User Guide for the Mondrian Model

William S. Currie1, Kenneth Elgersma2, Jason P. Martina3, and Sean J. Sharp1

Mondrian Version 4.3

User Guide last updated 9/9/19

1. School for Environment and Sustainability (SEAS), University of Michigan. Email: [wcurrie@umich.edu](mailto:wcurrie@umich.edu), [seanjsharp@gmail.com](mailto:seanjsharp@gmail.com)
2. Department of Biology, University of Northern Iowa. Email: [kenneth.elgersma@uni.edu](mailto:kenneth.elgersma@uni.edu)
3. Department of Biology, Texas State University. Email: [jason.marti993@gmail.com](mailto:jason.marti993@gmail.com)

# Table of Contents

[Table of Contents 2](#_Toc515045244)

[I. Introduction and Overview 6](#_Toc515045245)

[Model history 6](#_Toc515045246)

[Purpose of this User Guide 7](#_Toc515045247)

[Basic structure of the Mondrian model 8](#_Toc515045248)

[II. Model structure 9](#_Toc515045249)

[Spatial framework of grid cells and plant locations 9](#_Toc515045250)

[Vertical model structure and conventions 10](#_Toc515045251)

[Vertical dimensions of detrital pools 11](#_Toc515045252)

[Water level 12](#_Toc515045253)

[Living plants: vertical dimentions 12](#_Toc515045254)

[Plant individuals, tissues, and nutrients 12](#_Toc515045255)

[Conception of individuals and tissues 12](#_Toc515045256)

[Nutrients in living tissues 13](#_Toc515045257)

[Detritus and inorganic nutrient pools 14](#_Toc515045258)

[Standing dead litter 14](#_Toc515045259)

[Aboveground litter 14](#_Toc515045260)

[Belowground litter 14](#_Toc515045261)

[Muck 15](#_Toc515045262)

[Mineral soil / sediment organic matter 15](#_Toc515045263)

[Inorganic N pools 15](#_Toc515045264)

[III. How Ecological Processes are Modeled in Mondrian 17](#_Toc515045265)

[Beginning with bare sediment and achieving a steady-state community 17](#_Toc515045266)

[Arrival and establishment of plant propagules 18](#_Toc515045267)

[1. Initialization of plant populations at the start of a model run 18](#_Toc515045268)

[2. Introduction of invasives into an established plant community 19](#_Toc515045269)

[3. Background seeding in or propagule pressure 19](#_Toc515045270)

[Production of a new stem at the location of a rhizome node 20](#_Toc515045271)

[Litter thickness affecting C and N thresholds needed for establishment of new stem 20](#_Toc515045272)

[Acquisition of C and N for new stem establishment 21](#_Toc515045273)

[Light competition 22](#_Toc515045274)

[Plant nutrient uptake, nutrient competition, and growth 23](#_Toc515045275)

[Belowground competition and growth of the rhizome node 24](#_Toc515045276)

[Allocation of N to aboveground stems and C gain 24](#_Toc515045277)

[Plant reproduction 25](#_Toc515045278)

[Plant mortality 26](#_Toc515045279)

[Inability to make a stem 27](#_Toc515045280)

[Flooding mortality #1 27](#_Toc515045281)

[Flooding mortality #2 28](#_Toc515045282)

[Maximum longevity 28](#_Toc515045283)

[Litter production 28](#_Toc515045284)

[Senescence 28](#_Toc515045285)

[Belowground litter produced through mortality 29](#_Toc515045286)

[Decomposition and N dynamics in detritus 29](#_Toc515045287)

[Temperature 29](#_Toc515045288)

[Aerobic versus anaerobic decomposition rates 30](#_Toc515045289)

[Detrital mass losses and transfers 30](#_Toc515045290)

[N immobilization and mineralization 31](#_Toc515045291)

[Detrital export from the wetland 31](#_Toc515045292)

[N inflow, ecosystem N retention, and N export 32](#_Toc515045293)

[N inflow in precipitation 32](#_Toc515045294)

[N inflow from land 32](#_Toc515045295)

[N diffusion into grid cell pools and ecosystem N export 33](#_Toc515045296)

[Model structural options or ‘switches’ 34](#_Toc515045297)

[IV. Management treatments for control of invasive plants 36](#_Toc515045298)

[Management treatments in the vertical dimension 36](#_Toc515045299)

[V. Obtaining and installing Mondrian 36](#_Toc515045300)

[Software and obtaining Mondrian 36](#_Toc515045301)

[Fair use guidelines and citing Mondrian 37](#_Toc515045302)

[Installation and directory structure 37](#_Toc515045303)

[VI. Workflow for Setting up and Running Mondrian 37](#_Toc515045304)

[Designing a set of model runs 38](#_Toc515045305)

[Define the dynamic problem to be explored 38](#_Toc515045306)

[Decide on the number of model runs needed in the set 39](#_Toc515045307)

[Stochastic replicate runs 39](#_Toc515045308)

[Decide on the size of the space and related parameters 40](#_Toc515045309)

[Parameter estimation and gathering input data 40](#_Toc515045310)

[The *Parameter* input file 40](#_Toc515045311)

[The *Seed* input file 42](#_Toc515045312)

[The *Scenario* input file 42](#_Toc515045313)

[The *Batch* input file 43](#_Toc515045314)

[File setup template and data check 43](#_Toc515045315)

[Model testing, verification, and validation 43](#_Toc515045316)

[Executing a set of model runs 44](#_Toc515045317)

[Input and output file names and locations 44](#_Toc515045318)

[Strategy and conventions for a large set of model runs 44](#_Toc515045319)

[The main output file 46](#_Toc515045320)

[Additional output files 50](#_Toc515045321)

[Parameter values printout file 50](#_Toc515045322)

[Logical Flow Analysis output file 50](#_Toc515045323)

[Post-processing and analyzing results 52](#_Toc515045324)

[VII. Error checks and known glitches 53](#_Toc515045325)

[Automated error checking 53](#_Toc515045326)

[Logged errors with numbered error codes 53](#_Toc515045327)

[Mass balance checks 54](#_Toc515045328)

[Known glitches 54](#_Toc515045329)

[VIII. Best Practices for Archiving Runs and Continued Development 55](#_Toc515045330)

[Archiving input and output data from model runs 55](#_Toc515045331)

[Naming and saving: Exercises 55](#_Toc515045332)

[Naming and saving: Sets 55](#_Toc515045333)

[Set Hierarchy of files that should be archived 56](#_Toc515045334)

[Additional files not necessarily part of the Set hierarchy 56](#_Toc515045335)

[Who is responsible? 57](#_Toc515045336)

[Informal cheat-sheet to track multiple Exercises and Sets over time 57](#_Toc515045337)

[Model versions, development, and testing 57](#_Toc515045338)

[IX. Potential Research Questions to Explore 58](#_Toc515045339)

[X. Acknowledgments 61](#_Toc515045340)

[XI. References Cited 62](#_Toc515045341)

# I. Introduction and Overview

## Model history

Mondrian is a computer model of wetland community and ecosystem ecology. The name is an acronym for *Modes Of Nonlinear Dynamics, Resource Interactions, And Nutrient cycling*. It is a complex model that was initially developed for ecological research that sought to synthesize and better understand the interactions among plant populations, plant communities, and ecosystem processes including nutrient cycling (Currie et al. 2014). Since its initial publication, it has been used for a variety of purposes, each time expanding the model to include more processes and more complex integration across ecological levels of organization. It was expanded to include light competition and the effects of varying water levels and hydroperiod and used to study the integrated effects of water level and nutrient inflows on the joint outcomes of wetland C budgets and the success or failure of plant invasions (Martina et al. 2016). It has been used to better understand the ecology of clonal wetland plants and plant competition more broadly (Elgersma et al. 2015, Goldberg et al. in press). The model was also expanded to simulate the effects of wetland management and restoration practices to fight invasive plants in coastal wetlands (Elgersma et al. 2017). Recently, the model has been expanded to include more detailed, belowground, clonal branching architecture and more components of the wetland N cycle including nitrification and denitrification (not yet published). To date, the model has been applied only in Great Lakes coastal wetlands. However, the model processes are general enough that it could be used to study inland wetlands and wetlands in other regions.

We designed Mondrian so that it could be used by different research groups to address a wide variety of research questions in different wetland systems. Hundreds of parameters that represent plant traits or environmental drivers can be changed by the user by making simple changes to parameters that are contained in input files. At the same time, because site-measured values for such a wide array of parameters are often not available, default values are supplied for each parameter. Many of the parameters that represent plant traits are species-specific. For example, traits like maximum relative growth rate (***relGrate***), tissue target C:N ratios, and dozens of others can be given different values for each species, if species-specific values are known or needed for a particular research question*. This highights the importance of identifying a specific research question before the model is used*; depending on the research question, it may or may not be important to use different parameter values for different species. Expressing the way that certain traits differ among species may add more realism, but the increased realism also adds more complexity and may make results more difficult to interpret. Sometimes simpler assumptions and approaches can offer better clarity to address a research question (Miller and Page 2007). Mondrian was designed to allow exploration of a range of questions from theoretical on one end of the spectrum to highly realistic on the other end of the spectrum.

Depending on the research question, the model user can turn various ecological processes in Mondrian “on” or “off” through switches. For example, for a particular model run, the user can turn light competition on or off, or turn the spatially explicit distribution of litter on or off. We see this as one of our most interesting innovations with Mondrian. This provides a powerful set of tools for exploring wetland ecological processes. We have found the best way to use the model is, to address specific research questions, design large sets of model runs (hundreds to thousands) where most parameters and drivers are held constant but a few are varied in a factorial design.

As of 2019, future plans for continued model development include adding P cycling and adding a means for simulating the propagation of invasive plants across a larger, more heterogeneous area. We are also in the midst of developing a simplified meta-model that captures key aspects of Mondrian functionality. The meta-model will be incorporated into the LHM (Landscape Hydrology Model) of hydrogeochemical processes in large river watersheds that our collaborators are developing at Michigan State University.

## Purpose of this User Guide

**Important Note: This is a living document.** While it is fairly comprehensive in describing how Mondrian works and how to use it, we, the development team are continually updating this guide. Areas where we plan future expansion of this guide may be highlighted in blue.

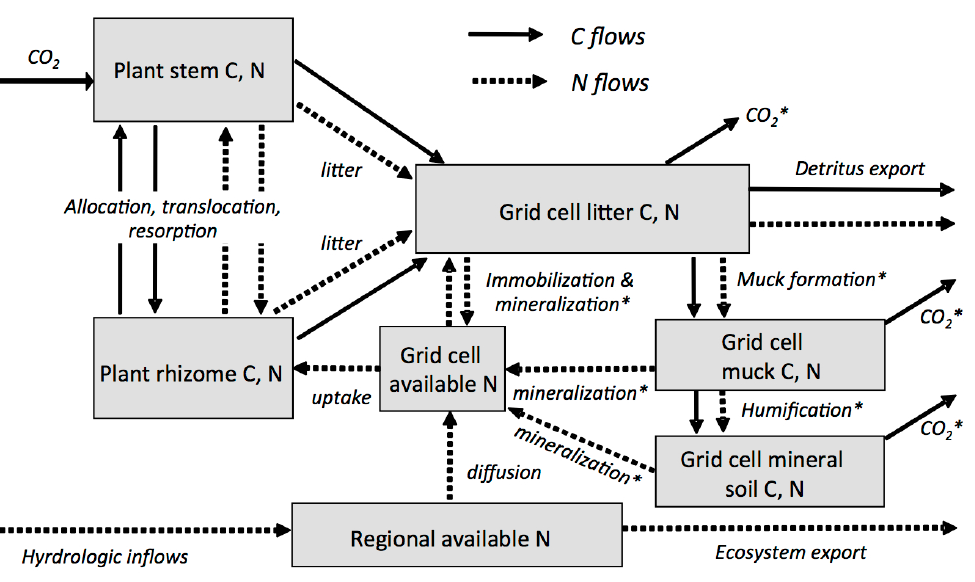
This user guide serves a few purposes. First, it describes and documents the detailed processes in the model to serve as a reference for collaborators currently using Mondrian, including students. Another goal is to facilitate broader use of Mondrian by other research groups who may wish to work in a variety of wetland systems on a range of basic or applied research questions. This guide describes the workflow that we use to estimate model parameters, create the needed input files, and interpret the model output.

We also see Mondrian as being a potentially valuable teaching tool for educating graduate or undergraduate students about a variety of topics: wetland community processes, invasive plants, ecosystem processes such as N cycling and C storage, how water level mediates ecosystem processes in wetlands, management and restoration of wetlands, and modeling concepts and methods. Over time we have made the model more user-friendly and straightforward to simply install and run on any computer and get interesting results. We are also working on lesson plans for teaching labs that would use Mondrian, although at present these are not ready for distibution. If you are interested in using this model for either research or teaching, see the section on ‘Obtaining Mondrian’ – we encourage you to do so, and we would like to hear from you.

Mondrian papers in the literature provide some details on the ecological processes in the model. However, the model has changed over time, so each paper in the literature contains a description of the model at a particular point in time and each later paper supersedes the earlier papers. Published papers also contain a limited amount of detail. This user guide contains much greater detail.

**Model version number**: After each set of changes and model augmentations, Mondrian is released with a new version number, or a dated update without a change in version number (if the changes were minor and the same input files can be used). Our vision is that this user guide will be kept current, so that each time there is a model change it is described in the gui de. However, the user guide sometimes gets ahead of the latest public release, for example if the user guide describes a new process that is in a beta-testing version that is being tested by the development team but has not yet been released. In such a case, the majority of the user guide will still be accurate. We will include notes along the lines of “If you have version 4.0 or earlier, process such-and-such is not included.”

## Basic structure of the Mondrian model



**Fig. II-1.** Pools and fluxes in the Mondrian model. Solid lines represent C fluxes, while dashed lines represent N fluxes. Martina et al. (2016).

Mondrian uses water level, nutrient inflows, and the arrival of plant propagules to simulate local community-ecosystem processes in wetlands. It is a spatially explicit, individual-based model that simulates clonal growth, plant competition for N and light, and population-community interactions. It is capable of simulating each individual in a dense herbaceous wetland community, approximately 5,000 plants/m2. It includes ecosystem-level N cycling in plant uptake, translocation within clones, above and below ground litter production, N immobilization in litter, and N mineralization from litter, muck, and sediment organic matter. C and N pools are closely linked through C:N stoichiometry; C pools include above and belowground plant tissues, litter, muck, and sediment organic matter. First-order kinetics govern decomposition rates (Currie et al. 2010), which slow in each detrital pool separately based on muck and litter thickness relative to water level.

A key strength of Mondrian is that ecological processes in the model span four levels of ecological organization:

Individual plant level: Individual plant physiology, including movements of C and N between aboveground and belowground tissues and among individual ramets connected within a clone, drives clonal growth and reproduction.

Population level: Reproduction and mortality in individual ramets and clones drives population growth or decline.

Community level: Competition for N uptake and for light (including shading) among individual plants of different species drives changes in plant community composition over time. This produces competitive hierarchies among native plants in the model as well as the success or failure of invasive plants that attempt to invade native plant communities in the model.

Ecosystem level: Mondrian includes sophisticated ecosystem-level N cycling and C cycling, including N inflows, wetland N dynamics, N retention, and N export from a wetland, as well as a balanced C cycle that allows either dynamic buildup or loss of C in detrital pools over time.

Because Mondrian is a process-based model, processes operating at one level of organization affect and interact with processes acting at all of the other levels of organization in a variety of realistic ways, allowing us to model a wetland ecosystem as a complex adaptive system. Because it is an individual-based model, it allows population-community processes to be simulated and studied as emerging from ecological interactions among individuals (Grimm et al. 1999). As an example of how levels of organization enter into ecosystem biogeochemistry, modeled N availability for plant uptake in any particular spatial location is determined through complex interactions among N inflow, ecosystem-level losses of N driven by hydrologic flushing and denitrification, and N immobilization and mineralization from litter, muck, and sediment organic matter. Ecosystem N retention, driven partly by NPP (net primary productivity) and litter production, works together with N inflow to control N availability to individual plants over the long term, while at the same time elevated N availability drives greater NPP and litter production (Currie et al. 2014). Because nutrient cycling takes place in separate, spatially explicit grid cells, Mondrian also simulates spatial heterogeneity in nutrient cycling and its effects on plant growth and competition (Huston et al. 1988).

# II. Model structure

This section provides greater detail on the model structure, while the following major sections provides greater detail on the modeling of ecological proceses.

## Spatial framework of grid cells and plant locations

Mondrian treats horizontal space as a square grid of cells. The number of grid cells is a user parameter ***ncells*** in the Parameter input file. The value of ***ncells*** must be a square of an integer, e.g. 36 (for a grid of 6 x 6 cells), 49 (for a grid of 7 x 7 cells), etc. (The maximum number currently allowed is 625, i.e. for a grid of 25 x 25 cells). The size of each grid cell is also specified by a user parameter ***s*** (for ‘side’ of a single cell) in the Parameter input file. The size of a grid cell controls the scale of plant competition for nutrients and light because the competitive interactions take place among all individuals within a cell.

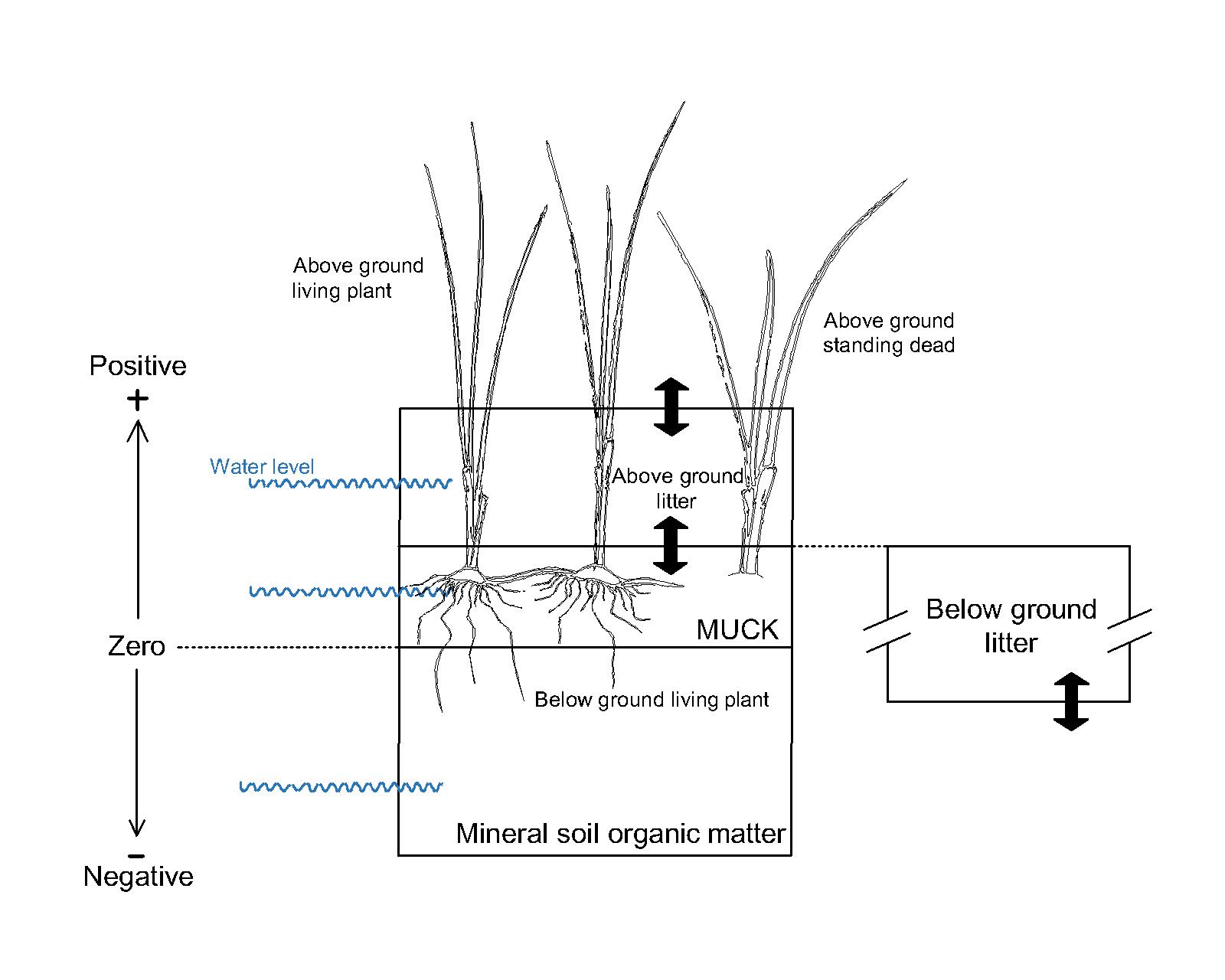
By setting the number of grid cells and the size of each grid cell, the user also sets the total size of the space. For example, in the first Mondrian paper (Currie et al. 2014) we used 49 grid cells (7 x 7) with a cell size 0.075 m on a side, producing a square area of 0.525 m x 0.525 m.

Each plant in Mondrian has a spatial location stored as a pair of real numbers (decimals) as (x, y) coordinates. Plant locations are always established within the total modeled area and are independent of numbers and sizes of grid cells.

**Example.** A user could explore the effects of using grid cells of 0.1 or 0.2 m on a side while keeping the total area constant, by comparing a 10 x 10 grid of cell size 0.1 m versus a 5 x 5 grid of cell size 0.2 m. Both approaches use a total modeled area of 1 m x 1 m. In this example, if plants were “seeded in” the same way in both cases and all plant traits were identical in both cases, then one could study the ecological effects of assuming that the scale of competition for light and nutrients was 0.1 m versus 0.2 m. A research question could focus on the effects of these alternative scales of plant competition on population, community, and ecosystem processes.

In Mondrian, as a clonal plant individual reproduces vegetatively, it produces a daughter rhizome node at a certain distance and angle from the parent. Depending on the size of the grid cell and the distance from the parent to the daughter (both of which are user-set parameters), the daughter rhizome node may be created in either the same grid cell or a different grid cell. If the daughter is in an adjacent grid cell to the parent, it experiences a competitive environment that is different from that of the parent.

If vegetative reproduction takes the location of a daughter rhizome node outside of the modeled area, its location is wrapped to the grid cell on the opposite side of the space. In other words each edge of the space wraps onto its opposite edge. Modelers sometimes refer to this as “wrap the map”, or refer to the topology of the modeled space as a torus.



