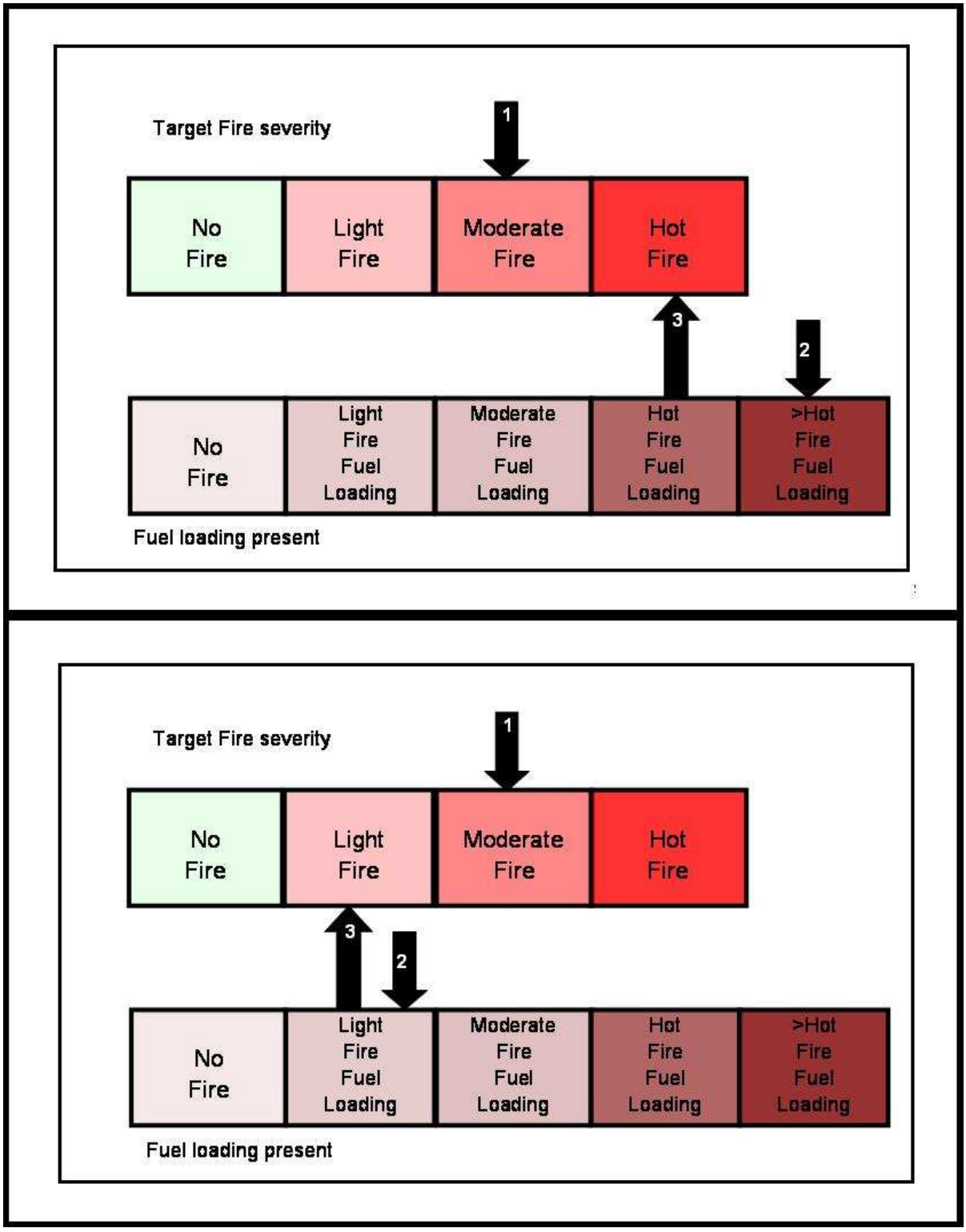

Figure 13-3. Conceptual examples of how the LANDCARB model modifies the realized fire severity from the initial target fire severity. 1- A target fire severity is selected; 2-the fuel loading is compared to the range for the target; and 3- the final fire severity level is adjusted up or down depending on the fuel load.