**Fig. II-2.** Definitions and conventions used for the vertical dimension in Mondrian. AG = above ground; AGL = Above ground Litter; BG = below ground; BGL = below ground litter; MSOM = mineral soil organic matter. Water level (depth) varies on a daily basis, expressed as a positive or negative number relative to zero: Zero in the vertical dimension is always defined as the top of the MSOM pool and bottom of the MUCK pool. Bold double-ended vertical arrows indicate that the height of the top or bottom of the indicated pool can vary over time and from one grid cell to another. See text for explanation. (Figure: Ye Yuan and William Currie)

## Vertical model structure and conventions

Mondrian has a vertical dimension in which model pools such as muck and litter have defined vertical upper and lower bounds, plant individuals have defined heights, and water level (or depth) varies vertically on a daily basis (Fig. II-2). The vertical dimension is tracked in meters, with zero always defined as the top of the MSOM (mineral soil organic matter) pool. Water level (or “depth”; we use the terms interchangeably) can be positive or negative on this vertical axis. The water depth determines which pools are aerobic or anaerobic. This affects biogeochemical processes and also determines the depth at which individual plants are submerged. Individual plant heights are used several ways: in light competition, in determining flooding mortality, and in comparing plant height against depth of the litter layer. Some simulated wetland management activities such as mowing and herbicide application also affect individual plants different based on their height (*See* Management Activities section).

### Vertical dimensions of detrital pools

The MSOM pool has a fixed vertical dimension: It always extends from zero to – 0.3 m (negative 0.3 m) depth. The MUCK pool sites above the MSOM pool and its thickness can vary over time and from one grid cell to another. The above ground litter (AGL) pool sits atop the MUCK pool and similarly has thickness that varies. The below ground litter (BGL) pool is somewhat more complicated; its upper boundary is even with the bottom of the AGL pool (top of the MUCK pool), while its lower boundary varies based on the amount of litter in the pool at any point in time. The BGL pool is viewed as being physically intermingled with the MUCK pool, although the two have distinct biogeochemical functions in the model.

For the MUCK, AGL, and BGL pools, the vertical thickness of each pool varies based on its mass and its bulk density. (As indicated above, the vertical dimension of the MSOM pool is fixed at 0.3 m). Mondrian tracks the organic C content (g C / m2) of each of these pools as dynamic variables based on ecological and biogeochemical processes. Each simulated day, accounting for the changes in organic C contents of these pools, Mondrian re-calculates the lower and upper vertical dimensions of each detrital pool. The organic C content of each pool (g C / m2), together with the C concentration per unit mass (parameters specified by the user) and the bulk density of each pool (dry mass per volume, parameters specified by the user) together determine the vertical thickness of each pool. Mondrian tracks the upper and lower vertical bounds of each pool on a daily basis. (Note that since the MSOM pool has a fixed thickness of 0.3 m, and since its organic C content varies dynamically, and since its overall bulk density [organic + mineral] is set by the user, its organic C concentration [g C / cm3] varies dynamically.)

### Water level

Water level can vary on a daily basis. Note that zero water level (or depth) is defined as even with the top of the MSOM pool (Fig. II-2); higher water levels are positive numbers while lower water levels are negative numbers. There are no model processes that affect the water level in Mondrian; it is simply read in on a daily basis from the *Scenario input file*. The user can construct hypothetical scenarios of seasonal water level fluctuations or annual or decadal time-scale changes in water level or use data from a hydrograph for a particular field site. The ability to vary the water level on a daily basis allows the user to explore a large number of effects such as anaerobic versus aerobic decomposition in detrital pools or differential flooding mortality by plant species; these effects of water level are described elsewhere in this user guide.

### Living plants: vertical dimentions

Living plants have tissues aboveground (herbaceous stems) and belowground (rhizome nodes, which conceptually include roots). In Mondrian, the bases of plant stems are located at the top of the MUCK pool of detritus (Fig. II-2). In model runs in which the MUCK accretes and grows thicker over time, the bases of the plants rise with the top of the MUCK pool as it rises. Plant belowground tissues are assumed to be physically located in the MUCK and MSOM pools.

Note that when an aboveground stem dies it becomes standing dead litter; the vertical dimensions of this standing dead litter are thus the same as the plant when it died (Fig. II-2).

For an explanation of how individual plant heights are affected by wetland management activities in the model, *See* Management activities section.

## Plant individuals, tissues, and nutrients

### Conception of individuals and tissues

Here we outline the definition of some terms used in the Mondrian model code and in this User Guide. (Note that these terms may be used differently in different papers in the literature.)

*Ramet*: We use this term to refer to an individual plant, including an aboveground part (shoot, or stem) together with a belowground part (rhizome, including roots). In clonal plants, ramets can be connected belowground. (In the model code and variable names, ramet is often abbreviated *ram*, or *rams*.)

*Rhizome node*: We use this term to refer to a belowground entity that (a) can grow to produce both belowground tissue and aboveground tissue (a stem), and (b) can connect physiologically to other rhizome nodes in the genet. In Mondrian a living rhizome node is considered a living individual whether or not it has an aboveground stem. A rhizome node may only have one living aboveground stem at a time. If the rhizome node lives through the end of the growing season and the winter, it may produce another aboveground stem in the next growing season.

When an individual (ramet) reproduces, the first part of a new individual to be produced is the new rhizome node. Each rhizome node may produce only one stem. At present, each rhizome node may produce only one daughter rhizome node (the model contains no rhizome branching).

*Genet*: In the model, individal ramets that are clonally connected compose a *genet*. A genet may comprise only one individual ramet, or it may contain many. Each genet is ultimately limited in size by the rates at which individuals reproduce and experience mortality.

*Parent* and *daughter*: When one individual plant reproduces clonally to produce another individual connected (and immediately adjacent to it) in a clone, we refer to the one that reproduced as the *parent* and the offspring individual as a *daughter*.

*Aboveground tissue*, or *stem*: We use the terms *aboveground tissue* and *stem* somewhat interchangeably to refer to the same thing. It combines structural and foliar tissue into a single, aggregated unit. A *stem* refers to an individual and *stem density* refers to the number of individual stems within a unit area. *Aboveground tissue* can refer to an individual, or all of the aboveground tissue aggregated within a grid cell, or within a species, or across all of the species in the entire space. At present, all plants in Mondrian have herbaceous, annual aboveground tissues, meaning all aboveground tissues senesce and become standing litter every year at the end of the growing season.

*Belowground tissue*: This refers to the physical rhizome, roots, and node (as a botanical plant part) together. It can can refer to an individual, or all of the belowground tissue aggregated within a grid cell, or within a species, or across all of the species in the entire space. At present, all plants in Mondrian have belowground tissues that are perennial and that are able to survive through the winter, and able to potentially produce a new stem the following growing season (*See* Ecological Processes).

*Maximum size*: This is an important concept in Mondrian. The size of each individual at any point in time is important because it determines N uptake demand, competition for light (shading others and being shaded by others), response to management treatments, and other model processes. Because individual plants in the model start very small and grow, their sizes vary over the growing season and vary from one individual to another. The maximum size is a species parameter that describes the maximum allowable size (mass) that an individual of that species could potentially attain. The value of this parameter has important and complex effects for a species in the model. It is related to plant sizes measured in the field; at the same time, if plants are in competition in the model, they seldom reach this maximum size. This suggests that field-measured actual sizes for plants in competition do not provide the right value for this parameter in the model. The value of this parameter needs to be set slightly larger if we want the model to produce individual sizes that reflect those in the field, when plants are experiencing competition from neighbors.

### Nutrients in living tissues

Aboveground and belowground tissues each have a target C:N ratio. These are set by the user in the Parameter input file. These are species-specific parameters, meaning the user may set different target C:N ratios for each plant species.

These function as “target” C:N ratios because model processes shift C and N among tissues to try to achieve these target ratios in growth and clonal reproduction, but because tissues are connected together in complex ways it is not always possible to achieve these C:N ratios exactly in all tissues in every time step. As a result, the actual C:N ratios in tissues of individual plants may ‘float’ somewhat around these targets.

The target C:N ratios are key in controlling competition for N and how plants respond to competition. (*See* Ecological Processes / plant growth).

## Detritus and inorganic nutrient pools

Detritus and available nutrients (available for plant uptake) are stored separately in each grid cell. This produces spatial heterogeneity in detritus and nutrient pools and in nutrient cycling.

### Standing dead litter

In an individual plant, when its living aboveground tissue (stem) senesces, it becomes standing dead litter. Over time this shreds and falls as aboveground litter (*See* Ecological Processes / Litter production). In the detailed model processes, a stem of standing dead litter is a dead stem that remains attached, at its base, to its rhizome node. It prevents a new live stem from being produced by that rhizome until the standing dead litter shreds and falls to the aboveground litter pool.

For bookkeeping purposes, pools of standing dead litter C and N are combined across species and tracked separately within each grid cell.

### Aboveground litter

Aboveground litter refers to litter that conceptually is “downed”, meaning it has shredded from the standing dead litter, is no longer attached to the rhizome, and has “fallen.” Conceptually, it sits atop the muck layer in the model. In the wetlands that we simulate in reality, the aboveground litter can still be suspended somewhat in the air by other stems, forming semi-dense layers held above the ground or water. It can also form floating mats. We call it ‘aboveground litter’ simply to distinguish it from the standing dead stems (still attached) and from belowground litter produced by dead rhizomes and roots.

Pools of aboveground litter C and N are combined across species and tracked separately within each grid cell.

Mondrian tracks not only the mass of this litter layer but also its thickness (based on bulk density, a user-set parameter). Based on the thickness of this layer in the vertical dimension, and based on the water level which can fluctuate daily, the model determines what proportion of the aboveground litter layer is inundated on a daily basis. This affects its decomposition rate; *see* Ecological processes / decomposition.

### Belowground litter

Litter produced by the mortality of belowground tissues, rhizomes and roots. Pools of belowground litter C and N are combined across species and tracked separately within each grid cell.

### Muck

Muck is a detrital pool that conceptually includes partially but incompletely decomposed litter (*See* Ecological Processes / Decomposition). Muck is an organic layer that exists in a bounded, single layer that sits directly atop the mineral soil or sediment. Pools of Muck C and N are combined across species and tracked separately within each grid cell. Mondrian tracks not only the mass of Muck but also its vertical thickness (based on bulk density, a user-set parameter). This allows the model to determine what proportion of the Muck layer is inundated on a daily basis, based on the water level. The degree of inundation determines its decomposition rate.

If production of Muck outpaces its decomposition, the Muck layer can grow in thickness over time. This represents accretion of organic Muck in wetlands. Alternatively, if decomposition outpaces the production of Muck, the Muck layer will decrease in thickness over time. This repesents loss of organic soil through decomposition, or subsidence in wetlands. (There is no erosion of Muck in Mondrian). In Mondrian, as the level of the top of the Muck layer rises or falls, the bases of the plants rise and fall with it. This can produce changes in the heights of plants relative to the water level.

### Mineral soil / sediment organic matter

We use the term MSOM, which can be interpreted either as mineral soil organic matter or mineral sediment organic matter. This pool represents a well-decomposed, relatively recalcitrant pool of organic matter that contains humic substances and that occurs mixed within a primarily mineral soil (sediment). Pools of MSOM C and N are combined across species and tracked separately within each grid cell. Unlike the other detrital pools, the vertical dimensions or thickness of this layer do not vary. The vertical dimensions of the top and bottom of this layer are fixed.

### Inorganic N pools

Plant-available nitrogen in Mondrian is separated into pools of nitrate-N (NO3-N) and ammonium-N (NH4-N)[[1]](#footnote-1). These are stored separately in each grid cell, which we refer to as **grid cell available NH4** and **grid cell available NO3**. N cycling takes place separately in each grid cell (*See* Ecological Processes for more explanation). This allows spatial heterogeneity, in which N may be more available in one spatial grid cell but less available in another spatial grid cell, depending on the masses and dynamics of detrital pools in each grid cell on the individual plants and their uptake demands in each grid cell (Huston et al. 1988). The idea of having nutrients that can cycle and be depleted differentially in grid cells is similar to the 'local depletion zone' pools from Loreau (1998). However, the Loreau (1998) model is an analytical mathematical model that would not be able to include the complexity present in Mondrian. Important differences in Mondrian are that plants of either the same species or of different species can compete within the same local depletion zone, and that the grid cells in Mondrian not only differ from one another, but are also spatially explicit, meaning each grid cell has an explicit spatial location in the (x, y) grid.

In addition to the pools of available NH4-N and NO3-N in grid cells, Mondrian contains a **regional NH4-N pool** and a **regional NO3-N pool**, which again borrow conceptually from Loreau (1998). Conceptually, these regional N pools spans the entire modeled space. The regional NH4-N pool and can exchange NH4-N with each of the grid cell pools of NH4-N. Similarly, the regional NO3-N pool can exchange NO3-N with the grid cell pools of NO3-N. The N in the regional NH4-N and regional NO3-N pools is not plant-available; the N must move into a grid cell pool to be plant-available (*See* Ecological Processes). Conceptually, the regional NH4-N and regional NO3-N pools can be thought of as existing in the water that moves horizontally through the wetland, whether it is standing water or whether the water level is below the surface of the Muck layer.

# III. How Ecological Processes are Modeled in Mondrian

## Beginning with bare sediment and achieving a steady-state community

Normally we want to simulate a wetland that has a steady-state in plant densities and NPP of multiple plant species, so that we can then see what happens dynamically in response to a change such as arrival of an invader, or some other change driven by our research question. However, Mondrian always begins at the start of year 1 with bare sediment, i.e. with no living plants. At the start of each run, the model simulates the arrival of propagules that become established and reproduce clonally, filling the space. The user should complete test runs to see how many years it takes for the steady-state in plant densities and NPP to arise with the traits of their particular plant species and under their particular environmental conditions (grid cell initial pools of C and N, N inflows, hydroperiod, and so on). We have found that a near steady-state can be produced in about 15 years under the right conditions.

Mondrian contains numerous feedbacks across levels of organization; it simulates the wetland community-ecosystem as a complex adaptive system. At the start of a run, as plant propagules arrive they reproduce, shed litter, take up N and begin to drive the ecosystem N cycle, which in turn affects plant growth and competition. As the plant canopy grows more dense, plants shade one another and compete for light, which feeds back to control NPP, litter production, and alters N cycling further. The dynamic model essentially searches for an attractor in the hypervolume of variables that include N inflows, plant N uptake, hydrologic flushing of N from the wetland, litter production and decomposition, muck production and decomposition, plant heights and shading, plant reproduction rates, and others. A true steady-state is never reached because there will always be some interannual fluctuation in plant densities, NPP, and N cycling; it is better to think of it as an attractor state. We use the term ‘steady-state’ because it is a more familiar term.

Many dynamic ecosystem models work this way, in which an initial spin-up period is needed to establish a steady state in key variables of interest. Using Mondrian, it is important to recognize that a true steady-state of “constant” C and N cycling may be difficult to achieve because detrital pools turn over on a range of time periods, acting as capacitors or reservoirs that buffer change. Once a steady-state in NPP becomes established, the aboveground litter pool should also reach steady-state after another 5 to 10 years. However, the dynamics of the Muck and Mineral soil organic matter (MSOM) pools change over much longer time periods. If the initial pools of C and N in Muck and MSOM reflect oligotrophic conditions, but the N inflow that the user chooses is high, then these pools will become sinks for C and N for many decades or centuries. Alternatively, if the initial pools of C and N in Muck and MSOM reflect high-NPP conditions, but the N inflow that the user chooses is low, then these pools will slowly decline, releasing C and N over decades or centuries. To complicate the matter further, the turnover rates of these pools are strongly affected by the hydroperiod, the frequency and amplitude of flooding and inundation.

The user sets the initial sizes of the Muck, and MSOM pools of C and N in the Parameter input file. In setting the sizes of these pools, consider your research question and consider the type of near steady-state you wish to achieve as the plant propagules arrive and begin to fill the space. If you are unsure of the best values to use to initialize the Muck and MSOM pools, run the model under the environmental conditions you wish to simulate and see if these pools are shrinking or growing over time. The user can also change the turnover rates of these pools, which are controlled by parameters in the Parameter input file.

## Arrival and establishment of plant propagules

Plant propagules arrive and become established in three ways. We describe this as “seeding in” new plants, but there is no seed production or germination in Mondrian; instead, propagules arrive they immediately become established as partially grown individuals.

The model user has control over how many individuals of each species arrive as propagules in each year. To learn more about how the user sets these options, *see* Workflow / The Seed input file.

### Initialization of plant populations at the start of a model run

Plants can be established in year 1 of a model run, while other plants can become established in subsequent years. The user decides how many new individuals to introduce for each species in each year. This provides a great deal of flexibility to allow some species to become established before others are introduced, to study founder effects, attempted invasions into an established plant community, or other ecological phenomena. It also allows the user to grow the population of a species over time by establishing multiple cohorts – new individuals added each year, the first few years of a model run – instead of all at once.

**Example.** Suppose the user wishes to introduce 20 individuals per species of species 1, 2, and 3, in two cohorts of 10 individuals each per species in years 1 and 6. In year 1, for species 1, 10 spatial locations are chosen at random throughout the entire model space. At each location, a single, partially grown individual of species 1 is established. Each individual introduced is the sole member of a unique genet (thus 10 unique genets are introduced). This is repeated for species 2 and 3, and repeated for all three species in year 6, in this example.

The initial size of each individual introduced is controlled by a user parameter but also has a stochastic element. The user-set parameter ***InitRamC*** (species-specific) in the Parameter input file sets the proportion of the species’ maximum size that should be used to initialize an individual when it is “seeded in” in this way. Mondrian uses ***InitRamC*** as the mean of a normal distribution with standard deviation 25% of that value, samples that distribution, and uses it to initialize each individual’s C pools (i.e. size). The individual’s N pools are then initialized using the size of the C pools in that individual and the user-set C:N ratios for each tissue type, which are also set in the Parameter input file.

In subsequent years, whenever new individuals of the same species are “seeded in”, each new individual will also be the sole member of a new genet.

We advise introducing new individuals at the start of a model run in multiple cohorts over a period of several years. This stops all individuals of a species from being synchronized in growth and mortality, which could potentially produce unrealistic oscillations in plant densities and NPP. We have found that seeding in about 180 individuals per m2 of model space per year, four times (beginning in year 1, with 4 cohorts and a 2-year interval – i.e. in years 1, 3, 5, and 7), allows a robust population to reach a steady state in ramet densities and NPP typically by year 15 of the model run. Note that the densities will usually grow larger than the initial number, as plants reproduce clonally and fill the space. The steady-state density and NPP reached by each species will depend on environmental factors such as N inflow and water level, as well as the other species that are present, due to competitive effects.

### Introduction of invasives into an established plant community

This is simply a variant on the initialization of plant populations described above, however it has a distinct ecological purpose and has been a major application of Mondrian for research. Thus we describe it separately for clarity.

To explore what happens when an invasive species tries to invade an established plant community, first allow the native species to become established, then introduce cohorts of smaller numbers of potentially invasive individuals (i.e. propagules). For example, if species 1, 2, and 3 represent native plants, the user could allow steady-state populations to become first established by “seeding in” a number of individuals in several cohorts at the start of the model run, as described above. Then introduce attempted invaders, i.e. individuals of a potentially invasive plant species, after the native populations have become established and have reached steady-state. Typically we find the native community to reach steady-state by year 15, so we introduce attempted invaders beginning in year 15. We typically introduce multiple cohorts of attempted invaders, for example two cohorts five years apart. We call these ‘attempted invaders’ because depending on their traits, the traits of the established native plants, and other environmental factors such as nutrient availability and water level, they may or may not invade successfully (Example below).

**Example.** One of the most interesting uses of Mondrian that we have found has been to create a hypothetical series of attempted invaders that are identical except for one trait (Currie et al. 2014 used maximum size) that takes on a set of discrete values over a possible range. We then explored whether each hypothetical, would-be invader could successfully invade an established native community over a range of levels of N inflow. With this approach, we found that species of larger maximum size, all else being equal, invaded more successfully where N inflow was high, and less successfully where N inflow was low (Currie et al. 2014, Goldberg et al. 2017). To show that this resulted from competition, we showed that large-sized invaders could successfully become established at low N inflows in bare sediment, but not in cases when established native communities had already become established (Goldberg et al. 2017).

### Background seeding in or propagule pressure

The third way that new individuals (those *not* arising from in-model reproduction) are “seeded in” is through a background rate of seeding-in. A background rate is new arrival and establishment of individuals (each representing a new genet) that occurs each year. This serves three purposes. First, since the model space is small and random mortality occurs, it may be necessary to have a small background rate of seeding-in to offset random mortality of genets in the model space. Second, conceptually we could think of it as new genets that attempt to move into the modeled space from the surrounding wetland.

Third, for invasive species, this background seeding-in rate can be thought of as background propagule pressure, the steady or irregular arrival of small numbers of propagules.

As with the other parameters related to seeding-in, the ***background seeding rate*** is controlled through a parameter in the *Seed input file*. This is a species-specific parameter. If this rate is:

* Zero: There is no background seeding-in for that species.
* A decimal less than 1: This is interpreted as a probability of having one individual seed in, in a given year. For example, if the value is 0.2, then a single individual will seed in every five years on average. A random number generator is used to execute this in the model. (Note that this can add a great deal of variability from one model run to the next.)
* 1 or greater: This is rounded to an integer, and that number of individuals is seeded in every year for that species.

Note that background seeding-in does not begin until after a species has first been introduced in a cohort. For example, if an attempted invader species (or any species) first arrives in a cohort in year 15, then background seeding-in for that species will not begin until year 16. This allows the user to choose a specific year when an attempted invader first begins to arrive.

## Production of a new stem at the location of a rhizome node

For definitions of terms such as rhizome node, stem, etc., *see* Model structure / Plant individuals.

A living rhizome node can produce an aboveground stem if it is able to acquire enough C and N (see below). If it is during the growing season and a rhizome node does not already have an aboveground stem, it will try to create one. If it fails, it can **re-try every 5th day** during the growing season.[[2]](#footnote-2)

To create a new stem, a living rhizome node must accumulate enough C and N above a threshold. The threshold is expressed as a proportion of its own, species-specific, maximum belowground mass. The value of this threshold is tracked for each individual plant in the individual-based variable ***PBGCNN***, short for ‘proportion of belowground C and N needed’. For example, if the species-specific maximum belowground C mass for an individual is 0.8 g C, and if ***PBGCNN*** for that individual is 0.2, then that individual rhizome node must accumulate (in translocations from its parents), 20% of 0.8 g C, or 0.16 g C before it can make a new stem.[[3]](#footnote-3) A similar threshold test applies to its accumulation of N. However, C can only be accumulated through translocations from parent nodes (ramets), but N can be accumulated in two ways: through translocations from parent nodes (ramets) and from belowground uptake from the available N pool in the grid cell.

When a new rhizome node is created, the value of ***PBGCNN*** for an individual is set as ***PBGCNNmin***, a species-specific parameter set by the user in the Parameter input file.

### Litter thickness affecting C and N thresholds needed for establishment of new stem

Mondrian has the ability for a massive litter layer to suppress the establishment of new stems, particularly new stems of small-sized species. We created this feature in order to explore whether large-sized, highly productive invasive plants could suppress small-sized natives in part by producing massive or thick litter layers. If this model structural option is switched on (*See* Model Structural Options), the variable that controls the individual-based threshold for production of a new stem, ***PBGCNN***, can be affected by the mass of the litter in its grid cell. For an individual rhizome node, the value of ***PBGCNN*** begins, at the start of a model run, at the value of ***PBGCNNmin*** for that species, then increases as the litter layer in the grid cell of that rhizome node grows more massive. Note that the mass of the aboveground litter layer is tracked separately in each grid cell.

As ***PBGCNN*** for an individual rhizome node increases, this requires the rhizome node to accumulate a greater amount of C and N before it can establish a new stem. This is meant conceptually to represent the increased investment in a stem, as a proportion of its maximum size, that a small-sized plant would have to make in pushing up through a more massive (and deeper) litter layer before it enters the sunlight and can photosynthesize for itself. Once it is able to get through the litter layer and access sunlight, it can grow on its own and potentially become a source of photosynthate for other rhizome nodes in its genet. This is a proxy, in part, for shading of small stems by a thick layer of litter. In Mondrian the penetration through the litter layer is all wrapped up into one ‘establishment’ step that is binary: the rhizome node can either establish a successful stem, or it can not. Note that even once the stem is established, it must still compete for light in a grid cell where it may be shaded by dense, taller individuals.

If this model structural option is switched off, then the mass of the litter layer has no effect on the ability of new stems to become established and the value of ***PBGCNN*** is always equal to ***PBGCNNmin*** for that species, set by the user in the Parameter input file.

This is a fairly complicated feature to parameterize in Mondrian. To allow the development team and model users to see how it is functioning, in the main output file, the average value of ***PBGCNN*** is printed for each species each year (see Workflow / Main output file). Unless the user spends some time testing this feature to fully understand it and to build confidence, we recommend keeping this switch turned off for parsimony and for ease of interpretation of model results.

### Acquisition of C and N for new stem establishment

Consider a connected set of rhizome nodes, whether branched or unbranched. For any individual node we can trace back its immediate parent, then that individual’s parent, and so on. As a convenience of language we sometimes refer to these as the 1st-back parent, 2nd-back parent, 3rd-back parent, and so on. Note that in a branching genet, a node can be the 1st-back parent for more than one node, or it can be the 1st-back parent for one node and simultaneously the 2nd- or 3rd-back parent for another node, and so on.

A living rhizome node acquires C and N from its 1st-back parent (immediate parent) when it is created. Once created, it can gain N via root uptake, subject to N availability and competition from other nodes (ramets) of the same and other species in its grid cell (*see* Plant nutrient uptake). If a rhizome node does not have enough C and/or N to create a stem, it can request C and/or N subsidies via translocation from parents in its genet, one by one from its 1st-back parent, 2nd-back parent, etc. The user-set parameter ***SubsidyPar***, set in the Parameter input file, determines how many parents back in the chain can be used to supply subsidies via translocation: this can range from 1 to 5 parents back in the chain. This is a species-specific parameter.

Translocations of C and N are recipient-driven. A node can only receive subsidies from, or through, rhizome nodes that are still alive. A parent rhizome node back in the chain can supply a C and/or N subsidy only if, while doing so, it is still able to retain minimum levels of C and N (which are set by user parameters in the Parameter input file). These minimum retention levels of C and N represent structural C and N that can not be translocated.

The question arises whether a particular parent rhizome node back in the chain, if it lacks an aboveground stem of its own, can either supply from its own internal pools, or allow the pass-through from others farther back in the chain, subsidies of C or N. The user can decide whether to allow this or not through a switch in the Parameter input file, ***PTNSN*** which stands for “Pass-through non-stem nodes.” If this switch is turned ON, then subsidies of C and N can come from, or pass through, a parent rhizome node back in the chain, even if that rhizome node does not have a stem of its own. If this switch is turned OFF, then translocations can not come from or pass through such a rhizome node.

Once a C and/or N subsidy has been requested and received via translocation, the rhizome node that made the request and received the subsidy tries again to create a stem. If it now has enough C and N to do so, it can do so immediately (in the same timestep, i.e. on the same simulated day). If it still does not have enough C and N, and there is another parent back in the chain that it is potentially able to get a subsidy from, it will request the subsidy from that next parent back in the chain. This continues until it either makes the stem successfully, or until all parents that are potentially able to provide a subsidy have done so. If the individual node that is attempting to make a stem fails to do so in that timestep (that day), the C and N that the node has acquired remain in the node until the next time it is able to attempt to produce a stem. In this manner, N uptake and C and N subsidies are cumulative; they can accumulate within rhizome node until the threshold is reached to establish an aboveground stem.

## Light competition

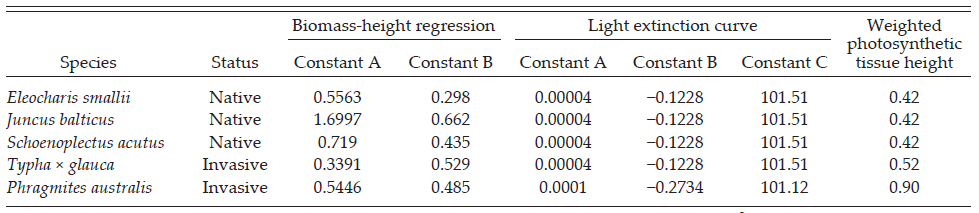
Plant growth takes place only during the growing season. The start and end dates of the growing season, ***GSstart*** and ***GSend***, respectively, are set by the user in the Parameter input file.

Each individual plant begins each day with a maximum possible relative growth rate, defined as the growth that could occur in one day, as a proportion of current plant mass, under optimal conditions. As in many individual-based ecological models, this maximum possible rate of growth is then scaled back based on the specific competition conditions experienced differentially by each individual on a daily basis. In Mondrian the maximum relative growth rate is a species-specific parameter ***relGrate***, set for each species in the Parameter input file. This applies to both aboveground and belowground tissues.

In Mondrian, light competition can be turned on or off for an entire model run (*see* Model Structural Options). This is a powerful tool for studying competition. Doing identical sets of model runs with light competition on in one run then off in another allows an investigator to explore the importance of light competition versus nutrient competition in a hypothetical manner that would be impossible to do in the field. Note that turning light competition off also has other effects: The plant canopy usually grows much more massive and NPP grows much higher with light competition off because there is no self-shading.

If light competition is switched on, then each individual plant, each day, has its relative growth rate scaled back based on the light extinction caused by the presence of other plants (both living and standing dead biomass) in its grid cell. Thus the size of the grid cell sets the scale of competition for light.

Light availability is calculated in 10-cm vertical segments separately in each spatially explicit grid cell[[4]](#footnote-4). The shading calculation uses species-specific light extinction curves applied to the plant biomass (stem + foliar dry weight) present on a daily basis, by species, in each vertical segment of each grid cell. Plant height is determined using species-specific biomass-height allometric equations obtained from our own field data (Table III-1). The effect of shading on each individual stem is simplified by the light environment at a fixed proportion of its height, a species-specific parameter that represents the typical vertical distribution of photosynthetic tissue for the species (J. Knops and H. Hager, unpublished data; Table III-1). Growth rate is then scaled back using a Michaelis–Menten equation of relative growth rate as a function of light availability based on species-specific data on photosynthetic rate as a function of irradiance (Knops and Hager, unpublished data).



**Table III-1.** Species-specific model parameters for biomass-height allometric equations and canopy architecture (Martina et al. 2016).

**Biomass-height allometric equation**, where height is in meters and biomass is in g dry mass:

height = A × biomassB (eqn. III-1)

where A and B are species-specific constants (Table III-1).

**Light extinction curve**: Canopy architecture is modeled using a polynomial-shading curve:

% full light = Ax2 + Bx + C (eqn. III-2)

Where x is plant biomass (dry weight) in a particular grid cell, calculated in 10-cm vertical segments, and A, B, C are species-specific light extinction parameters (Table III-1). In Martina et al. (2016), native species and *Typha x glauca* (invasive cattail) used the same equation for light extinction (with different height ranges) because of their similar leaf–stem architecture. *Phragmites australis* used a distinct light extinction equation because of its dramatically different leaf–stem architecture compared with the other parameterized species.

While light extinction curves are species-specific, the total light environment for an individual is calculated by summing the light extinction from all individuals of all species present in a grid cell.

**Weighted photosynthetic tissue height parameter**: Mondrian uses this parameter to model individual responses to shading; it is expressed as a proportion of the height of an individual. For example, for *Typha x glauca*, for a 1 m tall individual, the light environment at a height of 0.52 m (Table III-1) is used to approximate the shading effect on that individual, using eqn. III-2.

Note that all of the parameters discussed in this subsection are species-specific and set by the model user in the Parameter input file. This allows the user complete freedom to use different parameter values, or to add a new species, or to do a sensitivity analysis on these parameters, or to do simulated experiments where some parameters are kept constant across species while others are varied.

## Plant nutrient uptake, nutrient competition, and growth

Unless nitrogen (N) inflows and availability are extremely high, plant growth in Mondrian is N limited. Living rhizome nodes within each grid cell compete for the available NH4-N and NO3-N in that grid cell on a daily basis. The spatial size of the grid cell thus sets the scale of belowground competition for N (as it does aboveground for light).

Mondrian contains a submodel of N source-sink physiology within plants, making the belowground and aboveground plant tissues function somewhat independently but also translocating N upward to the stem and translocating C downward to the rhizome node.

Plant growth takes place in two stages. First, belowground rhizome nodes compete for NH4-N and NO3-N within each grid cell (described below). After all of the available inorganic N in the grid cell has been taken up by rhizome nodes, then for each rhizome node that has an existing aboveground stem, the stem requests and receives a translocation of N from its rhizome node, which then determines the amount of aboveground growth.

### Belowground competition and growth of the rhizome node

Plants within a spatial grid cell take up both NH4-N and NO3-N, with no preference for either one, in the ratio that NH4-N and NO3-N are available in the particular grid cell. For clarity, this section refers simply to competition for “N”.

Belowground competition for N arises mechanistically[[5]](#footnote-5) from the aggregate demand (*Dl*) summed over all rhizome nodes within a particular grid cell (*l*)

(eqn. III-3)

Where (*i*) is each species, (*ns*) is the number of species in the model run, (*j*) represents the individual rhizome node within a species, (*ni*) represents the number of individuals of species (*i*) in grid cell (*l*), *Cij* is the carbon mass (g C) of the individual, *ρi* is the target C:N mass ratio for new growth of belowground tissue in species (*i*), and *rij* is the maximum relative growth rate for the individual. (Note that light competition does not affect belowground competition for N; light competition enters later when aboveground stems seek translocations of N from their rhizome nodes).

After calculating the aggregate N demand summed over all living rhizome nodes within a grid cell (*Dl*, eqn. III-3), the ratio of inorganic N availability to aggregate N demand in that grid cell (*Nl* / *Dl*) is calculated on a daily basis. This ratio is then used to scale back the N uptake of each individual rhizome node in that grid cell for that day. The daily N uptake of each individual rhizome node (*Uij*) is then calculated as follows:

(eqn. III-4)

Where ‘*min*’ denotes minimum and *Nl* is available inorganic N (g) in the grid cell prior to any plant uptake during that daily time step. In other words, the ratio of available N to aggregate N demand by all rhizome nodes within a grid cell (when this ratio is <1) is used to scale back the belowground N uptake for each individual within the cell. This formulation allows competition to be highly dynamic in Mondrian as the numbers and sizes of neighbors change, for each of the thousands of simulated individuals, on a daily basis. When values of maximum relative growth rate and target C:N ratios in new tissue are equal across species, competition for N is directly proportional to size differences among individuals (perfectly size-symmetric; Schwinning and Weiner, 1998).

Once NH4-N and NO3-N are taken up into plant tissue, then inside the plant, NH4-N and NO3-N are no longer differentiated. They are combined conceptually into plant internal N, or organic N, which can exist in various tissues of each plant.

### Allocation of N to aboveground stems and C gain

This occurs after competition for belowground NH4-N and NO3-N uptake is complete and all rhizome nodes within a grid cell have taken up NH4-N and NO3-N for that day. Mondrian then loops through all living rhizome nodes, one by one. If a living rhizome node does not have an aboveground stem, it simply retains the N that it was able to take up, “banking” this N to help it attempt to establish an aboveground stem in the future. It may need to compete for N over several simulated days or weeks before it acquires enough N to create an aboveground stem. (Note that as the rhizome node accumulates N, this narrows the simulated C:N ratio in that rhizome node; this is an example of how tissue C:N ratios in individuals “float” as a result of the Mondrian submodel of internal source-sink plant physiology.)

If a living rhizome node does have an established, living aboveground stem, then that stem requests a translocation of N upward into the stem to allow it to grow. The amount of N requested is determined by the aboveground plant size (g C), the species-specific target C:N ratio for growth of aboveground tissue (the species-specific parameter ***InitAGCNrat*** set by the user in the Parameter input file), and the maximum relative growth rate for that individual stem ***as scaled back by shading from other plants in the grid cell*.** This is where light competition enters in to scale back aboveground growth. All of these factors determine the absolute amount of N (g N / day) requested by the stem to receive from its rhizome node.

Given this request for N from its aboveground stem, the individual rhizome node supplies as much of the N translocation request as possible, translocating N upward into the stem, but with two constraints: (1) if the translocation of N aboveground (divided by the target C:N ratio for aboveground growth) would cause the aboveground stem to be larger than the maximum size (g C) for that species, the translocation is scaled back by the appropriate amount so that the stem never grows above its species’ maximum size; (2) if the translocation of N aboveground would cause the rhizome node to retain too little N below a threshold, the N allocation to the stem is also scaled back by the appropriate amount. This minimum amount of N that needs to be retained by the rhizome node conceptually represents structural N that can not be translocated. (In the model code, this amount of structural N that must be retained in the rhizome node is fixed at 10% of the maximum size [g C] of the belowground rhizome node divided by its target C:N ratio.)

Once it has been determined how much N (g N / day) can be translocated from an individual rhizome node to its stem that day, this directly determines the growth of the stem. As a simplification, photosynthesis is not explicitly modeled in Mondrian. Carbon gain by the stem (g C / day) is simply calculated as the N translocated upward to the stem multiplied by the species-specific, aboveground target C:N ratio set by the user. Of course, it is assumed implicitly that this C is acquired from the atmosphere through photosynthesis.[[6]](#footnote-6)

In addition to the C gain for stem growth, another C gain also implicitly occurs and is allocated belowground. This is determined by a species-specific parameter set by the user in the Parameter input file: ***CallocBGprop***, defined as the proportion of photosynthate C allocated to the belowground rhizome node. This automatically occurs and is translocated from the stem to the rhizome node. It is important to note that this C allocation of photosynthate belowground is not determined directly by belowground N uptake; instead, it is determined by the amount of N that can be translocated to the aboveground stem, which includes the effects of light competition.

## Plant reproduction

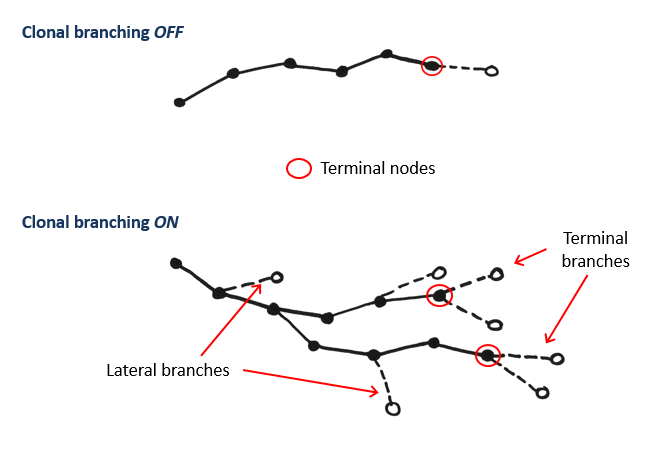
Plant reproduction involves a complex set of processes in Mondrian. At present, clonal or vegetative reproduction is the only means; there is no sexual reproduction, seed production, or seed germination in the model. Plants are “seeded in” as established plants, meant to represent propagules (*see* Arrival and establishment) and then plants compete with one another, withinin the model space, to reproduce clonally.

The first step for a parent ramet to produce a new individual daughter ramet is to produce the new daughter rhizome node. In Mondrian this takes place by having the would-be parent rhizome node attempt to create a daughter rhizome node. No individual rhizome node is given the opportunity to establish a daughter rhizome node until after it has established an aboveground stem in that growing season. (Thus, at the start of a growing season, the rhizome nodes that survived through the winter must first produce new aboveground stems.)

Rhizome nodes that have established aboveground stems then fall into two categories: those that have not yet reproduced (terminal nodes, Fig. III-1) and those that have already reproduced (non-terminal nodes). Every 8th day during the growing season, a subroutine is called to conduct tests to see whether any new daughter rhizome nodes can be created. This **8-day interval** is important because it sets a limit for the maximum rate of vegetative reproduction and number of new individuals that could be produced within a genet in a given growing season. To reproduce, a rhizome node has to pass tests to see whether it has accumulated enough C and N. If it fails the tests, 8 days pass before it is tested again. Under low nutrient conditions, vegetative reproduction is limited by nutrient availability. But under high nutrient conditions, this built-in 8-day interval becomes the limiting factor.

### Terminal nodes, non-terminal nodes, and branching

The user chooses whether clonal branching is OFF or ON by setting this option in the Parameter input file[[7]](#footnote-7). In either case, **terminal nodes** are always tested to see whether they have accumulated enough C and N to reproduce a single daughter rhizome node. When clonal branching is OFF, only the terminal nodes can reproduce, and they can reproduce only once (Fig. III-1).



**Fig. III-1.** Differences in clonal architecture with clonal branching OFF versus ON. Terminal nodes are those that have not yet reproduced. Terminal branches occur at terminal nodes, while new lateral branches can be established at non-terminal nodes. Dotted lines represent attempts to produce new daughter rhizome nodes (open circles).

Alternatively, when clonal branching is ON, two more sets of C and N tests happen. First, a terminal node that successfully reproduces may immediately (with a species-specific probability of *terminal* branching[[8]](#footnote-8)) try to pass the C and N tests to create a second new daughter rhizome node on a terminal branch (Fig. III-1). Second, non-terminal nodes may (with a species-specific probability of *lateral* branching) try to pass the C and N tests to create new daughter rhizome node on a new lateral branch (Fig. III-1).

In Mondrian there is no maximum branch order imposed; nested branching can continue indefinitely within a genet. At the same time, the mortality of individual rhizome nodes in the trailing end of the chain (or anywhere in the chain) will implicitly limit the size of the continuous, connected set of nodes and will ultimately break a branching genet into separate, independent chains.

### Tests to see if enough C and N have accumulated to make a daughter node

To create a daughter rhizome node, whether it is the first new daughter, a terminal branch, or a lateral branch, the would-be parent node must have enough C and N for three things: (a) to construct the internode; (b) to pass enough C and N to the new daughter node to establish the daughter as a viable individual; and (c) retain a minimum amount of C and N for itself.[[9]](#footnote-9) These parameters are species-specific, meaning different values of these thresholds can be set for different species in the model.

Note that the rhizome node that is trying to create a new daughter rhizome node can not ask for C and N subsidies from its parents in order to do so. (Subsidies from parents back in the chain, within the genet, are only used for creation of new stems; *see* Production of a new stem).

The fact that new rhizome nodes can not be created until adequate N is accumulated in the would-be parent rhizome node is an important way in which N limitation enters into Mondrian. This also has important ramifications for competitive hierarchies as related to species sizes. Because the thresholds of N accumulation for reproduction are expressed as a proportion of maximum size, where N availability is low and competition for N is intense, large-sized species have a more difficult time than smaller-sized species have in accumulating enough N in a single rhizome node to reproduce. This feature of the model was used to demonstrate that in a nutrient-competitive environment, a clonal plant species has an optimal individual size, and that size shifts along a nutrient gradient from small to large-sized individuals as nutrients go from low to high availability (Goldberg et al. 2017).

The maximum number of living individuals allowed in a single species, across the entire modeled space, is controlled by a user parameter ***maxindivs*** in the Parameter input file[[10]](#footnote-10).

### Nutrient limitation, branching, and exploration of space

When a daughter rhizome is established, it is placed at a certain distance from the parent and at a certain angle relative to the line made by the last two nodes in the genet. The distance is controlled by a user parameter that is species-specific: ***cidist1*** for species 1 (short for clonal internode distance), ***cidist2*** for species 2, and so on. The angle is chosen stochastically within a range controlled by a user parameter that is also species-specific: ***MaxVRAngle1*** for species 1, ***MaxVRAngle2*** for species 2, and so on. The daughter will be established stochastically within a “V” from the negative to the positive value of this angle. All of these parameters are set by the user in the Parameter input file[[11]](#footnote-11).

In an environment where nutrient availability is heterogeneous, a large value of ***cidist*** and the lack (or low probability) of branching will allow a species to explore farther, more rapidly, for a high-nutrient patch. A smaller value of ***cidist*** and the ability to branch more frequently will allow a species to better exploit a high-nutrient patch, once one is found. Note that in order for a node to produce a new branch in Mondrian, enough C and N must be accumulated to create the new daughter on the branch; this means branching is more likely to occcur in Mondrian in nodes located where nutrient availability is higher. We believe this may be realistic and suggest that it would be an interesting research question to explore by combining modeling and empirical experiments.

## Plant mortality

Plants have separate aboveground and belowground tissues. Aboveground tissues senesce at the end of each growing season and become standing litter. Belowground rhizome nodes can persist alive through the winter, and re-generate another aboveground stem in the next growing season. A plant individual is considered alive if its belowground rhizome node is alive; if the rhizome node dies, the individual is dead. When mortality occurs in the belowground rhizome node, any living aboveground stem dies too and becomes standing litter. All C and N subsidies that would need to pass through that rhizome node are also cut off.

Mortality in the rhizome node can occur in the following ways.