14-Forest Products

This module processes the harvested carbon from grid cells into forest wood products that are used and disposed (Figure 14-1). In addition, some of the harvest can be processed for either internal or external bioenergy. This model is patterned after the FORPROD model (Harmon et al. 1996), although some pools have been combined and bioenergy has been added. This module operates on an annual time step. In addition to being able to process the harvested carbon directly, the amount of harvest carbon is also output as well in case the user wishes to process it using a stand-alone model.

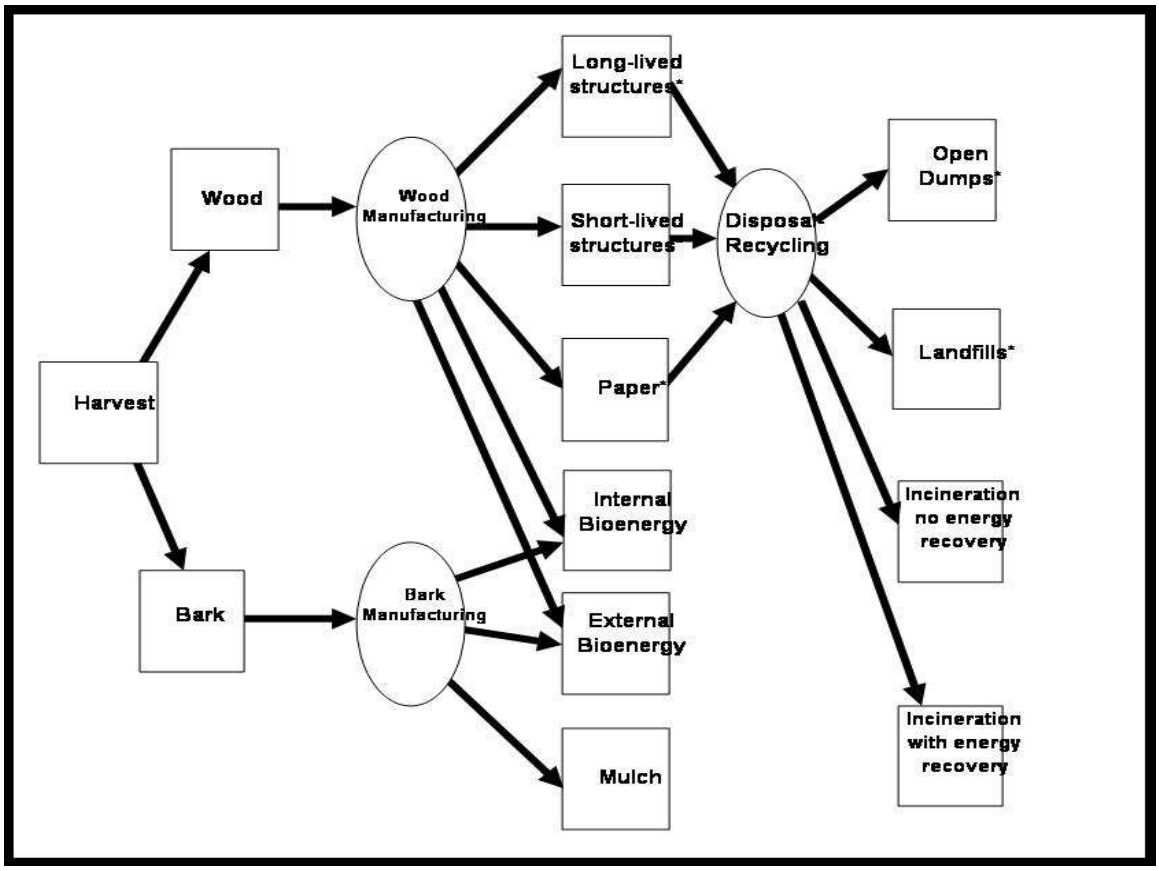

Figure 14-1. The forest products related pools and flows tracked by the LANDCARB model.

The files required to run this module include Manufacturing.prm which determines how the harvest is processed during the manufacturing step, ProductsUse.prm which determines how products are used and their life-span, and Disposal.prm which determines how products are disposed of once their life-span is exceeded. For each of these files there is a default set of parameters; however it is also possible to provide yearly changes in the parameters to reflect changes in manufacturing, use, and disposal practices. Each year need not be specified as the module will change the values when a new year is encountered.