### Inability to make a stem

If the rhizome node gets to the end of the growing season without being able to establish an aboveground stem, it may die, or it may live through the winter to try again next year. Whether it dies or persists is controlled through a parameter set by the user: ***BGpersist***. If ***BGpersist*** = 1 then the rhizome node will die at the end of the first growing season that it could not make a stem; if ***BGpersist*** = 2, it will die at the end of the second growing season that it could not make a stem, and so on. (Note that this tally of unsuccessful years is cumulative over the lifetime of the rhizome node; it is not re-set to zero if the rhizome node has a successful year.) The reason this is a parameter set by the user is to allow the model user to potentially experiment with the effects that different values of this parameter would have on plant competition in clonal plants over longer time periods.

### Flooding mortality #1

Mondrian includes a switch, ***FloodMortality***, that determines whether mortality can occur due to flooding (see Model Structural Options).

For the flooding mortality #1 mechanism, Mondrian tracks how long (in days) each individual plant stem has been completely submerged due to flooding. For any individual plant stem, if it is continuously and completely submerged for the number of days that matches the flooding mortality parameter ***FloodMortDays***, the aboveground stem dies, but the rhizome node continues to live. The parameter ***FloodMortDays*** is species-specific and set by the user. This would allow a researcher to experiment with the interaction between hydroperiod duration and plant mortality due to flooding, or to explore how differential flooding mortality effects on different species might affect plant competition, plant invasion, or plant community dynamics during and after years with high water. The fact that the height of each individual plant determines whether it is submerged means that there will automatically be some species differences in flooding mortality where species differ in size.

* Example 1: Suppose ***FloodMortDays*** = 60 days for all species and the water level stood at 20 cm depth for 60 days. This would kill the stems of all individuals (of all species) of height 20 cm or less, but not individuals of height 21 cm or greater. (For simplicity, this example assumes MUCK has zero thickness. Because plant individual heights are measured from the top of the MUCK, whereas the water depth is measured from the bottom of the MUCK – the MUCK / SOM interface – Mondrian automatically accounts for MUCK thickness when determining whether a plant is submerged by using the individual plant height measured from the top of the MUCK plus the average MUCK thickness in its grid cell.)
* Example 2: Suppose ***FloodMortDays*** = 40 days for species 1 and 60 days for species 2, the water level stood at 20 cm depth for 50 days (or for any continuous period between 40 and 59 days), and this was followed by a quick lowering of the water depth to zero by day 60. This would kill the stems of all individuals of species 1 that were shorter than 20 cm, but no individuals of species 2 regardless of their height. (Again, for simplicity the numbers in this example assume MUCK has zero thickness.)
* Example 3: Suppose ***FloodMortDays*** = 60 days for species 1, the water level stood at 20 cm for 50 days, lowered to 15 cm for two days, then went back up to 20 cm for another 8 days. This would kill the stems of all individuals of species 1 that were 15 cm or shorter during the entire period, because they would be completely submerged for the 60 day period. In this example, no plant individuals of species 1 that were 16 cm tall or taller would be killed by flooding, because there was a 2-day period in the middle of the 60 days in which they were not completely submerged. Note that for each individual plant, once it ceases being submerged even for one day, its tally of continuously submerged days re-sets to zero.
* Example 4: Note that plants continue to grow normally when submerged (at present there is no light attenuation or other growth reduction from being submerged). If ***FloodMortDays*** = 60 days and the water depth were 20 cm for 60 days, an individual plant could begin the 60 day period at any height less than 20 cm and grow above the 20 cm water level before the end of the 60 days, thus no longer being submerged. Rhizome nodes can also continue to reproduce and establish new stems while being submerged.

### Flooding mortality #2

This is a second, independent mechanism for mortality to occur due to flooding. (This second mechanism is also turned on or off by the ***FloodMortality*** switch.)

Through this second mechanism, if the water level is above some long-term depth for 2 years, all plants in the entire model space are killed, including their aboveground stems and all of their rhizome nodes. The long-term depth at which this occurs is nominally 1.2 m, but it can be changed by the user. It is specified by the parameter ***LTflooddepth*** in the Parameter input file (where LT indicates long-term.) (For simplicity, MUCK depth is not taken into consideration here.)

Since flooding mortality #2 kills all plants in the space, including their belowground rhizomes, the only way for plants to grow after that would be to re-seed in, either through the background seeding rate, or in another seeding cohort specified by the user in the Seed input file.

### Maximum longevity

A rhizome node will die over the winter if it has reached its maximum longevity. This is controlled by a species-specific parameter ***MaxLongev***, expressed in years, set by the user in the Parameter input file.

## Litter production

This section focuses on natural senescence and litterfall in the absence of wetland management; for differences that occur as a result of burning, mowing, or herbicide, *see* Management treatments.

### Senescence

All aboveground (stem) tissue is annual; it senesces at the end of the growing season. During senescence, N is resorbed from stem tissue into the belowground rhizome node of that individual. The proportion of N resorbed is a species-specific parameter, ***resorbProp***, set by the user in the Parameter input file. Since this process of resorption lowers the C:N ratio in the belowground rhizome node, this is another example of how the submodel of N source-sink physiology in Mondrian causes the C:N ratio to “float” in belowground tissue rather than to stay constant. The extra N, which is translocatable, is thus stored in the rhizome node until the next growing season when it could help the rhizome node to establish a stem.

Once senesced, the aboveground stem is no longer living. Its C and its remaining N are then counted in the standing dead litter pool for that grid cell. The standing dead litter remains standing through the winter. In the following spring, the standing dead litter shreds over approximately a 45-day period beginning in day 60 (thus from around March 1st  to April 15th). As it shreds, its C and N are transferred from the standing dead C and N pools to the aboveground litter C and N pools. For bookkeeping purposes, this transfer is what is summed and reported in the main output file as the aboveground litterfall flux.

Mondrian includes a switch that allows litterfall to be distributed in one of two ways, mean-field across the entire model space, or spatially concentrated near the grid cell that produced the litter (for more detail *see* Model Structural Options).

### Belowground litter produced through mortality

When a rhizome node dies, all of the C mass and N mass contained in the belowground tissue of the individual are added to the belowground litter C and N pools in its grid cell. There is no resorption of N into other living tissues in the genet.

## Decomposition and N dynamics in detritus

### Temperature

Mondrian allows a constant mean annual air temperature, or a rising or declining trend in mean annual air temperature. In the Parameter input file, the user simply sets ***TambAnnAvgStart***, the starting mean annual air temperature, and ***TambAnnAvgEnd****,* the ending mean annual air temperature. If these two are the same, the mean annual air temperature will be constant throughout the model run. At present, air temperature only has a potential effect on decomposition (depending on other model ‘switches’); it has no effect on growing season length or on plant growth.

Mondrian includes a switch that allows the user to turn on, or off, a seasonal fluctuation in air temperature (*see* Model Structural Options). If seasonal temperature fluctuation is switched on, then the user-set parameter ***TambMaxDev*** in the Parameter input file determines the maximum intra-annual deviation from the annual mean air temperature. Air temperature then follows a sinusoidal intra-annual pattern, with a minimum on January 1 (the mean annual average for that year minus the maximum deviation) and maximum on July 1 (the mean annual average for that year plus the maximum deviation).

Mondrian includes a switch that turns an air temperature effect on litter decomposition on or off (*See* Model Structural Options). If this switch is turned on, a temperature factor for litter decomposition (TfactLit in the model code) is calculated as follows:

(eqn. III-5)

Where T is air temperature in degrees Celsius. This function is a Q10 relationship where the factor is equal to 0.5 at 10°C, 1 at 20°C, and 2 at 30°C. In calculating litter decomposition, this factor is multiplied directly into the absolute mass loss.

Mondrian also includes a switch that turns an organic-matter temperature effect on organic-matter decomposition (the MUCK and MSOM pools) on or off (*See* Model Structural Options). If this switch is turned on, a temperature factor for organic matter decomposition (TfactOM in the model code) is calculated as follows:

(eqn. III-6)

Where TOM is organic-matter temperature in degrees Celsius. This function is the same Q10 relationship as that for litter (eqn. III-5).

Mondrian includes another switch, ***TeffLitOM***, that determines whether organic-matter temperature (MUCK and MSOM) are the same, or different, from air temperature (*see* Model Structural Options).

### Aerobic versus anaerobic decomposition rates

In the Parameter input file, the user supplies first-order (exponential) decomposition *k* constants for litter (the same *k* constant applies to above- and belowground litter pools), MUCK, and MSOM. These decomposition constants are used unmodified under aerobic conditions. However, decomposition is slowed if these pools are inundated either partially or fully.

Mondrian calculates a *5-day trailing average water level*, i.e. the average water level over the preceding 5 days. The trailing average water level, together with the vertical upper and lower bounds of the litter, MUCK, and SOM pools (Fig. II-2), are used to determine which proportion of each of these detrital pools is aerobic (above the 5-day trailing average water level) or anaerobic (below the 5-day trailing average water level) on a daily basis. If all or part of each pool is anaerobic, the decomposition rate of the anaerobic portion is slowed. Parameters that control the degree of this slowed decomposition are ***LitAnmod, MuckAnmod,*** and ***MSOMAnmod,*** (where Anmod is short for ‘anaerobic modification’), supplied by the user in the Parameter input file.

### Detrital mass losses and transfers

Detrital C and N dynamics, including litter, MUCK, and MSOM pools, are simulated separately within each grid cell.

Litter in the aboveground litter pool of each grid cell and litter in the belowground litter pool of each grid cell both undergo first-order (exponential) mass loss controlled by a single first-order *k* constant (Currie et al. 2010), which is a species-specific parameter set by the user in the Parameter input file. These litter layers are made up of litter from different species that accumulate over a few years. Different species may have different first-order *k* decomposition constants. To calculate the decomposition *k* constant of the aboveground litter pool as a whole (and similarly for the belowground litter pool as a whole), Mondrian uses a mass-weighted single *k* value based on the masses of litter inputs by species and their respective species-specific *k* constants from the preceding year.

Fresh plant litter typically has a C:N ratio that is higher than that of partially decomposed detrital pools. In the litter pools, during decomposition, C mass is lost (mineralized to CO2) but N mass is retained if the litter mass in the pool overall has a C:N ratio greater than a critical ratio (***LitCritCNRat***, set by the user in the Parameter input file; we use 30:1 as a default value). If, as C mass is lost through decomposition, the C:N ratio of a litter pool becomes low enough to equal the critical C:N ratio, then N mineralization to the available N pool occurs at the same rate of C mineralization (mass loss).

Here we describe the transfer of aboveground litter to MUCK. Concomitant with decomposition mass loss, some portion of the aboveground litter pool is transferred during each time step to MUCK in the same grid cell. This transfer is expressed as a proportion of litter mass loss, ***GCPhumProp***, (short for Grid Cell Pool humification proportion), set by the user in the Parameter input file. Nominally we use a value of 0.2 for this parameter, although admittedly this process is difficult to quantify in wetlands and likely to be highly variable. The C and N that are transferred out of the aboveground litter pool and into the MUCK pool in each grid cell are transferred at a C:N ratio set by the user through the parameter ***HumifCNrat*** (short for humification C:N ratio; we use a nominal value of 54:1). In reality, the separation of litter and MUCK layers in the field is ambiguous, and this transfer process can include humification to some extent, but also includes the transfer of non-humified aboveground litter to MUCK, particularly when the litter and MUCK are inundated.

For the belowground litter pool, transfers out of the pool occur in an identical manner, except that some proportion of the transfer goes to MUCK and some goes to MSOM based on the relative vertical positions of each of these pools (Fig. II-2) in each particular grid cell on each particular day.

Mondrian includes a mechanism for some MUCK in each grid cell to be transferred to MSOM, which has a much slower turnover rate. This rate of transfer is controlled by the parameter ***kBioturb***, set by the user in the Parameter input file. This is a one-way transfer of C and N from MUCK to MSOM that conceptually represents two processes: bioturbation and eluviation of particulate matter. This process only occurs when the MUCK pool is aerobic

### N immobilization and mineralization

The concept of N “immobilization” is sometimes used differently by different investigators. In Mondrian, no detrital pools are able to remove N from the inorganic available N pool in a grid cell. However, litter pools can retain and concentrate N until the pool reaches the critical C:N ratio. If decomposition rates are rapid and fresh litter is rich in N, N could be mineralized from litter. More commonly, N concentrates in litter and is passed to MUCK, where it is mineralized. N is mineralized from MUCK at the rate of MUCK turnover. MUCK turnover is typically intermediate between litter decomposition (which is faster) and MSOM turnover (which is much slower). If a wetland retains N over time, much of it accumulates in the MUCK detrital pool, which can then produce relatively large fluxes of N mineralization. This becomes an important source of ecosystem-internal N cycling, often a greater source of N for growing plants than wetland N inflows in Mondrian.

### Detrital export from the wetland

Detrital export from wetlands is a process that is not well understood or well quantified. Floating mats of wetland plant litter are often seen in coastal wetlands and are sometimes seen in open water. These may be flushed from wetlands during periods of high water level and flow. At the same time, some organic compounds are undoubtdedly leached from wetland litter and muck as particulate C or dissolved organic C and flushed into receiving waters. Mondrian combines these into a single process called detrital export. Its rate is controlled through the parameter ***DetExpProp***, set by the user in the Parameter input file. When detritus is exported in this manner, its C:N ratio is equal to the overall C:N ratio of the aboveground litter layer.

## N inflow, ecosystem N retention, N biogeochemistry, and N export

A strength of Mondrian is that it includes ecosystem internal N cycling as well as ecosystem N inflow, retention, and export. Community and ecosystem processes are strongly coupled through the biogeochemical cycling of C and N. Ecosystem internal N cycling is driven by plant uptake and N dynamics in detritus pools (litter, muck, and sediment organic matter). Together, these ecosystem processes exert a strong control on ecosystem N retention or loss. For example, if the wetland internal N cycling is in steady state, plant N uptake would be constant over time, N mineralization from each detrital pool would be constant over time, and N inflows would equal N export (wetland N retention would be zero). If something then changed to allow more biomass production, for example N inflows increased (in a situation where N is limiting), an increase in plant N uptake and litter production will retain N, increasing the N retention in the wetland ecosystem. This increase in wetland N retention would continue until the ecosystem reached a new equilibrium with the higher level of N inflow, which could take decades or longer.

### N inflow in precipitation

In the parameter file, the user sets the value of ***NinputPpt***, the flux of N in precipitation in g N m-2 y-1. We consider this to represent total N in precipitation: wet + dry deposition of NO3-, NH4+, and atmospheric organic N (Neff et al. 2002). This N flux is added to the regional N pool on a daily basis, distributed evenly throughout the year.

### N inflow from land

This represents combined N inflows from surface runoff, streams and rivers, and from groundwater movement into the wetland. The user decides on the total annual amount of N inflow and on its seasonal distribution through the values of parameters set in two input files, described immediately below. In addition, the user can make the N inflow from land change over time (increasing or decreasing over a period of years or decades, for example) or make its seasonal distribution change over time.

There are two steps. First, in the Scenario input file, the user sets a daily value of N input from land in g N m-2 day-1. Mondrian reads this value for each individual day of the model run. Here, in the Scenario input file, is where the user has complete control over making this N inflow vary seasonally or increase or decrease over time. Use the Excel file *ParaFileSetup.xls* to set this up and then copy and paste the values into the Scenario ASCII text input file. Admittedly this is tedious, but it gives the user complete control to make any desired seasonal dynamics and/or trends in N inflow and it does not need to be set up more than once. Various Scenario input files can be setup, saved and re-used.

**Example**: Suppose a researcher wanted to test the effect of increasing, versus flat, versus declining N inflows over a 20 year period. She could create three of these Excel files based on *ParaFileSetup.xls* , give them different names, and save them. The Excel file *ParaFileSetup.xls* does not need to keep that specific name; it is just a tool to help the user create the values that will be copied and pasted into the Scenario ASCII text input file, which is what Mondrian reads. In this example, in the three different Excel files, the researcher could use formulas in a separate column to easily create daily values of N inflow that increased or decreased steadily over a 20 year period, then copy and paste the values (using the Excel feature “paste values”) into the appropriate column. Then these could be copied and pasted into the Scenario ASCII text input file.

The second step in determining N inflow from land is explained here. For each day of the model run, Mondrian takes the value of N inflow from land from the Scenario input file (g N m-2 day-1) and multiplies it by the value of ***NinputLandMult***, which is a user-set parameter in the Parameter input file that is constant through the model run. If the user wants the daily values of N inflow to be used exactly as listed in the Scenario input file, she would set the value of ***NinputLandMult*** = 1. If she wanted to double all of the daily values listed in the Scenario file, she would set the value of ***NinputLandMult*** = 2.

There is an important reason for having the additional parameter ***NinputLandMult*** in the parameter file. If you have a seasonal distribution of N inflows in the Scenario file and you simply want to double or triple the annual N inflow while keeping the same seasonal pattern, ***NinputLandMult*** allows you to do that easily. In addition, if you are doing a large factorial set of model runs, ***NinputLandMult*** can be one of the parameters in the Parameter input file that you change systematically using the Batch input file (*See* Workflow for setting up and running Mondrian).

One final note: When we want to do Mondrian model runs with N inflow constant throughout the year and level over time, we use a value of 0.00274 g N m-2 day-1 for each day in the Scenario file. Added up over a year, this produces an N inflow from land of 1 g N m-2 y-1. Then the value of ***NinputLandMult*** in the Parameter input file will represent an inflow flux of N in g N m-2 y-1. For example, if ***NinputLandMult*** = 10, then the flux of N inflow from land will be equal to 10 g N m-2 y-1.

### N diffusion into grid cell pools and ecosystem N export

As described above (*See* Model Structure), inorganic N is stored in a pool in each individual grid cell (where it is available inorganic N for plant uptake), and it is also stored in a regional N pool that spans the entire space and is not plant-available.

The purpose of the regional N pool is to receive inflows of inorganic N that enter the ecosystem, exchange inorganic N with the grid cell pools of inorganic N, and then to serve as a source for N export from the wetland through hydrologic flushing.

Exchanges between the regional N pool and grid cell pools work as follows. On a daily basis, inorganic N is exchanged between each individual grid cell and the regional N pool. This is meant to conceptually represent diffusion of N from water moving horizontally through the wetland, into pore spaces where it is plant-available, or, depending on concentration gradients, the opposite: diffusion of N out of sediment pore spaces and into the water moving horizontally through the wetland. In the model this is carried out by simply transferring N along the concentration gradient, in either direction, in the amount that is needed to equilibrate the N concentration per unit area in each individual grid cell and in the regional N pool, while preserving mass balance. Thus the spatial heterogeneity in available N that may exist among grid cells can not persist for more than one day because each grid cell equilibrates with the regional N pool on a daily basis.

We recognize that this is an oversimplification of how N advection and diffusion occurs in wetlands. At the same time, it offers advantages by providing a mechanistic N sink or source for water moving through the wetland. If the plants and organic matter in the grid cells are N-poor relative to the ecosystem N inflow, there will be a net diffusion of N into the grid cells where it will be taken up by plants, thus lowering N availability in the grid cell, causing more diffusion into grid cells the following day. At the end of the growing season, the N in plant tissues will enter detrital pools in litterfall, and accumulate in litter, MUCK, and MSOM pools over time. The rate at which this occurs is a mechanistic model of wetland N retention, and it will be related to the accretion or loss of organic detritus. Alternatively, if the detrital pools are N-rich relative to the ecosystem N inflow, high rates of N mineralization from detrital pools will cause N to accumulate in the grid cell available N pool, where it will allow plant N uptake and growth much greater than that which could be supported by the N inflows alone.

Within the model code and in the model main output file, the exchange of inorganic N between the regional N pool and grid cell pools is called “N transfer.” It can occur in either direction, and the amounts and directions of the transfers can differ from one grid cell to another and from one day to another. Mondrian tracks the net amount of N transfer that occurs each year and prints this in the main output file. A positive value represents a net transfer of N from the grid cell pools into the regional N pool; a negative value represents a net transfer from the regional N pool into the grid cell pools.

Each simulated day, after the regional N pool receives inputs, and after N cycling occurs within each individual grid cell, and after all grid cell inorganic N pools equilibrate with the regional N pool by making N transfers (representing diffusion), then a small portion of the N in the regional N pool is exported from the ecosystem, representing hydrologic flushing. The rate of hydrologic flushing is controlled by a single parameter, ***RNPINexportProp***, which is set by the user in the Parameter input file and is constant throughout a model run. This is directly related to the residence time of water in the wetland, which is thus set by the user. For example, a value of 0.015 / day corresponds to a hydrologic residence time of 1/0.015 = 67 days. A higher value of ***RNPINexportProp*** would correspond to faster moving water and a lower residence time of water in the wetland. By experimenting with the value of ***RNPINexportProp*** and other variables and drivers, an investigator could explore relationships among water residence time, water levels, ecosystem N retention, plant invasions, C accretion, and numerous other community-ecosystem phenomena.

Mondrian calculates ecosystem N retention in two ways and prints the results in the main output file (*See* Main output file).

### Active Zone depth and aerobic / anaerobic conditions

Nitrification and denitrification rates in Mondrian (described below) take place within a vertically-defined ‘active zone.’ Conceptually, the top and bottom of the vertical ‘active zone’ represent where the available NH4-N and NO3-N pools are located for the purposes of determining what proportions of those pools are in aerobic vs anaerobic conditions on a daily basis, as water level fluctuates. The proportions that are aerobic vs anaerobic directly determine the proportion of the NH4-N pool that is subject to nitrification (requiring aerobic conditions), and directly determine the proportion of the NO3-N pool that is subject to denitrification (requiring anaerobic conditions). As the water level fluctuates, layers of detritus and soil that were previously inundated become immediately aerobic when the water lowers. Layers that were previously aerobic have a 5-day delay for turning to anaerobic after the water level rises.

Here we describe the locations of the top and bottom of the active zone. The top of the active zone differs slightly for the available NH4-N and NO3-N pools. For the available NH4-N pool, the top of the active zone is located 0.05 m up into the aboveground litter layer, upward from the top of the Muck layer (or to the top of the aboveground litter layer, if it is less than 0.05 m thick). For the available NO3-N pool, the top of the active layer is located at the top of the Muck layer. The bottom of the active zone is the same for the NH4-N and NO3-N pools; it is located a certain distance down into the upper layers of the MSOM layer. This is determined by a user-set parameter ***AZDepth***.

The way that the active zone depth affects nitrification and denitrification may be clearer with an example. For example, if the user sets AZDepth = – 0.05 m, then when the water level drops to – 0.05 m, the entire active zone is aerobic. Alternatively, if the user sets AZDepth = – 0.15 m, then when the water level drops to – 0.05 m, there will still be 10 cm of the active zone that is anaerobic.

### Nitrification

Nitrification was added to Mondrian in version 4.2.

Nitrification in Mondrian is source-constrained, where the source is NH4-N. Within a daily timestep, within each grid cell, N immobilization in litter and mineralization from all detrital pools are calculated first, building up the available NH4-N pools within each grid cell. Next, plant N demands are calculated and plant uptake of NH4-N and NO3-N take place. After this, if a significant amount of available NH4-N remains in the grid cell, it is subject to nitrification.

Note that hydrologic flushing of water (the inverse of water residence time) has an effect on nitrification rate, because as NH4-N diffuses out of grid cell pools into the regional NH4-N pool and is flushed out of the wetland, it is not then available for nitrification the following day.

Daily nitrification flux (g N / m2 day) within a grid cell

= NitrifParm \* PropNH4Aer \* TfactMUCK \* (Available NH4-N in the grid cell)

Where ***PropNH4Aer*** is a model internal variable that represents the proportion of the available NH4 pool that is aerobic (based on the daily water level in relation to the top and bottom of the vertical ‘active zone’), ***TfactMUCK*** is a model internal variable that conveys a temperature effect on nitrification (see below), and ***NitrifParm*** is a unitless scaling parameter between zero and 1. Conceptually, ***NitrifParm*** can be thought of as the volumetric proportion of microsites where nitrifiers are present and other conditions are optimal for nitrification. In practical terms, this scaling parameter is probably not possilbe to measure operationally and probably needs to be calibrated in Mondrian to produce a certain flux of nitrification at the ecosystem level under certain conditions.

### Denitrification

Denitrification was added to Mondrian in version 4.2.

Denitrification in Mondrian is source-constrained, where the source is NO3-N. Within a daily timestep, within each grid cell, plant N demands are calculated and plant uptake of NH4-N and NO3-N take place. After this, nitrification takes place (see description of nitrification above). Afterward, if a significant amount of available NO3-N is present in the grid cell, it is subject to denitrification.

Note that hydrologic flushing of water (the inverse of water residence time) has an effect on denitrification rate, because as NO3-N diffuses out of grid cell pools into the regional NO3-N pool and is flushed out of the wetland, it is not then available for denitrification the following day.

Daily denitrification flux within a grid cell =

= DntrParm \* HetRespFact \* PropNO3Anaer \* (Available NO3-N in the grid cell).

Where ***PropNO3Anaer*** is a model internal variable that represents the proportion of the available NO3-N pool that is anaerobic (based on the daily water level in relation to the top and bottom of the vertical ‘active zone’), ***HetRespFact*** represents the rate of heterotrophic CO2 production in detritus in the grid cell, and ***DntrParm*** is a scaling parameter. ***DntrParm*** (similarly to ***NitrifParm***) can be thought of as the volumetric proportion of microsites where denitrifiers are present and other conditions are optimal for denitrification. In practical terms, this scaling parameter is probably not possilbe to measure operationally and probably needs to be calibrated in Mondrian to produce a certain flux of denitrification at the ecosystem level under certain conditions.

***HetRespFact*** is used as a variable affecting denitrification in Mondrian because denitrifier microorganisms require a source of reduced, detrital C and in our literature review we found that some other wetland ecosystem models simply use hetertrophic respiration of CO2 to quantify this. Heterotrophic respiration of CO2 scales directly with both the sizes of detrial pools and their lability (ease of decomposition). In Mondrian, heterotrophic respiration of CO2 also already includes the effect of temperature explicitly, because temperature affects rates of detrital decomposition. In Mondrian, ***HetRespFact*** combines heterotrophic respiration in Muck, MSOM, and BGL detrital pools but not the AGL detrital pool. Its effect on denitrification is benchmarked using a parameter in the Parmeter input file, ***HetRespBench***. When heterotrophic respiration of Muck, MSOM, and BGL equal the benchmark value, then ***HetRespFact*** = 1 and it neither increases nor decreases the rate of denitrification; the value of this factor scales linearly with the flux of heterotrophic respiration from these pools. While this is calculated on a daily basis, the units of ***HetRespBench*** are expressed on an annual basis. We suggest a nominal value of 100 g C / m2 y for ***HetRespBench*** based on model testing.

### Temperature effects on nitrification and denitrification

The temperature effects on nitrification and denitrification are both conveyed through the model internal variable ***TfactMUCK***. A simplifying assumption is made here that if the temperatures of litter, Muck, and MSOM differ, the Muck temperature is used. The form of ***TfactMUCK*** is a Q10 function with a value of 0.5 at 10° C, 1 at 20° C, and 2 at 30° C.

The user has the ability to switch this temperature effect ON or OFF through the parameter ***TeffOM***. When the ***TeffOM*** switch is OFF, then ***TfactMUCK*** is always equal to 1.0, so temperature has no effect on nitrification or denitrification. (Note that turning ***TeffOM*** to OFF also turns off the temperature effect on decomposition.)

In addition, when Muck temperature falls below zero in Mondrian, both nitrification and denitrification are shut down (they become zero), no matter what switches the user has set.

### Wetland N retention

Wetland N retention, or ecosystem N retention, has the units of a flux; it refers to the proportion of N inflows that are retained in the wetland ecosystem. This a highly integrative quantity in Mondrian because anything that affects N cycling, plant N uptake, N mineralization, denitrification, or hydrologic flushing (residence time of water) will affect wetland N retention. Losses of N from the wetland ecosystem include hydrologic flushing of NH4-N and NO3-N, denitrification, and N in exported detritus.

Ecosystem N retention is calculated in two alternative ways; see the section on the Main Output File for details on the two calculations.

## Model structural options or ‘switches’

A strength of Mondrian is that some processes can be turned ‘on’ or ‘off’. This includes, for example, light competition. When light competition is ‘on’, individual plants shade one another within a gride cell and respond to the reduced light environment in each grid cell. When light competition is ‘off’, plants compete only for nutrients but light and shading have no effect on their competition. While turning light competition ‘off’ is hypothetical and does not occur in nature, having the ability to turn such processes on or off in the model is a powerful means to explore the interaction of ecological processes and cause-and-effect in ecological outcomes. For example, an investigator can explicitly and directly explore the effect of light competition on a particular simulated ecological outcome such as invasion success.

We call the set of these switches and options “model structural options.” They are set by the user in the Parameter input file. They can also be systematically varied in a large set of model runs; e.g. with an identical subset of runs conducted with and without light competition, or with and without spatial litterfall, or the factorial combination of these two or any such combinations of structural options.

A complete, current list of the model structural options is given here, with some details about each:

***LitSuppressVR***: (0 or 1: 0 = false = OFF; 1 = true = ON). When ON, a thicker litter layer increasingly suppresses vegetative reprodution. The conceptual idea here is that highly productive wetland plants, including large invasive plants, produce such thick accumulations of litter that it is more difficult for smaller-sized native plants to push up through the litter to obtain sunlight. When OFF, the thickness of the litter layer has no effect on vegetative reproduction.

***LitSpatDistr***: (1 or 2: 1 = mean field; 2 = spatial). Litter spatial distribution option. When mean field, all litter production from all plants is distributed in the litter layer evenly throughout the model space. When spatial, litter production from a plant is distributed evenly among the set of 9 grid cells that include the plant and that are adjacent to it. This allows investigators to explore the manner in which large plants may cause litter to accumulate heterogeneously in a particular area and what the effects of such spatially heterogeneous litter accumulation may have on ecological processes.

***TeffLit***: (0 or 1: 0 = false = OFF; 1 = true = ON). When ON, seasonal or daily temperature changes affect the rate of litter decomposition.

***TeffOM***: (0 or 1: 0 = false = OFF; 1 = true = ON). When ON, seasonal or daily temperature changes affect the rate of sediment organic matter (SOM) and MUCK decomposition.

***TeffLitOM***: (0 or 1: 0 = false = OFF; 1 = true = ON). When ON, the aboveground litter layer mass, as it accumulates, causes a lower temperature in MUCK and MSOM. This is conceptually meant to represent reflection of shortwave insolation and thermal insulation provided by a litter layer during the growing season. (Note: this is present in the model for future expansion, but has not been parameterized or tested, so it should not be used; it should be set to 0 = OFF.)

***TeffSeas***: (0 or 1: 0 = false = OFF; 1 = true = ON). When ON, air temperature varies in a seasonal, sinusoidal fluctuation about the mean. When OFF, air temperature is constant throughout the year.

***PlantDistrOpt***: Plant distribution option: (1 or 2: 1 = a specific, small set of plant individuals distributed in a known way for model testing. This is used only by the model development team for testing; 2 = plants are seeded in randomly in cohorts as specified in the Seed input file.) Model users should always use option 2.

***LgtComp***: (0 or 1: 0 = false = OFF; 1 = true = ON). Light competition. When ON, individual plants compete for light and also compete for nutrients. When OFF, plant heights and shading have no effect on competition; individuals within a grid cell compete only for nutrients. (Note that when light competition is OFF, not only is there no competition between species for light but there is also no self-shading within a species, so if nutrient inflows are high, community NPP is usually greater that it is with light competition ON.)

***FloodMortality***: (0 or 1: 0 = false = OFF; 1 = true = ON). When ON, mortality of plants can occur due to flooding. When OFF, flooding has no effect on plant mortality, neither through the flooding mortality #1 mechanism nor through the flooding mortality #2 mechanism. (*See* Ecological Processes / Plant mortality.)

***ClonalBranching***: (0 or 1: 0 = false = OFF; 1 = true = ON). When ON, within a genet, during reproduction of new daughter rhizome nodes, branching can occur. When OFF, branching can not occur.

# IV. Management treatments for control of invasive plants

## Overview

Mondrian includes subroutines that allow the user to simulate three common methods of invasive plant control: fire, herbicide application, and mowing (with or without biomass removal). User-specified parameters dictate the years and days of the year in which management treatments take place, together with other parameters that dictate other aspects of teach treatment as described below. These user-specified parameters are all contained in the *Parameter input file*.

### Fire

Fire is modeled as a uniform proportion of biomass loss across the entire model space, without stochasticity or patchiness. High fuel-load grid cells will have a greater *absolute* amount of fuel burned, but the *proportion* is constant across all grid cells. The user specifies a management period by specifying the starting and ending year during which burning occurs, on up to five days of the year in each year of the management period. The burn occurs in each grid cell on the specified date, after growth calculations are called for that grid cell on that date.

Three main parameters (***BurnTrLitSev***, ***BurnTrLiveEff***, and ***BurnTrNVol***) control fire severity. ***BurnTrNVol*** controls the proportion of N volatilized in the fire. High-temperature fires can be simulated by increasing this proportion, resulting in higher rates of N volatilization. All carbon in biomass that is burned is volatilized, but the proportion of litter and biomass that burns in the fire is controlled by the remaining two parameters: ***BurnTrLitSev*** and ***BurnTrLiveEff***. These are the proportions of the litter layer C and live plant biomass C that are volatilized in the burn, respectively. ***BurnTrLitSev*** is used to calculate a "burn depth" in the litter layer; above this height all downed litter and standing dead litter is volatilized and removed from the modeling space. Litter below this height is not volatilized and continues to decompose, and live tissue below this height remains alive and can continue growth. If water level is higher than the “burn depth”, the burn depth is set to the water level; in this way, living plants, standing dead stems, and the litter layer can only burn down to the water level. ***BurnTrLiveEff*** is the proportion of the living biomass above the "burn depth" that is volatilized in the fire; the remainder of that individual above the “burn depth” is considered scorched, and converted to litter. As a result, live biomass below the “burn depth” remains alive; above the burn depth a proportion is burned completely and removed from the model space, while the remainder is killed and converted to litter.

### Herbicide application

Herbicide application occurs on up to five days of the year in each year of the user-specified management period. Herbicide application is conceptually modeled as a uniform aerial spray; spot-spraying or selective spraying is currently not implemented in the model. Spray intensity is controlled by the ***HbcidePen*** parameter, which is the distance from the top of the plant canopy (determined by the tallest plants), vertically down into the plant canopy, that the herbicide penetrates. Greater values for this parameter conceptually represent heavier spraying (or less windy conditions) and will result in greater plant mortality.

In each grid cell on the specified application date(s), after growth calculations are called the model determines the maximum height of the canopy in that grid cell. The top of the canopy is then used to calculate the height to which herbicide will penetrate (which is the maximum canopy height minus the value of ***HbcidePen***). For example, if the tallest individual within a grid cell is 1.1 m and HbcidePen = 0.5 m, then herbicide will penetrate to 0.6 m above the soil surface. In this example, any individual within the grid cell that has 10% or more of its aboveground biomass at or above the height of 0.6 m will be killed by the herbicide, i.e., that individual’s aboveground and belowground biomass is converted to litter. Aboveground biomass is converted to standing dead litter and can continue shading surrounding plants until the standing dead litter shreds and falls (which typically occurs the next spring in Mondrian). Plants with less than 10% of their biomass above the herbicide penetration line remain alive and can continue growing. Mortality from herbicide is modeled on a per-ramet basis, not by genet. For the sake of model parsimony, Mondrian currently does not model herbicide translocation from one ramet to the next.

### Mowing

Mowing can occur on up to five different days of the year in each year of the user-specified management period. During mowing, aboveground biomass above the cutting height (***BioCutHeight***) is killed; aboveground biomass below that height and all belowground biomass is alive and can continue growing. Mowed biomass is either converted to litter (if ***BioCutRemoval*** = 0), or if ***BioCutRemoval*** = 1 then it is removed from the modeled space (such as when mowed biomass is removed for bioremediation, composting, or biofuel production).

## Key Differences Among Management Techniques

The main differences among these invasive control techniques are:

* Mowing does not cause direct mortality of any individuals and usually leaves live aboveground biomass that can continue to grow. Fire typically will not leave any living aboveground biomass, but most belowground biomass will remain alive and can create new stems. Herbicide can cause direct mortality of both aboveground and belowground tissues.
* Mowing creates litter on the soil surface (if biomass is not removed), herbicide creates standing dead litter that casts shade, fire removes (volatilizes) most aboveground biomass.
* Fire removes some C and N from the ecosystem; herbicide does not. Mowing removes C and N from the ecosystem if litter removal is turned on (***BioCutRemoval***  = 1), but otherwise does not.

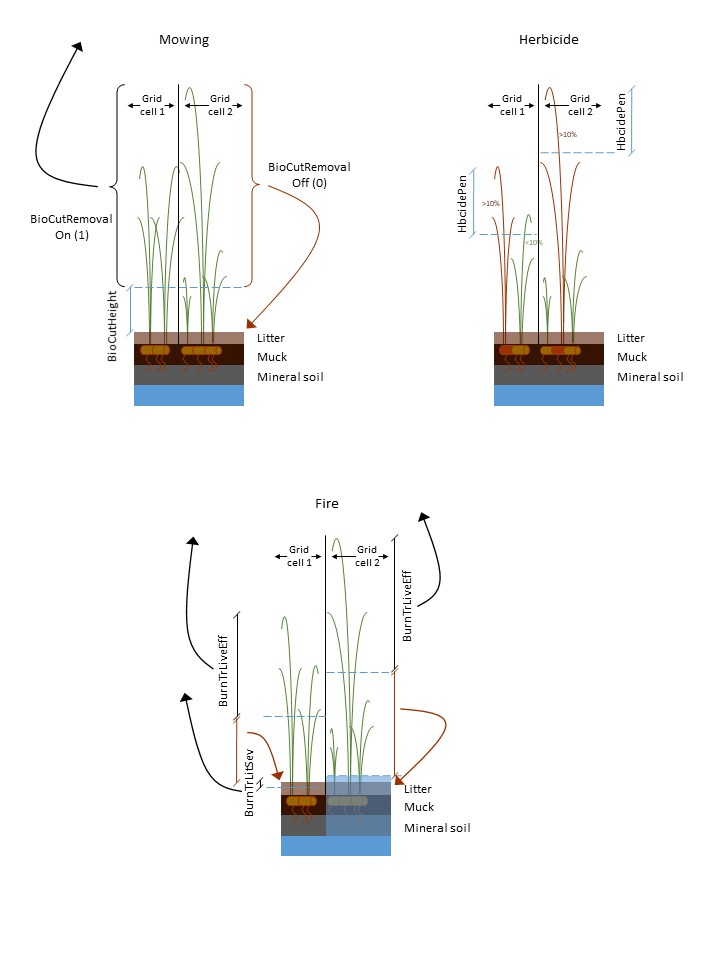
## Management treatments in the vertical dimension

Biocut height and Burn height are defined as height relative to base of the plants, i.e. relative to the top of the MUCK. Penetration of herbicide treatment is measured as a depth of effectiveness from the top of the plant canopy. In this way, herbicide treatment height is also defined as height relative to the base of the plants.

Fig. IV-1 below depicts the vertical manifestation of the three management treatments. Mowing simply moves biomass above the ***BioCutHeight*** line to litter when ***BioCutRemoval*** is turned off (0), or removes the biomass from the modeling area otherwise. Note that ***BiocutHeight*** is defined relative to the base of the plants; i.e. the top of the MUCK layer, not to mineral soil.

Herbicide application sets a vertical line in each grid cell that is the distance specified by the ***HbcidePen*** parameter from the top of the canopy in that grid cell. This parameter therefore is relative to the top of the canopy in each grid cell. All individuals in the grid cell with at least 10% of their aboveground biomass above that line are converted to standing dead litter and left in place. Both aboveground and belowground biomass of killed individuals is converted to litter.

Fire creates two vertical thresholds; one in the litter layer and one aboveground. The bottom threshold is defined relative to the top of the MUCK layer via the ***BurnTrLitSev*** parameter. Aboveground and belowground biomass that falls below this threshold is unaffected by fire. Aboveground biomass above the top threshold (which is controlled by the ***BurnTrLiveEff*** parameter) is volatilized; all C is and a percentage of N (specified by ***BurnTrNVol***) is removed from the model area and the remaining N is returned to the regional available mineral N pool. Biomass that falls between the two vertical thresholds is “charred”, i.e., killed and returned to the litter layer.



**Fig. IV-I.** Depiction of the vertical structure of the three management treatments in Mondrian.

# V. Obtaining and installing Mondrian

## Software and obtaining Mondrian

Mondrian is programmed in Microsoft VisualBasic (hereafter, VB), part of Microsoft Visual Studio, an integrated development environment (IDE). The current version of Mondrian was modified and tested in Visual Studio 2017 running in Microsoft Windows 10 on a PC. The latest version that has been publicly released and freely distributed[[12]](#footnote-12), Mondrian40.exe, is an executable file that can be run on computers running Microsoft Windows without needing the Microsoft Visual Studio 2017 IDE. (Note: the .exe version has not been tested on Apple computers or under different operating systems.) Using the .exe version, model users are able to do thousands of model runs and change a large number of model parameters and drivers that are read from the ASCII input files.

To modify the source code, one needs to open the modifiable code in the Mondrian40.sln file and a number of associated project, form, and module files within the Microsoft Visual Studio 2017 IDE. (*See* Best Practices / Model development). Typically the source code is not distributed, but it can be obtained from the development team in some circumstances[[13]](#footnote-13).

Mondrian40.exe is freely distributed, along with default versions of all of the input files needed to run the model. At present, to obtain Mondrian contact Bill Currie at the University of Michigan ([wcurrie@umich.edu](mailto:wcurrie@umich.edu)) or go to the website <http://williamcurrie.net> to download the model and its input files.

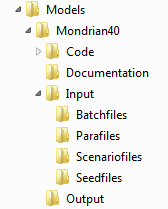
## Fair use guidelines and citing Mondrian

Mondrian was developed by university researchers, using funds from federal and state research grants and supported by universities (*see* Acknolwedgments). It is freely distributed for non-profit use in research, education, and environmental management.

Any presentation or publication using the Mondrian model should cite the original peer-review publication that introduced the model (Currie et al. 2014) together with later publications depending on what model features are being used. Where model enhancements are used such as carbon cycling, hydroperiod, or simulation of management practices, then the peer-reviewed papers describing those model enhancements should also be cited (e.g. Martina et al. 2016, Elgersma et al. 2017). This user guide can also be cited for a variety of purposes: for more detailed technical information about how the model works; model enhancements that are not yet described in the peer-reviewed literature; or guidance on model application such as parameter estimation and model testing and validation.

## Installation and directory structure

There is no ‘installer’ for Mondrian. The simplest way to install it is to create a folder C:\Models on your computer C drive and put all of the files there. Within the C:\Models folder, put the folder Mondrian40 (where 40 is the version number), with all of its subfolders and files organized as they were provided. The folder hierarchy should look like the figure at right. Mondrian needs to know where the input files are located and where to print the output files. The default locations are in C:\Models\Mondrian40 in the subfoldes as shown here. If desired, the locations of input files and the location of the folder to print the output files can be changed, and you can browse to find them at run-time (you will be prompted for this in a dialogue window when the model starts up).



The executable version of the model, Mondrian40.exe (again where 40 is the version number), is located under the Code subfolder, under Code\bin\release. Simply double-click on the .exe file to run the model. Note: The Mondrian40.exe file can be moved to anywhere on the computer and it will work just fine, but its input and output files should be kept in the folder hierarchy under C:\Models\Mondrian40 as shown.

To understand the use of the input and output files, *See* the Workflow section.

# VI. Workflow for Setting up and Running Mondrian

Mondrian can be used in a single model run, small sets of e.g. 5 to 50 model runs, or large sets of 5,000+ model runs. We generally use single model runs only during model development and testing. Once we have a research question we want to explore for a presentation or publication, we develop a planned and consistent set of model runs, where certain parameters or drivers are the same across all model runs while others are varied in a systematic way. In this section we provide an overview of the numerous steps involved in planning and setting up a set of model runs and then organizing and analyzing the results, in other words the workflow involved in doing sets of Mondrian model runs from inception to conclusion.

[develop a graphic of the workflow and a text explanation? Or perhaps it can be a bullet list]

The workflow includes defining the research question or dynamic modeling question, defining a baseline and sets of changes, parameter estimation and initial conditions, gathering input data, doing the runs, post-processing, statistics and graphics for journal articles. Different software packages are typically used in various stages.

## Designing a set of model runs

### Define the dynamic problem to be explored

First, define or articulate the research question or specify the dynamic problem to be explored. This will guide the design of the set of model runs.

Part of this process is to define a *reference mode*. The reference mode is a dynamic model result that a model was designed and calibrated to produce. The model user defines the reference model and sets up the model parameters and drivers and does test runs so that the reference mode is produced in a reliable and understood manner. This will always be hypothetical to some degree, but when included in a set of model runs, the referenc mode offers a baseline for comparison against the other model runs in the set. For example the reference mode might represent a long-term equilibrium of ecosystem N cycling and C balance under low-nutrient conditions, and the equilibrium vegetation community that occurs in that situation. Then the dynamic question might be, if N inputs became elevated by a factor of 10, how would this change N cycling, C storage, and plant community dynamics in comparison with the reference mode?