When there is a harvest in a stand grid cell, a portion of the live wood (sapwood and heartwood) can be removed and processed into forest products. The harvest is first processed in the manufacturing step to produce inputs for the different product stores such as long-term structures, short-term structures, paper, and mulch. Bioenergy can be produced during manufacturing some of which is used by the forest sector (internal) and some of which is used outside the forest sector (external). Only the external bioenergy is counted as a potential fossil fuel offset. Once the new product inputs been accounted for product stores are then followed with losses due to fire and decomposition as well as those from disposal. As products are disposed, a proportion is recycled. This effectively reduces disposal rate. Disposed products can be sent to open dumps (high combustion and decomposition rates), landfills (no combustion and very low decomposition rates), incinerated with or without energy recovery. In the case of the former a potential fossil fuel offset is counted.

Bark-Wood Separation. The first step is to compute the amount wood versus bark in the harvested carbon:

HarvestWood=(WoodPercent/100)*Harvest

Where HarvestWood is the amount of harvested wood, WoodPercent is found in the Growth.prm file, and Harvest is the amount of carbon harvested in a grid cell. Bark mass is:

HarvestBark=Harvest-HarvestWood

Bark Allocation. Harvested bark can be allocated into mulch, internally used biofuels or externally used biofuels:

MulchBark=BarkToMulchFraction*HarvestBark

ExternalBiofuelBark= ExternalBiofuelFraction*HarvestBark

InternalBiofuelBark=HarvestBark- MulchBark- ExternalBiofuelBark

Wood Allocation. The next step is to allocate the wood into different manufacturing streams (structural, pulp, or bioenergy) which generically looks like this:

*Stream*ProductsInput= *Stream*ManEffic**Stream*Wood

Where *Stream*ManEffic is the efficiency in creating the type of product and *Stream*Wood is the amount of wood entering the given manufacturing stream (StructuralWood, PulpWood, ExternalBiofuel).

It is assumed that harvested materials used for bioenergy have fixed conversion rate to a carbon offset; this number is less than 1.0 (as biofuel contains less energy per unit carbon

than fossil fuels) and is set lower when liquid bioenergy is created and higher when solid fuels are created.

Wood Manufacturing. The next step is to determine the fate of harvested carbon during manufacturing. Wood being processed for structural wood (i.e., lumber, plywood, and engineered materials) can end up as structural materials, chips for paper manufacturing, or internal biofuels (hogg fuel). For structural related products produced during manufacturing the generic equation is:

ProductWood= WoodToProductFraction*ManStructuralWood

Where ProductWood is the mass of structural, hogg fuel, or chips products, WoodToProductFraction is the fraction of wood converted to a given product, and ManStructuralWood is the mass of wood allocated to structural manufacturing.

The specific equations for each product are:

StructuralWood= WoodToStructureFraction*ManStructuralWood

HoggFuelWood= WoodToHoggFuelFraction*ManStructuralWood

ChipsWood= WoodToChipsFraction*ManStructuralWood

The next step in manufacturing is to produce paper stock is based on the amount of wood allocated to paper production and the amount of chips created from structural manufacturing. A fraction of chip-related carbon is lost during manufacturing used as an internal biofuel or decomposed on site in effluent treatment plants.

PaperStock=PaperManEffic*(ManPulpWood+ChipsWood)

Where PaperManEffic is the fraction of chips converted to paper stock. In addition, some internal bioenergy is created during paper manufacturing:

PulpBioEnergy= PulpBioFuelConversion*(ManPulpWood+ChipsWood)

Where PulpBioFuelConversion is the amount of wood chips and pulp converted to bioenergy that is used within the wood products sector.

Wood biofuels used external to the forest sector are computed as:

BiofuelExternalWood =ExternalBiofuelFraction*HarvestWood

Products Inputs. The next step is to compute the products in-use input flows:

LongTermStructureInput=Long-termStructureFraction* StructuralWood

Where LongTermStructureFraction is the fraction of structural wood manufacturing outputs allocated to long-lived structures.

ShortTermStructureInput= StructuralWood- LongTermStructureInput

PaperInput=PaperStock

MulchInput=MulchBark

BiofuelInternalInput= InternalBiofuelBark + PulpBioEnergy + HoggFuelWood

BiofuelExternalInput= BiofuelConvEffic*(ExternalBiofuelBark + BiofuelExternalWood)

Where BiofuelConvEffic is the efficiency of biofuels relative to fossil fuel energy (biofuels contain less energy per unit carbon than fossil fuels).

Product Losses In-Use. Once manufacturing steps are complete newly formed products are added to existing wood product stores including: mulch, long-term structures (average life-span > 30 years), short-term structures (average life-span < 30 years), paper, and external biofuels. While external biofuels are not a store per se, they are counted as one in the form of a fossil fuel offset. Each of these stores is potentially subject to losses via disposal and combustion/decomposition. In the case of mulch it is assumed that only decomposition losses occur. Internal biofuels have 100% loss each year. For external biofuels it is assumed that there are no losses (although in theory if the fossils fuels they are offsetting are eventually used then there is some degree of "decomposition" of their value). For long-term structures, short-term structures, and paper there are losses from both disposal and combustion/decomposition.

Generically the mass of forest products each year is computed as:

ForestProductsNew=
ForestProductsOld-ΣAverageLossRate*ForestProductsOld+ForestProductsInput

Where ForestProductsOld is the value from the year before and LossRate is the proportion lost per year for the various loss paths (e.g., disposal or decomposition) and ForestProductsInput is the amount of a particular forest product being produced via manufacturing. If a harvest has never occurred in a stand grid cell, then the forest products store is set to zero.

Disposal Allocation. Once products are disposed of, they can have different final "fates" or disposal sites: open dumps (Dumps), landfills (LandFill), incineration without energy recovery (Incin) and incineration with energy recovery (IncinBioenergy). The general equation for all disposal streams is:

ProductToDisposalSite=DisposalSiteFraction* NetProductDisposalLoss

Where DisposalSiteFraction is the fraction allocated to the different disposal sites. For disposal losses there is a potential to reduce the flow to disposal sites by recycling. Effectively recycling reduces the disposal rate. However, recycling can not reduce the flow to disposal sites to zero because some recycled materials are not suitable for reuse. Therefore this fraction of loss is deducted from the planned recycling rate.

Disposal Site Losses. Disposal sites include open dumps, landfills, incineration without bioenergy recovery, and incineration with bioenergy recovery. Open dumps and landfills are both subject to decomposition/combustion losses. Open dumps differ from landfills in that they are subject to high losses from decomposition and combustion. Although decomposition in landfills is quite slow, some of this decomposition results in production of methane. However, this is not captured in the current module so that the carbon balance can be computed (as opposed to the greenhouse gas balance). Incineration without bioenergy recovery is assumed to release carbon within one year of disposal. Incineration with bioenergy recovery is assumed to have no losses, although as noted above there can be some loss in offset value if the fossil fuels are actually used.

The general equation describing these losses is:

DisposalSiteLoss=DisposalSiteLossRateConstant*DisposalSiteOld

Where DisposalSiteLossRateConstant is the fraction lost in a year and DisposalSiteOld is the store in the disposal site the previous year. For incineration without energy recovery the DisposalSiteLossRateConstant is set to 1.0, whereas for incineration with bioenergy recovery it is set to 0.

The mass of forest products in disposal each year is computed as:

DisposalSiteNew=
DisposalSiteOld- DisposalSiteLoss + ProductToDisposalSite

Where DisposalSite Old is the value from the year.

Average Loss Rate Calculations. In LANDCARB the rate of losses from decay and disposal can change over time. Since the manufacturing and use parameters can change over time and products produced from one period can exist with another, the loss rate parameters represent a weighted average of the previous and new parameter values. The AverageLossRate is therefore a weighted average of the previous value and the input value.

AverageLossRate=
(PreviousAverageLossRate*ForestProductOld +InitialLossRate*InitialForestProducts)/
(ForestProductsOld+ InitialForestProducts)

Where PreviousAverageLossRate and InitialLossRate represent the fraction lost in a year for the existing forest products and the new harvest, respectively. ForestProductOld and InitialForestProducts represent the forest products store the previous year and the input from the most recent harvest, respectively. The weighted average allows the rate of forest products loss to change over time.

16-HARVEST SCHEDULER

The scheduling of harvests with or without a prescribed fire is determined by this module. Note that this process takes place outside the LANDCARB model, and although this model does have this capability, a simpler system is used for the Forest Sector Carbon Calculator.