In designing a set of model runs to address a research question, there are many other factors to consider. How many years will you run the model? If you are introducing an invasive plant species, what year will you introduce invasives, if any? How many different types of communities will you simulate? How many different levels of nutrient inflows will you simulate? In our first publication with Mondrian (Currie et al. 2014) we simulated a native plant community starting in year 1 and allowed it 15 years to go through population-community dynamics and reach equilibrium before we introduced a potential invader plant species in 2 cohorts, in years 15 and 20 (the user specifies these choices in the Seed input file described below). This gave the potential invader species 25 years (years 15 to 40) to attempt to become established and spread before the model run ended, which we found was enough time for the community to reach a new equilibrium. After 25 years, at low N inflows the potential invader failed to become well established, at intermediate N inflows it became established without dominating, and at high N inflows it completely dominated and extirpated the native plants. We chose to include 6 different levels of N inflow to explore the full dynamic range of invasion failure or success.

### Decide on the number of model runs needed in the set

We designed Mondrian so that hundreds of parameters representing plant traits or environmental drivers could be changed by the user, many of them species-specific. For example, traits like maximum relative growth rate (RGRmax), foliar tissue C:N ratio, and dozens of others can be given different values for each species. Part of the reason for this design was to enable different research groups to explore a wide range of research questions in different wetland systems. Mondrian is a complex model, and virtually any plant parameter or environmental driver can be altered by the user.

**Example set of 540 runs**: We develop this here as a detailed example and then refer to it in other parts of this guide. Suppose a user wanted to explore plant invasion success over a range of N inflows in a wetland in her region. She might include **6** levels of N inflow increasing exponentially (e.g. 1, 2, 4, 8, 16, 32 g N m-2 y-1), **5** different plant communities (Table 1), and **2** alternatives of one model structural option or ‘switch’ such as light competition either ‘on’ or ‘off’. Suppose she also wanted to explore **3** different cases of differential values of maximum relative growth rate (RGRmax); one in which all species have the same RGRmax, one in which invasives have a higher RGRmax, and one in which native plant species have a higher RGRmax. Suppose she wanted to include all of these in a fully factorial set with **3** stochastic reps of each case (see below). This overall set would inlcude 6 x 5 x 2 x 3 x 3 = 540 model runs. (In our research we have used Mondrian in such factorial sets of over 5,000 model runs.)

|  |  |
| --- | --- |
| **Name** | **Description** |
| sppcomb1 | 3 native species, no invader |
| sppcomb2 | Invader A into bare ground |
| sppcomb3 | Invader A into established native community (3 native species) |
| sppcomb4 | Invader B into bare ground |
| sppcomb5 | Invader C into established native community (3 native species) |

**Table VI-1.** Five combinations of species that could be used in a large set of model runs

Doing this example set of 540 runs would allow a number of interesting comparisons. By looking at a particular subset of this set, the researcher could examine the effects of different N inflow levels on community-ecosystem processes in her system, all else being equal. By looking at a different subset, she could examine how invader type A might grow and reproduce well on bare ground but not as well into an established native plant community, how this differs along the gradient of N inflows, and how it differs from the success of invader type B. By including the ‘switch’ of light competition either on or off in half of the model runs, she could examine whether light competition was critical, or not, to the differences in invasion success at different levels of N inflow. Doing a planned, systematic set of model runs allows the researcher to be confident that within this set, everything else was kept the same – all of the other model parameters such as tissue C:N ratios, plant maximum sizes by species, growing season start and end dates, seasonal changes in water level, and so on. This is a powerful way to focus in on specific ecological questions.

### Stochastic replicate runs

In designing a set of model runs, we typically include “stochastic replicates” (sometimes abbreviated sreps). If we use 3 stochastic replicates, for example, then every unique combination of parameters and drivers in the set is repeated 3 times. Within this set of 3, the only differences are stochastic, due to random spatial patterns produced by establishing plant locations at the start of each run, random spatial locations in which invaders appear, and random spatial directions in the growth and spread of genets during vegetative reproduction. We have found 3 stochastic replicates to be adequate because standard error bars for model results within each set of 3 tend to be small compared to simulated differences that result from changes in model parameters or drivers. This is controlled by the user-set parameter ***nStochasticruns*** in the Parameter input file (when doing a single model run this should be set to 1).

### Decide on the size of the space and related parameters

Mondrian simulations take place in a square grid of cells where the size of each grid cell, the number of grid cells in the space, and thus the size of the total simulated space, are established by parameters set by the user (see the section on Spatial framework). The size of each grid cell determines the scale at which competition for light and nutrients takes place.

A larger size space will make model runs take longer. More accurately, a greater overall number of plant individuals will make model runs take longer, and all else equal, a larger space will involve simulating a larger number of plant individuals.

In the Parameter input file, the user sets the maximum number of individuals that could be created per species, ***maxindivs***, and the maximum number of genets that could be created per species, ***maxgenets***. Round numbers that we use for these are about 20,000 maximum individuals and 2,000 maximum genets, per species, per m2 of model space. The reason for having these parameters ***maxindivs*** and ***maxgenets*** are so that the computer can allocate adequate memory at the start of a model run. If they are set much higher than needed, Mondrian may over-allocate memory, which could slow down the model depending on a particular computer’s speed and memory. One of our goals in creating and maintaining Mondrian has been to allow the model to run on an ordinary desktop or laptop. If these are set too small and the model run tries to create a new individual or genet beyond these numbers, it will not be created and an error will be logged and printed in the main output file. The model run will still continue but the results should probably not be trusted.

## Parameter estimation and gathering input data

This part of the modeling activity can be divided into three categories: Gathering input data for **drivers**, specifying **initial conditions**, and setting values for model **parameters**.

## The *Parameter* input file

The *Parameter input file* is an ASCII text file that Mondrian reads at run-time. By changing the values of parameters in this text file, the user has a tremendous amount of control over model runs. Distributed with the Mondrian input files is an Excel setup template that provides detailed descriptions of all of the parameters in the *Parameter input file*. We suggest that the user should save an unchanged copy of this Excel setup template with the default values and then use another copy to make changes. To run the model, copy and paste the appropriate columns from the Excel file to the ASCII text file (see the detailed instructions in the Excel file).

It is very important that the order of all of the parameters be unchanged. Mondrian reads the ‘index’ number and uses that as an unambiguous identifier of each parameter. (Note that many of the rows in this file contain ‘UnusedVar’ as the parameter name. These are simply placeholders for future expansion. They need to be kept in the order that they appear.)

Below we list a general overview of the types of data in this file:

Run setup specifications: The user defines the size of the modeled area and grid cells, the number of stochastic reps, and the maximum number of individuals and genets for each species.

Species-specific plant parameters: The user defines a variety of plant traits that can be different for each plant species, including the maximum relative growth rate, the decomposition constant for litter, the belowground and aboveground tissue C:N ratios, and more.

Non-species-specific plant parameters: The user defines a few additional plant traits and characteristics that do not vary among species.

Initial values for grid cell pools: The user sets initial pool sizes for all of the C and N pools in grid cells, including litter, muck, and so on. At the start of a model run, each grid cell is initialized with the same values of all of these pool sizes.

Ecosystem and Biogeochemistry: The user sets numerous parameters that control ecosystem processes, including the proportion of aboveground litter pool that is exported hydrologically via flooding; the bulk densities and C concentrations of various detrital pools; the first-order decay constant for mineral soil organic matter, and others.

Scenario parameters: The user sets “scenario” parameters that do not change over time such as the N input flux in precipitation and the start and end of the growing season. (Note: additional scenario drivers that do change seasonally or that change over time, such as water level and N inflows, are specified in the *Scenario input file*.)

Wetland management: The user sets a variety of parameters that control the simulation of mowing, burning, and herbicide as treatments to manage invasives and restore native plant communities.

Model Structural Options: The user sets a variety of structural options or “switches” that turn certain ecological processes ‘on’ or ‘off’. A complete list of these is given in Table 2. More detailed notes concerning the values of the parameters that set these options are provided in the Excel setup template file*.*

|  |  |
| --- | --- |
| **Name** | **Description** |
| LitSuppressVR | Switch controlling whether litter layer thickness impedes generation of new aboveground stems. Allows exploration of the effects of thick litter layers on the ecology of native plants and invasive plants. |
| LitSpatDistr | Controls whether plant litter is distributed mean-field over the space or in a spatially explicit manner (i.e. closer to the plant that produced the litter). Allows exploration of the effects of spatial heterogeneity in litter thickness by allowing a comparison of an identical model run with mean-field distribution of litter. |
| TeffLit | Switch determining whether temperature effects litter decomposition rate.\* |
| TeffOM | Switch determining whether temperature effects sediment organic matter decomposition rate.\* |
| TeffLitOM | Switch determining whether the litter layer insulates the muck and sediment organic matter from temperature fluctuations.\* |
| TeffSeas | Switch determining whether temperature is constant or varies seasonally. |
| PlantDistrOpt | Determines how plants will be initially seeded in. |
| LgtComp | Switch determining whether light competition is ‘on’ or ‘off’. |
| FloodMortality | Switch determining whether flooding mortality is ‘on’ or ‘off’. |

**Table VI-2.** Model Structural Options or “switches” that can be turned on or off in Mondrian. Note: The model features follwed by “\*”are present but have not been adequately tested.

## The *Seed* input file

Any number of individuals from each species can be introduced to the model space each year. This gives the user a great deal of flexibility to explore a variety of specific research questions and simulation approaches.

In most of our sets of simulations to date, we have used species 1, 2, and 3 to represent native wetland plant species of different sizes (Currie et al. 2014) or different species (Martina et al. 2016), while species 4 is treated as a potential invader (also sometimes modeled with the traits of an invader). We typically “seed in” a number of natives from species 1, 2, and 3, and give them 14 years to become established (model testing has shown that this is an adequate time for the three species to compete with one another and to reach stable levels of densities and NPP by species). We then typically introduce a population of invaders (species 4) in two cohorts, in years 15 and 20.

In model testing we have found that over multi-decade model runs, some genets die through random chance even when conditions are favorable. To counter this and to maintain a stable population of a species, we use a “background seeding rate” that introduces new genets at a low background rate, e.g. 2 new genets per species every 2 years. Conceptually this could be viewed as clones (genets) from outside the model space moving into the space to replenish fine-scale random mortality and to compete with individuals already in the space.

Mondrian runs can also include a background seeding rate for an invasive species (or for any plant species that arrives after the others have become established). The *Seed input file* is set up so that this background seeding rate of an invader occurs only after the initial wave of invasion. This allows users to set up a situation in which a species arrives in a certain year, but not before that. Conceptually, this could be viewed in two ways; (1) as a continuing arrival of propagules from other sites, or (2) additional invader genets in nearby patches moving randomly into the patch being simulated.

## The *Scenario* input file

The *scenario input file* allows users to define several environmental inputs in Mondrian on a daily basis. The user can construct hypothetical scenarios such as daily, seasonal, or decadal time-scale fluctuations of these drivers, or use observational data from a particular site. The data in the *Scenario input file* include daily water level (m), daily inflow of NH4-N from land (g N m-2 day-1), daily inflow of NO3-N from land (g N m-2), and daily valur of hydrologic flushing rate (day-1), described in more detail below. (Note that the daily inflows of N

*Water level*: As there are no model processes that affect water level, it is simply read in from this file on a daily basis. Water level is measured (in meters) from the top of the MSOM pool. In other words, water level even with the top of MSOM is zero, positive values are above this, and negative values are below (Fig. II-2). Refer to the *water level* section for more details.

*Nitrogen inflow from land*: In the *Scenario input file*,nitrogen inflows from land represent combined river, overland, and groundwater inflows of NH4-N and NO3-N but they exclude precipitation fluxes (precipitation fluxes are given in the *Parameter input file*). The fact that these N inflows are listed on a daily basis over the entire model run provides total flexibility for the user. The user can choose to use constant N inflows, equal or differing ratios of NH4-N and NO3-N inflows, seasonal fluctuations, or N inflows that rise over time or fall over time throughout the model run.

To allow the user to systematically vary the absolute levels of N inflow while retaining complex seasonal patterns, the user can adjust the magnitude of these inputs across multiple simulations using ***NH4NinputLandMult*** and ***NO3NinputLandMult*** in the *Parameter input file* in concert with the *Batch input file*. See the *N inflow from land* section for more details.

**Note:** Nitrogen inflows in the scenario file are expressed as g N m-2 day-1 to allow for daily fluctuation. The development team often uses values of the overland N inflows of 0.00274 g N m-2 day-1 for each of the N species, constant throughout the year. Added up over the year, this produces N inflow of 1 g NH4-N m-2 y-1 ***and*** 1 g NO3-N m-2 y-1, totaling 2 g N m-2 y-1. With these values, the ***NH4NinputLandMult*** and ***NO3NinputLandMult*** in the *Parameter input file* will equal the total annual value of each of the N inflows from land. For example, for a value of ***NH4NinputLandMult*** = 4 in the *Parameter input file*, the total annual value of NH4-N inflow would then be 4 g N m-2 y-1. When considering total N inflow from land, remember, this is the sum of both NH4-N and NO3-N inflows.

*FlushingProp*: This parameter determines the proportion of inorganic nitrogen stocks that are removed from the regional N pool every day via hydraulic flushing. This allows the user to control hydraulic residence time (i.e. the inverse of flushing proportion) on daily time steps to explore seasonal and annual fluctuations, such as low water flow in summer and rapid water flow in spring. Since Mondrian does not directly model water flow, this input can allow users to approximate the effects of high or low flow conditions on N flushing from the wetland.

To allow the user to systematically vary the absolute levels of hydraulic flushing while retaining complex seasonal patterns, the user can adjust the magnitude of these inputs across multiple simulations using the parameter ***FlushPropMult*** in the *Parameter input file*, in concert with the *Batch input file*. The value that Mondrian uses internally for ***FlushProp*** on a daily basis is the value read in from the *Scenario input file* for that day, multiplied by the value of ***FlushPropMult*** from the *Parameter input file* or its override, in batch sets of model runs, in the *Batch input file*.

## The *Batch* input file

When the user does a single model run (including a single model run with stochastic reps, but no other changes), the *Batch input file* is not used. The *Batch input file* is only used when the user wishes to design a methodical set of model runs with varying input parameters. The set might range from three or four model runs to hundreds or even thousands.

For example, using the *Batch input file*, a user could do a set of 10 model runs where everything is identical except the rate of N inflow, which could be given 10 different values. The *Batch input file* would list the specific parameter in the *Parameter input file* that the user wants to vary, and the 10 values to give the parameter, one per row. For another example, the user could do a set of 30 model runs where the rate of N inflow has 10 different values, and the length of the growing season has three different values, in a set of 10 x 3 = 30 combinations.

Any parameter in the *Parameter input file* can be varied systematically using the *Batch input file*. Up to 15 different can be varied within one batch set of model runs.

The first two lines of the *Batch input file* are user comments. These can be anything the user wishes to say about the input file – the name or purpose of the set of model runs, for example. (Note: do not include any comma in these comments because Mondrian will mistakenly interpret a comma as the end of the comments and try to read what follows it as the first piece of data).

### File setup template and data check

Distributed with Mondrian input files is an Excel file that serves as a *Batch input file* setup template. The Excel setup file has detailed directions on how to organize the data for the *Batch input file* there and then copy and paste the information into a text file (ASCII text); Mondrian reads the text input file.

It is important to follow the directions in the Excel setup template closely so that the text input file is formatted correctly. Here we emphasize one particular formatting issue: if fewer than 15 parameters are changed in any line, make sure that placeholders of ‘0’ are used as placeholders for the remainder of the 15 slots.

To help in checking that all of the data from the *Batch input file* were read in and stored correctly, Mondrian produces an output file for this purpose. Mondrian’s stores the information in an internal array, and this output file prints from that internal storage array. (Normally this output file is not needed; it was created during model development and left in so that, if desired, the development team and/or the model user could double-check that large Batch input files were read and stored correctly.)

## Model testing, verification, and validation

People use the terms verification and validation in a variety of ways. According to Winsberg (2010), a philosopher of science who considers simulation models, there is overlap between the two ideas but for practical purposes there is still a useful distinction for modelers. We use the definitions of Winsberg (2010). Verification refers to tests that the software implementation is functioning the way it was intended, i.e. that there are no coding errors or mathematical errors. Validation assesses whether the right model formulation was used in a broader sense. In other words, whether the right model structure was used to represent the theory that the modeler intended to simulate. Other writers use validation to refer to tests of model results against empirical data. When this latter sense is used, it is usually considered important that the validation test data be independent from any data used to calibrate the model.

Describe some validation tests that you can do. The user decides this.

Does the model produce the reference mode? This is a type of validation in the sense of Winsberg (2010).

## Executing a set of model runs

### Input and output file names and locations

If the user does not change the input and output folders and files when prompted at run-time, Mondrian will read the default input file names from their default locations and write the output files to default names and location. A subsequent model run would read the same input files and over-write the default output files.

If you are doing a small set of 5 to 10 model runs, you could do each run individually, making a simple change in the Parameter or Scenario input file (re-saving them with the default names each time), and re-naming the output file after each model run so it does not get written over.

### Strategy and conventions for a large set of model runs

For a larger set of model runs, it is important to develop a strategy for (a) indexing all of the factorial factors and (b) naming conventions for input and output files. We typically split up a large set into discrete subsets. The factor that we typically use to break a large set into subsets is each particular combination of plant species (we have run anywhere from 3 to 11 different combinations of species in a consistent set of model runs). *This is key because the Batch and Seed input files are designed to allow a large set of runs to be broken up this way*.

**Example set of 540 runs**. This builds on the example set of 540 runs outlined earlier (see Designing a set). In this example, there are 5 different plant communities in a factorial design. Using the Batch input file, it is possible to do a super-run where you set up the input files and run Mondrian so that it repeats with different levels of N inflow, or any of the ‘switches’ (model structural options) on or off, or different plant traits. But different plant communities can not be done this way. So we would split this set of 540 into 5 subsets (one for each plant community or “species combination”). Each of the 5 subsets would be conducted with one particular plant community and 108 model runs (i.e. 36 distinct runs x 3 stochastic reps). Specifics about how to execute this set of model runs is given below:

* One Parameter input file could be used across the entire set of 540. It would provide default values of all parameters and model structural options (‘switches’), some of which would be varied in the Batch file.
* For the Seed input file, 5 different versions would be needed to bring about the 5 different plant communities (or “species combinations”). Before starting any model runs, the researcher should create these 5 Seed input files and save them with 5 different names: e.g. Mondrian-Seed-v40-C1.txt, then …-C2.txt, …-C3.txt, …-C4.txt, and …-C5.txt, where C1 through C5 stands for community type 1 through 5. Inside each of these different Seed input files, the value of the variable ***sppcomb*** (species combination) would be 1, 2, 3, 4, or 5. This variable ***sppcomb*** is read by Mondrian and printed on the results printout, to help the investigator track this information.
* One Scenario input file would be used across the entire set of 540, because in this case the researcher is keeping the seasonal water level dynamics and the temporal patterns of N inflows (seasonal or trending upward or downward) the same for all model runs. The total annual N inflow will vary in this set, but that can be varied in the Batch input file through the parameter ***NinflowMult***. (In a different situation, several different Scenario input files might be needed. For example, if the researcher wanted to explore constant N inflows throughout the year versus seasonality in N inflows, this would require two different Scenario input files. In that situation, she should create the two versions before beginning the set of model runs and save them with different names.)
* One Batch input file would be used to describe the 36 factorial variations in parameters (36 x 3 stochastic reps = 108). This identical Batch input file would be used five times, once with each of the five different Seed input files. Each use of the Batch input file would produce a single Output file containing 108 model runs for the ecology and biogeochemistry results (MondrianResultsBGC.txt). Within that output file, there are a few columns that would allow the researcher to distinguish the model runs. First, the variable ***sppcomb***, read from the Seed input file, is printed in a column near the far right. Second, following that, the column ***iinsubset*** would vary from 1 to 36. As a double-check, the researcher should keep a separate design-list of these 36 values of ***iinsubset*** and what parameter combinations they correspond to; this design-list should be used to make the Batch input file. Finally, in the first column of the output file, within each unique value of ***iinsubset*** there would be 3 model runs with different values of ***i*** from 1 to 3 (stochastic replication), because in this case the user chose 3 stochastic reps.

A few final words will complete this example of how to execute a large set of 540 model runs that have been broken into 5 subsets of 108. Each time the Batch input file was run, it executed a single subset of 108. When each of these Batch runs is complete, it is important to re-name the output file so it will not be overwritten, and to give it a filename that indicates what is unique about it as a subset – in this case, one of 5 plant communities. For that reason C1, C2, …, C5 could be used in the filenames.

We have found that the best approach is to make a careful plan for the large set of runs and to make up and organize all of the input files in advance. That way they can all be checked and double-checked for names and consistency before the first model run is begun. Then executing the large set is simply a matter of following a simple set of steps and selecting the right set of input files, minimizing the chances for mistakes or confusion. Different subsets can even be run simultaneously on different computers to speed the completion of the set. Output files should be carefully re-named and saved when each subset is completed. When the set is finished, all of the input and output files can be archived beause they provide an unambiguous record of what was done.

See the “Post-processing and analysis” section below for tips on how we have approached the issue of post-processing and analyzing the large amount of data that results from a large set of model runs.

## The main output file

Mondrian produces a main output file *MondrianResultsBGC.txt* that includes a large number of model results in a single file. Typically this is the only output file that is needed. Here we describe details about each element in this output file.

The output file begins with a **header** that gives a date and time stamp, and reproduces the top two comment lines from each input file: the Seed input file, the Parameter input file, the batch input file (when applicable; see Workflow / Batch input file), and the Scenario input file. Use these two lines of comments in each input file to record explanations about the input file – having these comments printed on the output file allows you to keep a permanent record of explanations of the specifics of this model run.

After the header, the remaining output is organized in colums and rows. Columns represent output variables and other results. Rows represent simulation years, except for the first 4 rows which are column headings. Each year, output is printed at the end of the day on the last day of the year (i.e. December 31 of the simulation year); many variables represent annual sums or snapshots of model pool on that date. (Some selected variables are instead saved in the middle of the growing season, to be printed out at the end of the year).

Here we explain the output in each column in order, left to right:

**i:** If a number of replicate stochastic model runs is selected (where all parameter values and scenario inputs are the same, but all stochastic processes differ – See Workflow / Stochastic replicate runs), this is an integer representing the number of the stochastic replicate run.

**yr:** The simulation year.

**density spp1** [n / m2]: density of living ramets for species 1, in numer of individuals per m2.

(the same information is then given for spp2, spp3, spp4, spp5).

**ABNPP spp1** [g C/m2 y]: Aboveground + belowground NPP (net primary productivity) for species 1, given in g C / m2 y.

(The same information is then printed for spp2, spp3, spp4, spp5).

**ANPP spp1** [g C/m2 y]: Aboveground NPP (net primary productivity) for species 1, given in g C / m2 y.

(The same information is then printed for spp2, spp3, spp4, spp5).

**spp1 C sum** [g C]: The sum of all tissue C pools across all individuals in species 1, across the entire model space. Note this is not normalized per unit area. This is used mainly in model testing.

**spp1 N sum** [g N]: The sum of all tissue N pools across all individuals in species 1, across the entire model space. Note this is not normalized per unit area. This is used mainly in model testing.

**spp1 PBGCNN** [avg]: The average over all living individuals of species 1 of the value of the individual-based variable ***PBGCNN***, an abbreviation for ‘proportion of belowground C and N needed’. This variable is used in an algorithm that allows massive litter layers to suppress vegetative reproduction. (*See* Ecological Processes / Production of a new stem.)

**spp1 RGRLM** [avg]: The average over all living individuals of species 1 of the value of the individual-based variable ***RGRLgtMod***. This is an index that varies from 0 to 1 and that represents the scaling-back of ***RGRmax*** (maximum relative growth rate) based on shading, i.e. light competition. By looking at the value of this variable in the printout, the user can see the degree to which light competition is scaling back the growth in this species, on average, in a particular year of a particular model run. ***RGRLgtMod*** has a value of 1 if there is no shading, and a progressively smaller value as shading of the individual becomes increases. NOTE: The printed value of this variable is not its value on December 31; its value on a user-specified day is instead saved and printed. The day is determined by the parameter ***LgtCompPrintDay,*** set by the user in the Parameter file.

(The same set of variables, C sum, N sum, PBGCNN, and RGRLM, are then printed for species 2, 3, 4, and 5.)

**PIP C sum** [g C/m2]: Plant internal pools, C sum, given in g C / m2. This is a sum of all plant C pools across all individuals of all species, normalized to the total modeled area. This is used for model testing.

**PIP N sum** [g N/m2]: Plant internal pools, N sum, given in g N / m2. This is a sum of all plant N pools across all individuals of all species, normalized to the total modeled area. This is used for model testing.

**GCP AGlit C** [gC / m2]: This is the first of a series of output columns for “Grid Cell Pools” (GCP), reporting quantities that vary from one grid cell to another, averaged here across the entire model space. This pool is aboveground litter C, given in g C / m2.

**GCP AGlit N** [gN / m2]: Grid Cell Pools, aboveground litter N, given in g N / m2.

**GCP AGlit t** [m]: Grid Cell Pools, aboveground litter layer thickness, given in meters.

**GCP BGlit C** [gC / m2]: Grid Cell Pools, belowground litter C, given in g C / m2.

**GCP BGlit N** [gN / m2]: Grid Cell Pools, belowground litter N, given in g N / m2.

**GCP BGlit t** [m]: Grid Cell Pools, belowground litter layer thickness, given in meters.

**GCP MUCK C** [gC / m2]: Grid Cell Pools, MUCK C, given in g C / m2.

**GCP MUCK N** [gN / m2]: Grid Cell Pools, MUCK N, given in g N / m2.

**GCP MUCK t** [m]: Grid Cell Pools, MUCK thickness, given in meters.

**GCP MSOM C** [gC / m2]: Grid Cell Pools, mineral soil organic matter C, given in g C / m2.

**GCP MSOM N** [gN / m2]: Grid Cell Pools, mineral soil organic matter N, given in g N / m2.

**GCP IN sum** [gN / m2]: Grid Cell Pools, inorganic (available) N, given in g N / m2.

**RNP IN** [gN / m2]: Regional Nutrient Pool, inorganic (available) N, given in g N / m2.

**Ecos C tot** [gC / m2]: Ecosystem C total, given in g C / m2. This is a total of all C pools in the entire ecosystem, including all plant pools and all detrital pools of C, normalized per unit area across the entire modeled space.

**N influx** [gN / m2y]: This and the following several variables provide a complete ecosystem N budget. This term is ecosystem N influx, meaning the summed annual inflow of N to the ecosystem from all sources, normalized per unit area over the entire model space, given in g N / m2 y.

**N hydexp** [gN / m2y]: Hydrologic N export, meaning the summed annual hydrologic outflows of N from the ecosystem, averaged over all grid cells and normalized per unit area, given in g N / m2 y.

**N detexp** [gN / m2y]: Detrital N export, meaning the summed annual export of N in detritus from the ecosystem, averaged over all grid cells and normalized per unit area, given in g N / m2 y. Conceptually this represents litter and litter mats that are flushed out of the ecosystem, mainly in storms.

**N min, lit** [gN / m2y]: N mineralization from litter, averaged over all grid cells and normalized per unit area, given in g N / m2 y.

**N minMUCK** [gN / m2y]: N mineralization from MUCK, averaged over all grid cells and normalized per unit area, given in g N / m2 y.

**N minMSOM** [gN / m2y]: N mineralization from mineral soil organic matter, averaged over all grid cells and normalized per unit area, given in g N / m2 y.

**N trans** [gN / m2y]: The net annual transfer of inorganic N between the grid cell pools of available inorganic N and the regional inorganic N pool, averaged over all grid cells and normalized per unit area, given in g N / m2 y. A positive value represents a net transfer of N from the grid cell pools into the regional N pool; a negative value represents a net transfer in the opposite direction.

**N uptake** [gN / m2y]: N uptake by plants, from available inorganic N pools in grid cells, averaged over all grid cells and normalized per unit area, given in g N / m2 y.

**N AGlitf** [gN / m2y]: N flux in aboveground litterfall, summed across all plants of all species, averaged over all grid cells and normalized per unit area, given in g N / m2 y. This quantifies the movement of N from standing dead litter to aboveground litter pools. (It does not include the transfer of living aboveground tissue to standing dead litter pools, but in Mondrian where no standing dead litter remains for more than one year, on an annual basis that is equal to the litterfall flux).

**N BGlitf** [gN / m2y]: N flux in belowground litterfall, summed across all plants of all species, averaged over all grid cells and normalized per unit area, given in g N / m2 y. This quantifies the movement of N from living, belowground tissues to belowground litter pools that occurs with belowground mortality.

**N reten, l** [unitless]: Ecosystem N retention from land. This is calculated as follows:

N retention from land = 1 – [(N in hydrologic export + N in detrital export) / (N influx from land)]

**N reten** [unitless]: Ecosystem N retention. This is calculated as follows:

N retention = 1 – [(N in hydrologic export + N in detrital export) / (N influx from all sources)]

**C ABNPP** [gC/m2y]: This and the following several variables provide a complete ecosystem C budget. This term represents C flux in aboveground + belowground NPP (net primary productivity), summed over all species and all grid cells, normalized per unit area across the entire model space, given in g C / m2 y.

**C NEP** [gC/m2y]: Net Ecosystem Productivity, summed over all species and all grid cells, normalized per unit area across the entire model space, given in g C / m2 y.

**C HetResp** [gC/m2y]: Heterotrophic respiration, summed over all detrital pools and all grid cells, normalized per unit area across the entire model space, given in g C / m2 y.

**C AGgrow** [gC/m2y]: Aboveground plant growth, summed over all species, averaged over all grid cells and normalized per unit area, given in g C / m2 y.

**C BGalloc** [gC/m2y]: Belowground plant C allocation, summed over all species, averaged over all grid cells and normalized per unit area, given in g C / m2 y.

**C AGlitf** [gC/m2y]: C flux in aboveground litterfall, summed across all plants of all species, averaged over all grid cells and normalized per unit area, given in g C / m2 y. This quantifies the movement of C from standing dead litter to aboveground litter pools. (It does not include the transfer of living aboveground tissue to standing dead litter pools, but in Mondrian where no standing dead litter remains for more than one year, on an annual basis that is equal to the litterfall flux).

**C detexp** [gC/m2y]: Detrital C export, meaning the summed annual export of C in detritus from the ecosystem, given in g C / m2 y. Conceptually this represents litter and litter mats that are flushed out of the ecosystem, mainly in storms.

**C to MSOM** [gC/m2y]: The transfer flux of C from aboveground and belowground litter pools to mineral soil organic matter pools in each grid cell, averaged over all grid cells and normalized per unit area, given in g C / m2 y.

**C to MUCK** [gC/m2y]: The transfer flux of C from aboveground and belowground litter pools to MUCK pools in each grid cell, averaged over all grid cells and normalized per unit area, given in g C / m2 y.

**C Bal** [g C]: C mass-balance check (*See* Automated error checking). The value of this number does not matter, but it should be a constant value throughout the model run.

**N Bal** [g N]: N mass-balance check (*See* Automated error checking). The value of this number does not matter, but it should be a constant value throughout the model run.

**ErrCt**: Logged error count. Should be zero in every year of every model run. (*See* Automated error checking).

**ErrLast**: Number (code) of the last logged error. Should be zero in every year of every model run. (*See* Automated error checking).

**sppcomb**: Species combination. This is an integer representing the particular plant community of combination of species used in this model run, within a large set of model runs. This is assigned by the user to organize large sets of model runs; this number is simply read from the Seed input file and printed here on the main printout. (*See* Exectuting a set of model runs).

**iinsubset**: Read this as i-in-subset. Within a larger set of model runs, based on the organized subset of runs as defined in the Batch input file, this number will be incremented as outlined in the Batch input file, and printed here in the main output file to allow the investigator to track which model run this is within the organized subset. (*See* Exectuting a set of model runs)

**The last 15 columns**: The last 15 columns in the main output file are used to allow the investigator, in an organized set of model runs that uses the Batch input file, to track the values of parameters that are changed from their values in the Parameter input file, according to instructions in the Batch input file. If the Batch input file is not used (i.e. this is a single model run, with or without stochastic reps), the column headings will read “n/a” and the values in the columns will all be zeroes. If the Batch input file is used, column headings here represent the parameter names supplied by the user in the Batch input file, and the values of each parameter represent its value used in the present model run within the set.

## Additional output files

Mondrian has the ability to print additional output files, which can be turned on and off by the user. Typically these additional files are not needed. They were used during model development to make sure certain detailed processes were working as expected. However, they still exist, and could be useful in certain circumstances.

To turn on the printing of these additional output files, check the box on the Main User Interface form (the dialogue window where the input files are selected) that reads “Include additional output files for detailed model testing.) Note that these additional output files can only be used in a single model run (not in a large set of model runs).

### Parameter values printout file

This file simply prints out all of the parameters from the Parameter input file, after they have been read in and used to set the values of appropriate variables in the code. The purpose of this file is simply to check that all of the parameters were read in and stored correctly.

### Logical Flow Analysis output file

Mondrian includes a limited ability to conduct what we call a “logical flow analysis” (LFA). In doing the LFA, the model uses counters that increment each time certain sets of logical conditions are met, causing certain aspects of logical flow in the model. These focus on aspects of plant physiology, phenology, and competition in the model. This helps the development team and it can help the users to understand how the model is working by comparing, from one model run to another, how often certain conditions are occurring.

If the printing of additional output files is selected, the results of the LFA are printed in the file *MondrianResultsLFASpp.txt*.

Below are descriptions of the conditions that are counted. (These descriptions are also printed in the header of the output file.) Each of these are counted by species and printed each year by species, both as an absolute number of occurrences and as the number of occurrences per living individual of that species, on average. The “per individial” quantity is calculated simply by dividing the number of absolute occurrences in a species, over the entire year, by the number of living rhizome nodes of that species on December 31 of that year.

|  |  |
| --- | --- |
| **Counter** | **Condition** |
| 1 | Uptake N demand was scaled back due to limitation of available inorganic N in grid cell |
| 2 | There is enough rhizome C, but not enough rhizome N, to establish an aboveground stem without parental subsidy |
| 3 | There is enough rhizome N, but not enough rhizome C, to est an aboveground stem without parental subsidy |
| 4 | There is both enough rhizome C and N at same time to establish an aboveground stem, with or without parental subsidy. (i.e. total number of new aboveground stems created.) |
| 5 | Aboveground growth limited because not enough N is in belowground part to send up |
| 6 | Number of successful daughter rhizomes (belowground) created |
| 7 | Number of individual-days that a living aboveground stem was completely submerged |
| 8 | Number of times an individual living stem was killed because it was submerged too long |
| 9 | Number of times a first new daughter rhizome node is created by a terminal node |
| 10 | Number of times a terminal branch & node are created by a terminal node |
| 11 | Number of times a lateral branch & node are created by a non-terminal node |

## Post-processing and analyzing results

Mondrian produces output of results as csv (comma separated value) files.[[14]](#footnote-14) These can be read by any ASCII text editor or imported into other software (see below). There is one main output file that contains the ecology and biogeochemistry results as well as model self-test diagnostics and a header. The header contains information about the model run(s), including the user comments from each of the input files. In this way, the comments on the input files could be used to provide double-checks that provide information about parameter sets, drivers, scenario files, and so on. The main output file also contains column heading abbreviations for each result, together with an explanation (including units and clear definitions) of each column abbreviation [[15]](#footnote-15). Where multiple model runs are combined in one output file (organized in the batch input file), the header is printed only once at the top of each output file. The integer value in the ‘iinset’ column (short for “i” or index in the set) distinguishes one model run from another.

When a large set of model runs (hundreds to thousands) is executed, it typically includes multiple sets of output files. the next step in the workflow is to build a database that contains results from all of the model runs, together with information needed to separate and identify each individual model run.

Microsoft Excel is a convenient tool to use as an intermediate step to prepare the model results for statistical analysis and graphing of results. If the output text file will be imported into Excel, the header can be removed beforehand using a text editor (e.g. Microsoft Notepad). (We recommend also saving the original output files because the headers contain information that may be useful for double-checking model runs later if needed.)

Excel can be used to organize the results from hundreds or thousands of model runs in a consistent manner. Once organized in Excel, the results from many combined model runs can be exported as a dataset to Stata or R for statistical analysis. Alternatively, the results files in csv format can be read directly into R or Stata and merged there to create a database holding a large set of model runs.

Where multiple stochastic replicates of identical model runs were performed, the mean and variance (or standard error or standard deviation) for each model result, in each simulated year, need to be calculated within each small set of stochastic replicate runs. This can be done systematically in Stata or R.

Graphing of model results can be accomplished in a number of ways. Results can be exported from Excel, Stata, or R into SigmaPlot for graphing. Graphing can also be done directly in Stata or R.

# VII. Error checks and known glitches

## Automated error checking

### Logged errors with numbered error codes

Mondrian code includes a large number and variety of error checks. In the vast majority of model runs, and if the model is working correctly, the number of logged errors is zero (*See* Workflow / Main output file). If a model run generates any logged errors, something went wrong and that model run should not be used.

The logged error counting and error-code printing are used mainly by the development team, to ensure that the model is still working correctly after complex code changes are made. One category of logged errors includes errors in the logical operation of the ecological processes. For example, an aboveground stem is logically not allowed to live if its rhizome node has died; thus in the code where dead belowground tissues are added to belowground litter pools, error-checking code checks to see if there is a living stem at that rhizome node. In the design of the model, that should not happen; nevertheless Mondrian error-checking code checks for a variety of these types of errors and if one is found, an error is logged. Another category of errors includes checking for mathematical problems before they occur, for example if the model attempts to divide one variable by another when the second variable is zero. Mondrian code tries to check and avoid such an error so the program will not simply halt or hang; when these types of errors are found, the model skips past the bad calculation, and an error is logged and printed out.

When multiple errors are logged, the total number of errors is summed for each year, within each model run, and this is printed out in the main results output file. Within each year, next to the count of the total number of errors logged, Mondrian prints the number of the last error that was logged. (Mondrian has a different error identification number for each type of error check.) Normally only one type of error occurs, but if there are more than one type, the development team identifies and fixes them one by one.

*If you are a model user who has only the .exe (executable) version* of Mondrian and there are errors logged in your results printout, do not use that model run. If the error continues, contact the development team. (An exception is that if you are doing a large set of hundreds or thousands of model runs where a large number of parameters are being varied, or scenarios are being varied, or both. We have seen that in such a large set of runs, there are sometimes a few runs [much fewer than 1% of the model runs] that log errors. We believe those occur from unpredictable, random processes that happen to produce some variable values too close to zero. In those cases, simply discard those individual model runs, keep the rest, and re-run the individual model runs that had problems if needed to complete the full set.)

*If you have the VB source code for Mondrian and wish to track down a logged error*, simply search within the source code for “ErrLast = xx”, where xx is the number of the last logged error. This search will take you to all of the places in the code where that error could be logged.

### Mass balance checks

Because Mondrian has so many processes that move small amounts of C and N from one pool to another, it is important to check that no systematic bias enters into any calculation that could cause more C or N to be credited to a target pool than debited from a source pool during a transfer, or vice versa. It is important that mass balance be preserved. In other words, all of the C calculations have to add up properly and all of the N calculations have to add up properly.

Mondrian performs mass-balance checks for C and N at the end of each simulated year and prints these in the main results printout (*See* Workflow / Main results printout). Note that for the C column, the specific number printed does not matter. What matters is that the same number is printed each year; this shows that mass-balance is preserved. The same is true of the N column: what matters is that the same number is printed each year.

## Known glitches

This is a list of known model glitches and bugs. From time to time we work to track these down and fix them. If you find any other glitches, please email a description to Bill Currie, [wcurrie@umich.edu](mailto:wcurrie@umich.edu).

Touch screen. Mondrian was developed on computers without a touch screen; if it is being run on a computer with a touch screen and someone touches the screen while it is running, we have seen it simply stop (i.e. hang) in the middle of a model run. In other cases, it may appear to have stopped, but it is actually still running in the background and just not updating the screen.

Using the ‘x’ at the top right corner of a form, instead of the ‘close and exit’ button on the form. When a form is showing on the screen and you wish to stop the model run and exit, click the button that says ‘close and exit.’ Do not click the ‘x’ at the top right corner of the form – what that does is close the form but it does not tell Mondrian to halt the model run in a clean way. The problem that arises is that Mondrian never closes the input and output files. The next time you try to do a model run you may get an error message saying the input files are already in use. Unfortunately, the only resolution may be to re-boot the computer. We are working to fix this glitch in a future release.

Format of ASCII input files. Make sure that the format of each ASCII text input file matches the default (example) input files exactly. The first two lines in each in input file must be comments and must include no commas. Mondrian saves a copy of each comment line as a string; a comma would be interpreted as the end of the string variable. Note the placement of spaces and commas in other input lines in the example input files – this format must be reproduced exactly. The columns do not all have to line up, but the commas (or absence of commas), spaces, and carraige returns (at the end of each line) need to be present.

The end of each ASCII text input file must have exactly one carraige return, with nothing else included on that line. We have seen Mondrian fail to read the input file correctly if there is no carraige return at the end, or if there are two blank lines with carraige returns at the end of the file.

Overflow / underflow errors. Mondrian sometimes produces an overflow / underflow error where one or more model results can not be printed in the space available (sometimes written as “nan”, which is short for ‘not a number’, in the output where a number should appear). This occurs because the number is either too large or too small (too many zeroes after the decimal point before the first non-zero digit has the same effect that a too-large number has in this case, which is requiring too much memory to fit into the bin assigned to it in computer memory). We have also seen this occur infrequently with a logged error (number 39), when N inflow is small. We believe it arises from repeated calculations that involve very small values of N, producing a round-off error that propagates through the code; we are working to track this down and fix it. For now, when this happens, the model run should be discarded.

Need all light-extinction coefficient C for all 5 species. At present, Mondrian can simulate up to 5 species. The submodel for light extinction has a polynomial function for light extinction, with three coefficients (A, B, C) for each species. In the parameter input file, be sure to include a nonzero value for C for all 5 species, even if you are not including all 5 species in the model run. (In fact, the value for C should probably be 100 or greater for all 5 species, whether or not they are included in the model run). There is a subtelty in the programming of the light extinction code that causes too much light extinction to occur otherwise. We are working to fix this glitch in a future release.

# VIII. Best Practices for Archiving Runs and Continued Development

## Archiving input and output data from model runs

There are many reasons why it is important to save model input and output data in a consistent and unambiguous way:

* You may do a model test using a standard set of parameters, and wish to repeat that test exactly, after major changes have been made to Mondrian;
* You may apply the model across a set of field sites or scenarios and wish to record the differences in parameters and scenario files used;
* You may do a set of 5,000 model runs for a planned publication and wish to save a clear record of all of the input and raw output files before doing any post-processing for data analysis

In each of these cases what is needed is a mechanism for archiving the input data used in a model run or set of runs, archiving the output data files from the model runs, and identifying all of those files unambiguously with the particular model run or set of runs.

Described below are the conventions used by the Currie lab to achieve these goals. Each research lab using Mondrian should either adopt or adapt these conventions to their specific purposes.

### Naming and saving: Exercises

Individual model runs or small sets of runs can be named as an “Exercise” with the date, e.g. *Ex 2014 0720 A*. The date supplies a convenient, unique identifier for an Exercise. Subsequent exercises on the same date can be labelled B, C, etc. If there is concern that two people may be saving model Exercises on the same date, then the person’s initials can be added to the Exercise name.

In your model-runs archive, create a folder with the Exercise name (e.g. folder name: Ex 2014 0720A), and save the input and output files as .txt files in this folder.

In the Currie lab, we save a separate “log file” as a Word doc that describes the purpose of each dated / numbered Exercise, plus sometimes initial interpretations of the run results. It is easy to pair up the Exercise name (e.g. Ex 2014 0720 A) in the Word doc with input and output files stored in the folder of that name.

### Naming and saving: Sets

A large set of runs (hereafter “Set”) can be given a better type of name so that it can be referred to and archived more easily. For example, the first large factorial set of 1000+ model runs we did with Mondrian was named ***Sitran0808***, short for “Size Trait Analysis 08 / 2008”. The next large set of 2000+ runs was named ***Comp12***, for “Native and invasive competition 2012.” As these large sets of runs accumulate, a research group may want to create a single index file containing these abbreviations and brief information about these Sets.

For saving the input and output files, in your model-runs archive create a folder with the Set name and use a folder hierarchy beneath that to save as many files as needed. For example, you may have 10 different parameter input files or 10 different scenario input files that were all part of the set, together with a dozen or more output files.

### Set Hierarchy of files that should be archived

For a Set whose results will be published, all of the following should be permanently archived. These should be stored in a folder hierarchy where the top level has the date and the name of the Set.

**Sources for model parameter values**. A file describing where the values for parameters came from, including citations from the literature, should be archived. Using the filename and comments within, anyone should be able to unambiguously link the sources of the parameter values to the specific Set of model runs. Excel can be used for this.