Stand Level. At the stand level, all the grid cells are harvested in harvest years. For the past, the user supplies the year that past harvest events impacted the stand. For future harvests the user supplies the interval between harvest events. This information is used by a scheduler program to determine the years in which harvest occur. Harvests always occur exactly on the specified year, but within a year the percent of the stand disturbed (i.e., percent disturbed parameter) may be adjusted among cells to achieve an approximation of the target disturbance. For example, when the parameter Percent Disturbed is set at 100%, then the interval not affected by the value of Percent Disturbed. Moreover, all the stand grid cells will be simultaneously impacted. This is also true when this parameter is set to a value that can sum to 100% (e.g., 10, 20, and 50%). This is because each grid cell can be divided up into a number of parts that eventually will cover the entire grid cell exactly. However, when the parameter Percent Disturbed is set to a number that will not sum exactly to 100% (e.g., 45%) then the grid cells will receive variable amounts of harvest to achieve the desired value of Percent Disturbed.

Landscape Level. At the landscape level not all stands will have harvests the same year. It is assumed that there are sufficient stands to maintain a regular spacing between harvests. However, when harvest intervals are extremely long (i.e., longer than the number of grid cells in a landscape), then in some years harvest will not be possible. To determine the cells to be harvested, the program first determines the average number of entire cells that need to be harvested to achieve the desired regular interval of harvests. When the total number of stand grid cells being simulated cannot be divided evenly by this number of stands then the number of stands harvested each year varies. However, the program tries to keep this as close to the target as possible.

At the landscape level stands that have recently been disturbed by wildfire will not be harvested; there is a minimum stand age that is subject to harvest. This is currently set at no younger than 90% of the target harvest interval.

Stands in a landscape are harvested sequentially in the overall grid work of stands, but this has little practical impact because stands are independent of each other. To achieve the values specified in the Percent Disturbed parameter, the landscape level uses the same procedure as at the stand level. That is, it allocates harvests to stands so as to achieve as close to the desired percent as possible. **Note that at the landscape level, Percent Disturbed is implemented within stand grid cells. It does not remove that proportion of stand grid cells from disturbance.** Therefore all cells will eventually be disturbed. If one wishes to have a landscape in which there are different harvest regimes in different locations, then the best way to achieve this is to run a simulation for each

management system and then to combine them after the fact with an area weighted average.

Different harvests regimes may be linked within stands at the landscape level. For example, if thinnings occur in stands between clear-cut harvest, then these harvest regimes can be coupled.

Checking the Realized Regime. The degree to which the target intervals and Percent Disturbed are achieved at either the stand or landscape level can be checked by looking at the Harvest Events graph or by Run Output file under Output Files-Log Files window.

17-PRESCRIBED FIRE SCHEDULER

The scheduling of prescribed fires with or without a harvest is determined by this module. Note that this process takes place outside the LANDCARB model, and although this model does have this capability, a simpler system is used for the Forest Sector Carbon Calculator.

Stand Level. At the stand level, all the grid cells are burned in years with prescribed fires. For the past the user supplies the year that past prescribed fire events impacted the stand. For future prescribed fires the user supplies the interval between prescribed fire events. This information is used by a scheduler program to determine the years in which prescribed fires occur. Prescribed fires always occur exactly on the specified year, but within a year the percent of the stand disturbed (i.e., percent disturbed parameter) may be adjusted among cells to achieve an approximation of the target disturbance. For example, when the parameter Percent Disturbed is set at 100%, then the interval is not affected by the value of Percent Disturbed. Moreover, all the stand grid cells will be simultaneously impacted. This is also true when this parameter is set to a value that can sum to 100% (e.g., 10, 20, and 50%). This is because each grid cell can be divided up into a number of parts that eventually will cover the entire grid cell exactly. However, when the parameter Percent Disturbed is set to a number that will not sum exactly to 100% (e.g., 45%) then the grid cells will receive variable amounts of prescribed fire to achieve the desired value of Percent Disturbed.

Landscape Level. At the landscape level not all stands will have prescribed fires the same year. It is assumed that there are sufficient stands to maintain a regular spacing between prescribed fires. However, when prescribed fire intervals are extremely long (i.e., longer than the number of grid cells in a landscape), then in some years prescribed fires will not be possible. To determine the cells to be burned, the program first determines the average number of entire cells that need to be burned to achieve the desired regular interval of prescribed fires. When the total number of stand grid cells being simulated cannot be divided evenly by this number of stands then the number of stands burned each year varies. However, the program tries to keep this as close to the target as possible. Stands are burned sequentially in the overall grid work of stands, but this has little practical impact because stands are independent of each other. To achieve the values specified in the Percent Disturbed parameter, the landscape level uses the same procedure as at the stand level. That is, it allocates prescribed fires to stands so as to achieve as close to the desired percent as possible. **Note that at the landscape level, Percent Disturbed is implemented within stand grid cells. It does not remove that proportion of stand grid cells from disturbance.** Therefore all cells will eventually be disturbed. If one wishes to have a landscape in which there are different prescribed fire regimes in different locations, then the best way to achieve this is to run a simulation for each management system and then to combine them after the fact using an area weighted average.

Harvests and prescribed fires can be linked by the user by selecting a Harvest and Burn disturbance regime. In this case the prescribed fire will follow the harvest. If a harvest

and burn treatment has occurred, then a subsequent prescribed fire may not occur if the interval is less than 90% of the planned interval.

Checking the Realized Regime. The degree to which the target intervals and Percent Disturbed are achieved at either the stand or landscape level can be checked by looking at the Prescribed Fire Events graph or by Run Output file under Output Files-Log Files window.

Notes: Do we link prescribed fires to harvests? Is there a minimum age for prescribed fires?

18-WILDFIRE SCHEDULER

The scheduling of wildfires is determined by this module. Note that this process takes place outside the LANDCARB model, and although this model does have this capability, a simpler system is used for the Forest Sector Carbon Calculator. Wildfire presence in a grid cell is not restricted either by stand age or previous harvest, prescribed fire, or wildfire. This is not to say that these factors have no impact on fires in terms of severity; however, these interactions are determined in the WILDFIRE module. At both the stand and landscape level the natural fire return interval and fire size distribution for a region, elevation band, and ownership is used to randomly determine when wildfires will occur and how large they will be

.

Stand Level. At the stand level, the user supplies information about past wildfire events that impacted the stand. Future wildfires occur randomly, although the user specifies a level of wildfire suppression that controls the probability of this disturbance. For example, if total suppression is indicated, then no wildfires will occur in the future. If no suppression is indicated, then the natural fire cycle for that location will occur. Finally, if typical fire suppression is indicated, then the average interval between wildfires is doubled. This information is used by a program that schedules future wildfires. When a wildfire occurs at the stand level, all stand grid cells are simultaneously affected.

Landscape Level. At the landscape level not all stand grid cells will have wildfires the same year. Wildfires occur at random at the landscape level. To determine the stand grid cells to be burned, the program first determines how many of the stand grid cells must be burned during a time interval (e.g., 10 years) to achieve the desired average return interval of wildfires fires. This sets a total wildfire area for that time period. The program then randomly draws a series of fires of different sizes from a wildfire size distribution until the total wildfire area for a period is reached. These wildfires are then placed randomly in time over the time interval.

For wildfires the Percent Disturbed parameter is set at 100%; which means that almost all the stand grid cells specified in a given year will be equally impacted. A possible exception is the last cell in a given year that is burned. For this cell the Percent Disturbed may be set at a lower value to achieve the total wildfire area that was specified.

At the landscape level, the stand grid cell size interacts with the wildfire size distribution. For example, if the stand grid cell area is set larger than most wildfires, the program will set the Percent Disturbed so the wildfires smaller than the stand grid cell can occur, although wildfires that are less than 10% of the stand grid cell are not allowed to occur. If the stand grid cell size is set lower than most wildfires, then it is possible that the entire landscape will be burned at once and will act essentially like a stand. Regardless of stand grid cell size, a very large fire has the potential to burn over the entire landscape. If one wishes to have a landscape in which there are different wildfire regimes in different locations, then the best way to achieve this is to run a simulation for each fire regime and then to combine them after the fact by using an area weighted average.

Checking the Realized Regime. The degree to which the target intervals are achieved at the stand or landscape level can be checked by looking at the Wildfire Events graph or by Run Output file under Output Files-Log Files window.

19-Ecosystem Fluxes

To make comparisons with other models of carbon dynamics LANDCARB produces additional output variables related to ecosystem fluxes including gross primary production (GPP), net primary production (NPP), autotrophic respiration (Ra), mortality (M), heterotrophic respiration (Rh), and net ecosystem production (NEP). An example of how these fluxes or flows change over time is presented in Figure 19-1.

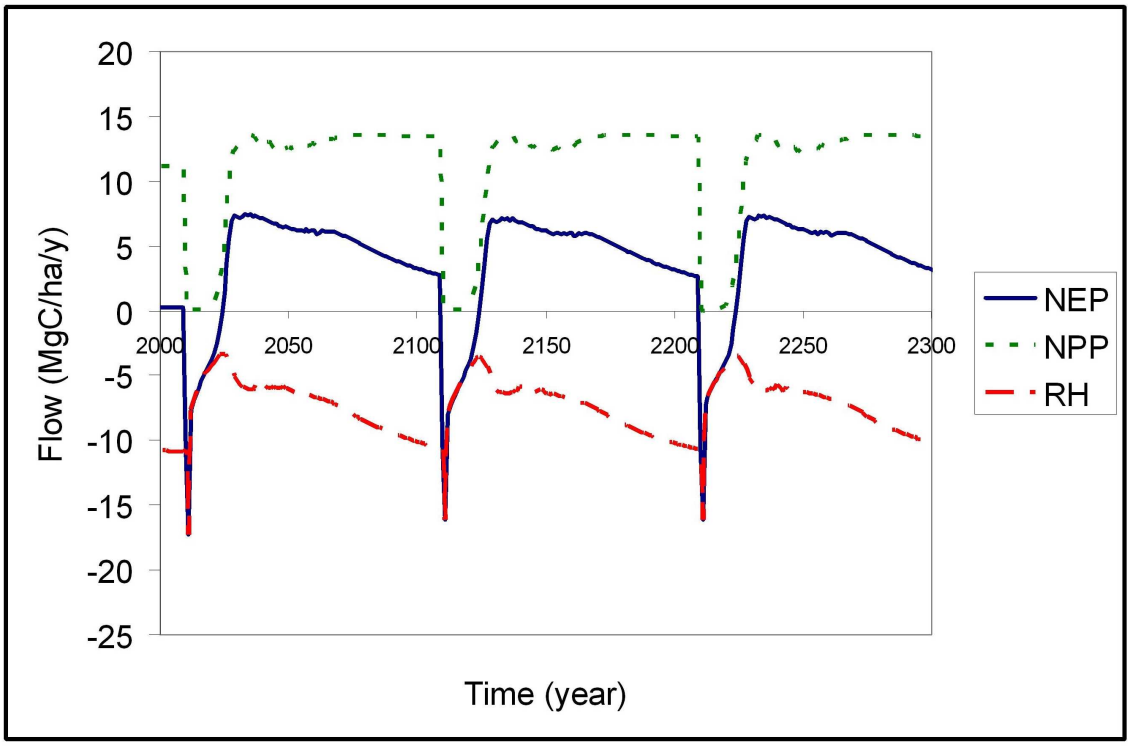

Figure 19-1. Changes in NPP, RH, and NEP in a forest after harvesting as predicted by the LANDCARB model.

NPP or Net Primary Production

NPP is calculated for all pools except the foliage, heartwood, and heart-rot as:

LayerPartNPP=LayerPartAlloc-LayerPartResp

Heartwood and heart-rot are not included in the NPP calculation because allocation to these pools does not involve the creation of new carbon or organic matter. For the foliage we need to use a different calculation because we do not have a gross allocation to leaves per se, but use a net allocation:

LayerFoliageNPP=LayerFoliageAlloc +LayerPartTurnover

TotalNPP= ∑ LayerPartNPP

Ra or Autotrophic respiration

Ra consists of two parts. The first is maintenance respiration MaintenanceRa which is for all plants parts accounted for by LayerPartResp. An exception is for foliage which is calculated as

LayerFoliageResp= *Layer*FoliageRespRate * LayerFoliage

To calculate MaintenanceRa one sums all the respiration for the different layers and parts

MaintenanceRa=∑ LayerPartResp

The second part of Ra involved construction-related respiration, which accounts for the fact that making different forms of organic matter from sugars has a metabolic cost. It is assumed that an average respiration cost of construction is 25% of NPP

ConstructionRa=0.25*TotalNPP

Ra is then the sum of these two forms of respiration

TotalRa= MaintenanceRa + ConstructionRa

M or Mortality

The mass of all plant parts dying is defined as mortality (M). For most of the time mortality is not associated with disturbances. Disturbance related mortality is not included in this output.

TotalM= ∑ *LayerPart*Mort + ∑ *LayerPart*Turnover + ∑ *LayerPart*Prune

Where *LayerPart*Mort accounts for losses associated with tree death, *LayerPart*Turnover accounts for losses associated with death of non-woody parts (i.e., foliage and fine roots), and *LayerPart*Prune accounted for death associated with non-stem woody parts (i.e., branches and coarse roots).

Rh or Heterotrophic respiration

This is the sum of all decay losses for the dead and stable pools and that associated with heart-rot decomposition.

TotalRh= ∑ *Pool*DecayLoss + ∑ Stable*Pool*Decay +LayerHeartrotResp

GPP or Gross Primary Production

Gross Primary Production is essentially net photosynthesis. It is calculated as

TotalGPP= TotalNPP + TotalRa

NEP or Net Ecosystem Production

Net Ecosystem Production is the balance between production and respiration, in this case the difference of NPP and Rh:

TotalNEP= TotalNPP- TotalRh

Note that NEP does not account for losses other than RH, so if there is harvesting or combustion related carbon removals NEP will not equal the net carbon balance of a stand (NECB) (Figure 19-2).

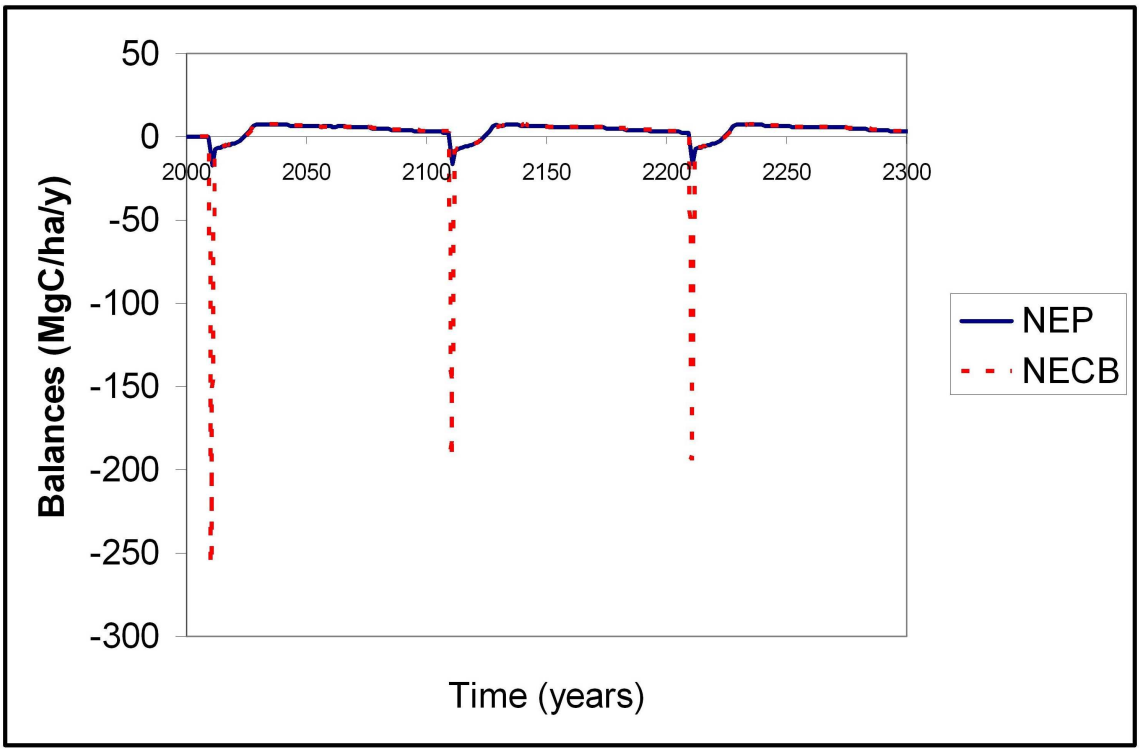

Figure 19-2. NEP and NECB are not the same when there are carbon losses other than respiration. In this example the forest is being harvested, and this leads to a large loss of carbon not captured by NEP alone.