**All text input files used** in the model runs in the Set. This includes all parameter files, scenario files, seeding files, and batch files. If the Set was broken into chunks for execution on multiple computers simultaneously, the best practice is to create separate input files in advance for each chunk, name these input files according to a clear system, and keep all of these input files. All of these files should be archived for each of the chunks. A key that explains how the chunks were executed and how the files were named should be part of the metadata included in the “planning file” described below.

**Metadata for the set of model runs**. This takes two forms. First, a “planning” file should be saved that provides a key explaining how the factorial set of model runs was set up. For example, for a set of runs that varies nutrient inflows, the planning file should provide the number of different nutrient inflows used and their values, the values and meanings of various flags and indices that were turned on or off for individual runs in the set, and so on. If the runs were performed in chunks (on multiple computers simultaneously for example), then this file should also explain how the chunks were set up for execution. MS Excel can be used for this.

Second, the testing, goals and choices made in leading up to a large set of runs should be described in a log file (MS Word). The log file can be saved in a chronological folder of model log files or saved in the folder hierarchy for the Set. Any other files associated with calibrations or parameter calculations and so on should be named in the log file and saved in the folder hierarchy of the Set.

**Model results**. All model results from the Set should be archived in at least one format that allows forseeable long-term access. This can be the full suite of original raw output files (as .txt), and/or an Excel file used to combine and process the raw output files, and/or a database in Stata or R. If a database is used for archiving the results, (i) ensure that it contains all of the model runs from the Set, and (ii) a separate metadata file should be saved that describes the database and software version. (A best practice would be to also save the full set of results as ASCII text files for ensured long-term access.)

### Additional files not necessarily part of the Set hierarchy

Graphics files (figures for papers) in preparation for a publication fall in this category. They need not be stored in the Set hierarchy. At the same time, graphics files for a publication should contain enough information that their source data can be unambiguously traced back. In SigmaPlot, graphs and their associated data can be stored in a ***section*** hierarchy, where ***notes pages*** within each section can be used to record free-form notes about where the data came from. Data analysis scripts (R, SAS, Stata, etc.) also fall under this category and should be well-commented so that other researchers can re-run or modify the analysis.

## Who is responsible?

When a publication, report, or conference presentation uses results from a Set of model runs, the ***lead author*** has the responsibility to ensure that all of the files, metadata, and results listed above are organized and archived. The lead author may delegate this responsibility to another member of the team, for example if someone else on the team carried out the model runs. But this transfer of responsibility should be explicit.

Once the lead author (or delegate) has constructed the folder hierarchy of files to archive, s/he should ensure that this is (a) backed up in more than one location, and (b) protected from unauthorized changes. Best practice: We expect that the lead author on a paper or presentation (or their delegate) has the responsibility to keep an archived copy of the Set folder hierarchy in perpetuity.

If a journal requires that the data underlying a publication be archived at the journal or in a community archive, the lead author (or delegate) will provide the model results together with metatdata as needed.

A funded grant may have a data management plan. In that case, the archiving described in the data management plan should be followed.

Best practice: If a journal or public archive takes the data, the lead author (or delegate) should also keep a copy of the archived folder hierarchy in perpetuity.

## Informal cheat-sheet to track multiple Exercises and Sets over time

In practice, having each lead author responsible for archiving their own Exercises and Sets is preferable to having one centralized repository where, for example, all parameter files and other files would be stored. The latter approach would be difficult to maintain as a formal structure because no one person is responsible.

That said, within a collaborative team, it may be useful to keep a collection of parameter files and scenario files for different sites or purposes in a common location. This should be viewed as a working directory, not the formal archive.

For the Currie-Martina-Elgersma-Goldberg collaboration, we have set up a Google drive folder where we place input files, results of model exercises, and planning files, so that everyone on the team can access them. The Exercise and Set conventions above should be followed for naming and should be used as folder names. An Excel file can serve as a cheat-sheet to identify, for each Exercise or Set, the model user, the site, the Mondrian version, and other information. This will allow us to quickly compare model parameter values used in different sets of model runs.

## Model versions, development, and testing

At present, Bill Currie keeps the definitive copy of each sequential version of Mondrian, together with examples of all of its needed data input files. Currie will archive old copies of all model versions and keep multiple backup copies of the current model version. Currie will continue to steer new development and testing for the present and will also keep a chronological model log that details dates, goals, model changes, and preliminary testing. More thorough testing may be carried out by others on the development team.

At present, Mondrian has several planned, major changes that have been funded, so it is very much still in a continuous cycle of planning, modifying, testing, and producing new applications. At the same time, we wish to share Mondrian with other teams of investigators that may be working in different regions or with very different research questions – the model was designed from the start to enable people to address a wide range of questions (*see* Potential research questions, below). We are also eager to see Mondrian used as a teaching tool about wetland ecology, plant ecology, biogeochemistry, invasive species, wetland management, and modeling itself. For these reasons we are happy to facilitate the use of the model by other teams.

*CAUTION! this is a very complex individual-based model. It would be difficult for someone who was not involved in model development to make self-consistent changes in the source code because many parts of the model depend on other parts in subtle but important ways.*

We recommend that other investigative teams who wish to use the model in new ways do so by changing the parameters and scenarios in the four input files, not by making changes to the source code. If any investigators do make changes to the model source code, we ask that you save either the VB solution file (\_\_\_.sln), or the main module file (MondrianMod1), or both, using a different filename so that it will never get confused with the definitive version maintained by our team.

Details on variable naming and coding conventions used in the Mondrian VB code are listed in Appendix III.

For citing the model and this user guide, *See* our Fair Use Guidelines.

# IX. Potential Research Questions to Explore

We designed Mondrian so that it could be used by different research groups to address a wide variety of research questions in different wetland systems. Hundreds of parameters that represent plant traits or environmental drivers can be changed by the user by making simple changes to parameters that are contained in input files.

Listed below is a short (but by no means complete) set of ideas for research questions that Mondrian could be used to explore with its current formulation. We also look forward to seeing what creative and innovative types of uses others may find for this model.

**Parental subsidies of C and N**. Rhizome nodes can receive parental subsidies from 1 to 5 parents back in the chain within a genet. This is set by the user in species-specific parameters in the Parameter input file. Mondrian could be used to explore how a set of hypothetical species that are identical in every way, except for this trait, compete against one another across a range of nutrient inflows. Mondrian could also be used to explore whether greater translocation in an attempted invader makes a species more invasive, or whether greater translocation in natives makes them better able to resist and attempted invader, and how this may interact with plant size and levels of nutrient inflows.

**Litter decomposition.** Litter that decomposes rapidly cycles N more rapidly in the ecosystem, which might differentially benefit more N-demanding species. At the same time, it could decrease wetland N retention and slow the buildup of N in litter and muck, making the plant community more sensitive to interannual variability in N inflows. Mondrian could be used to explore competition between plants with fast-decomposing or slow-decomposing litter under a range of nutrient inflows and/or water residence times (the latter is controlled by the user-set parameter ***RNPINexportProp*** in the Parameter input file). It could also be used to explore the interaction with water levels, which slow decomposition when the litter and muck are inundated and speed decomposition when water levels are lower.

**Wetland N retention**. Mondrian could be used to explore interactions among a variety of complex controls on wetland N retention, including rates of N inflow, plant traits such as maximum size and maximum relative growth rate, water residence time (controlled by the user-set parameter ***RNPINexportProp*** in the Parameter input file), export of detritus such as floating litter mats (controlled by the parameter ***DetExpProp***), rates of litter and muck decomposition, and fluctuations in water level (which strongly affect rates of litter and muck decomposition).

**Flooding mortality and re-colonization**. Mondrian includes mortality of individual plants based on the number of consecutive days each individual plant is completely submerged; the number of submerged days that cause mortality is a species-specific parameter set by the user, ***FloodMortDays1*** for species 1, ***FloodMortDays2*** for species 2, etc. Mondrian could be used to explore differential mortality among species from water level fluctuations having different frequencies, amplitudes, or timing in the growing season (all of which are set in the Scenario input file). It could also explore re-colonization by native and invasive plant species following mortality. For example, a large-size invasive plant may not compete successfully against an established native community at an intermediate rate of N inflow, but if the native community is knocked back from flooding mortality and then both native and invader propagules arrive, the invader may be more successful. Irrespective of invasion, a researcher could also explore how differential flooding mortality among species might affect plant competition and community dynamics during and after years with high water or over longer time periods with a variety of hydroperiod regimes.

**Carbon accretion**. Mondrian includes four levels of organization, including individual plant physiology, population-level processes, community-level process, and ecosystem-level processes including complete C and N budgets. It could be used to explore the complex interactions among plant traits, hydroperiod, N inflows, and other environmental factors that control C accretion (or loss) in litter, muck, and sediment organic matter pools.

**Scaling up**. Some key questions in wetland ecology address how ecological processes scale from a 1 m2 experimental plot, to a wetland site, to a variety or range of wetlands in a region. If we understand how environmental drivers combine to cause ecological or biogeochemical outcomes, then the scaling question is essentially a question about spatial and temporal heterogeneity in environmental drivers. If estimates of spatial and temporal heterogeneity in drivers can be made, then Mondrian could be used to explore the scaling up of ecological and biogeochemical processes and outcomes.

**Trends or seasonality in N inflows**. To date, Mondrian has been used mainly to explore attempted plant invasions, plant competition, C and N cycling, and wetland management practices only with constant (but differing) levels of N inflow. However, the Scenario input file allows the user to explore a rising trend in N inflow, a falling trend in N inflow, or different seasonalities of N inflow, as well as interactions of these with different lengths of the growing season and different scenarios of water level fluctuation. This suggests a wide range of scenarios that could be explored to simulate the effects on wetland N retention, C accretion or loss, and plant productivity and community dynamics.

**Exploration of space by clonal plants**. Mondrian allows the user to set the distance and angle from a parent that a daughter will be established. These are species-specific parameters. The model could be used to explore the competitive outcomes, or exploration of space, among species that are identical but differ only in these parameters.

# X. Acknowledgments

Model development was funded, in part, by the National Aeronautics and Space Administration, grant #NNX11AC72G. Modifications and additional model applications have been funded by the University of Michigan Water Center, the U.S. EPA under the Great Lakes Restoration Initiative, Invasive Species Control Program, EPA Grant # GL 00E01929-0; and the Michigan Department of Natural Resources Invasive Species Grants Program, Grant # IS15-5004.

# XI. References Cited

Currie, W. S., D. E. Goldberg, J. Martina, R. Wildova, E. Farrer, and K. J. Elgersma. 2014. Emergence of nutrient-cycling feedbacks related to plant size and invasion success in a wetland community-ecosystem model. Ecological Modelling 282:69-82.

Currie, W. S., M. E. Harmon, I. C. Burke, S. C. Hart, W. J. Parton, and W. L. Silver. 2010. Cross-biome transplants of plant litter show decomposition models extend to a broader climatic range but lose predictability at the decadal time scale. Global Change Biology 16:1744-1761.

Elgersma, Kenneth J., Jason P. Martina, Deborah E. Goldberg, and William S. Currie. 2017. Effectiveness of cattail (*Typha* spp.) management techniques depends on exogenous nitrogen inputs. Elementa, Science of the Anthropocene 5:19. DOI: <http://doi.org/10.1525/elementa.147>

Elgersma, K. J., R. Wildová, J. P. Martina, W. S. Currie, and D. E. Goldberg. 2015. Does clonal resource translocation relate to invasiveness of Typha taxa? Results from a common garden experiment. Aquatic Botany 126:48-53.

Goldberg, Deborah E., Jason P. Martina, Kenneth J. Elgersma, and William S. Currie. 2017. Plant size and competitive dynamics along nutrient gradients. American Naturalist 190(2): 229-243.

Grimm, V., T. Wyszomirski, D. Aikman, and J. Uchmanski. 1999. Individual-based modelling and ecological theory: synthesis of a workshop. Ecological Modelling 115:275-282.

Huston, M., D. DeAngelis, and W. Post. 1988. New computer models unify ecological theory. Bioscience 38:682-691.

Loreau, M. 1998. Biodiversity and ecosystem functioning: A mechanistic model. PNAS 95:5632-5636.

Martina, Jason P., William S. Currie, Deborah E. Goldberg and Kenneth J. Elgersma. 2016. Nitrogen loading leads to increased carbon accretion in both invaded and uninvaded coastal wetlands. Ecosphere 7(9): e01459.

Miller, J. H., and S. E. Page. 2007. Complex Adaptive Systems. An Introduction to Computational Models of Social Life. Princeton University Press, Princeton, NJ. 263 pp.

Neff, J. C., E. A. Holland, F. J. Dentener, W. H. McDowell, and K. M. Russel. 2002. The origin, composition and rates of organic nitrogen deposition: a missing piece of the nitrogen cycle? Biogeochemistry 57:99-136.

Schwinning, S., and J. Weiner. 1998. Mechanisms Determining the Degree of Size Asymmetry in Competition among Plants. Oecologia 113:447-455.

# Appendix I. Critical Checklist for Students or New Users

This is meant to be a brief list of key tasks and checks for both major and minor things that students need to be aware of in order to have Mondrian run successfully.

1. Before doing anything else, make a backup, unchanged copy of the entire Mondrian file folder including the model code an ALL input files. This will allow you to always go back and check what the default files look like and what the default input parameter values were.
2. Make sure that the file folder structure is set up correctly for both input and output files and that all of the input files are in the correct location and named correctly (*see* Obtaining and installing Mondrian).
3. For all four of the ASCII text input files, check that they have the exact same formatting as the default file examples. Make sure that the first two lines in the file are text comments with no commas. Make sure that inside each file, the last line of the file is a single carraige return and blank line (including no spaces or anything else). (*See* Workflow / input files).
4. A good way to begin using Mondrian is to do comparative model runs where you keep everything the same from one run to the next except one parameter in the Parameter input file or one column of drivers in the Scenario input file. Make a written plan for your set of runs in advance; this should include giving each planned run a code name that fits into a system. (*See* Workflow / Designing a set of model runs and Workflow / Executing a set of model runs.)
5. After you run the model, rename the main output file (using the code name you planned) and move it to a different folder so it will not be written over the next time you run Mondrian.
6. Before changing any parameter, look through the user guide to see if it is a reasonable thing to explore changes in. (Some parameters are present only for the devleopment team during model testing and are not useful for model users to explore changes in.)

# Appendix II. Lesson Plans for Teachers

This Appendix includes notes and lesson plans for teachers who want to use Mondrian in the classroom, whether in an undergraduate or graduate class. We envision this Appendix growing over time and eventually may become a separate document.

## Dong a basic comparison of different values of one parameter

In this exercise, students learn how to plan a consistent set of 5 single model runs one at a time (do not use the Batch input file for this exercise). The set of runs simply tests the effect of using 5 different values of a single parameter. In this case, the same Scenario and Seed input files should be used for all runs in the set. The same Parameter file should also be used for all of the model runs in the set, but with only one parameter changed from one run to the next. The value of the single parameter can simply be changed in the ASCII text file each time (and the file re-saved before each run).

If the parameter chosen is ***NinputLandMult***, which controls the annual amount of N inflow to the wetland, you can expect dramatic results in plant species densities, NPP, N cycling, and some C cycling as a result. Alternatively, a more subtle parameter such as ***resorbProp*** could be used, using the nominal (default) value as the center of the set of 5, and bracketing it with 2 lower and 2 higher values. This will have a smaller effect on a fewer set of output variables. It could be interesting to change a species-specific plant parameter like this for one species but not the others and explore the effects it has on plant competition.

Challenge students to think about what effects the parameter change is likely to have and articulate their expectations in advance. Then after conducting the model runs they can consider whether the model results matched, or differed, from their expectations.

Students should make a written plan of the 5 model runs in advance, giving code names to the 5 model runs. The code names could be ‘Ex’ (for exercise), the current date, and then the letters A through E: Ex030118A, Ex030118B … Ex 030118E. Alternatively, the names of the model runs could be N1, N2, …. N5 if the goal is to use 5 different values of N inflow and there is no need to save the 5 model runs after they have been completed and analyzed. An advantage to naming model runs by using the date is that the date is a unique identifier that will never repeat and that it helps the modeler keep exercises in chronological order. If a model is used over a long period of time, the date of the Exercise will also help students to remember what they were doing with the model at that point. For a set of only 5 model runs it may not seem so important, but part of the goal here is to teach students to be careful, clear, and systematic in the way they use a model. (Otherwise there is a strong tendency for students to approach modeling in a quick point-and-click manner and then not be able to remember what they did, or why).

Teach students to keep notes in a modeling log, e.g. a Word document, about what is the purpose of the set of model runs and to list and interpret some key aspect of the results. This is similar to how, if you were working in a chemistry lab, you would not mix chemicals randomly without writing it down; you would keep a lab notebook to methodically record the steps and the results. The modeling log should have a table or list of the 5 code names for the model runs and what parameter was changed, and what its value was in each run. In addition, another good practice in thoughtful modeling is to use the two lines of text at the top of the parameter file in each model run to record the code name of the model run. This will be then be printed in the header of the main output file, where it serves as a double-check in not confusing the output files from different model runs.

After each model run in the set of 5, students should re-name the main output file with a different name (using the 5 codes specified earlier) and move it to a different folder. Once all 5 model runs are complete, the 5 different output files can simply be scanned visually to look for differences, or they can be imported into Excel or another software package for quantitative comparisons or graphing of differences.

Questions for class discussion:

1. Did the differences among the set of 5 model runs agree with, or differ from your expectations? If they differed from your expectations, are there other clues in the model results that can help you to understand why?
2. Do the differences among the set of 5 runs represent a real difference in wetland ecology, or do they measure uncertainty produced by the model, or is it a mixture of both? How does an investigator determine whether differences in model runs represent something “real”?
3. In this exercise we varied only one parameter. What would be the pros and cons of instead varying 10 different parameters all at once?

# Appendix III. Mondrian VB coding conventions

This Appendix contains detailed conventions for coding, storing data in arrays, naming conventions for variables, and so on, in the Mondrian code in Visual Basic. This is needed only by the development team or by anyone wishing to make changes in the VB source code.

*CAUTION! this is a very complex individual-based model. It would be difficult for someone who was not involved in model development to make self-consistent changes in the source code because many parts of the model depend on other parts in subtle but important ways.*

## General conventions

Pools are named in all caps.

In model code, C and N pools are tracked and stored as absolute C and N pools in g C and g N, not per m2. This simplifies code for plant uptake, translocation among rhizomes, and litterfall. When writing code, wherever pools or fluxes are needed on an areal basis, just divide pool size for a grid cell by the cell size, or divide ecosystem-summed pool size by the whole grid area.

Fluxes of N generally begin with a captial N, then lowercase. Fluxes of C generally begin with a captial C, then lowercase.

Passing variables to subroutines: A variable is passed ByVal if the subroutine that is being called needs to know it but does not need to change it. A variable is passed ByRef if the subroutine being called needs to change the variable and return it back to the calling Sub with a changed value.

Variable scoping:

Defined and scoped internal to a subroutine where possible.

Scoped Private in MondrianMod1 (the code module) if many subroutines need access to it.

Scoped Public in MondrianMod 1 if the user inteface needs access to it.

## Array that stores plant individual locations

pil(i, j, k) : array that stores plant individual locations, stored as real numbers.

Indices i, j, k are integers, defined as:

i = plant species

j = plant individual, within the species

k = location coordinates

k = 1 : x location in coordinate system, real number, units m.

k = 2 : y location in coordinate system, real number, units m.

if either location coordinate, x or y, is zero, that signifies that there is no plant individual stored for that species i and individual index j, and that combination of i, j can be used to create a new individual.

## Array that stores plant genet and rhizome connections

pgr(i, j, k) : array that stores plant genets and rhizome node connectivity, stored as integers.

i = plant species

j = plant individual, within the species

k = genet and rhizome node information

k = 1 : number of the genet, or clone, within that species, as integer.

k = 2 : when this rhizome indiv was created, did it wrap around the torus?

value 0 = no

value 1 = yes

k = 3 : index recording whether this particular ramet has reproduced vegetatively yet

value 0 = no

value 1 = yes

k = 4 : age of the ramet, in years, as integer

value 0 = in its first year

value 1 = 1+ years old; in its 2nd year

etc.

k = 5 : number of years this belowground indiv has lived without making an aboveground stem

value 0 = either has made a stem or is in its 1st growing season without one

value 1 = did not make a stem last year, is in its 2nd growing season without one

k = 6 through 10 is used to store the identifiers of the parents of this individual up to 5 parents back in the chain. If any of these = 0, there was never a parent that far back. (Note: This array remembers the index numbers even if the parent, grandparent, etc. has died. So a separate test is needed to determine if that parent, grandparent, etc. is still alive.)

k = 6 : indiv index number (j), within this species, of the rhizome node 1 parent back

k = 7 : indiv index number (j), within this species, of the rhizome node 2 parents back

k = 8 : indiv index number (j), within this species, of the rhizome node 3 parents back

k = 9 : indiv index number (j), within this species, of the rhizome node 4 parents back

k = 10 : indiv index number (j), within this species, of the rhizome node 5 parents back

## Array that stores angles of rhizome node connections

pga(i, j) : array that stores plant genet angle: compass angle orientation of line between this individual and its immediate parent. Value is stored as a real number, an angle in radians, 0 to 2π (6.28).

i = plant species

j = plant individual

## Array that stores plant internal pools

PIP(i, j, k) : array that stores plant individual pools, stored as real numbers.

Indices i, j, k are integers, defined as:

i = plant species

j = plant individual

k = various pools

k = 1 : aboveground (including stem) living organic C pool, real number, units g C.

k = 2 : aboveground (including stem) living organic N pool, real number, units g N.

k = 3 : belowground (including rhizome) organic C pool, real number, units g C. This C pool is subject to relative growth rate, and can be used to produce aboveground stems, and to produce new rhizomes.

k = 4 : belowground (including rhizome) organic N pool, real number, units g N.

k = 5 : belowground (including internode) organic C that is structural, real no., g C. This C pool is a cost of vegetative repro.; it is not subject to relative growth rate for an individual rhizome.

k = 6 : aboveground (including stem) dead organic C pool, real number, units g C

k = 7 : aboveground (including stem) dead organic N pool, real number, units g N

(future additions could include plant P pool, or separation of N pool into foliage, stem, roots)

## Array that stores grid cell pools

GCP(i, j, k) : array that stores grid cell pools, stored as real numbers.

Indices i, j, k are integers, defined as:

i = grid index of x location, integer (mult by s to get x loc of upper right corner in meters)

j = grid index of y location, integer (mult by s to get y loc of upper right corner in meters)

k = various pools

k = 1 : aboveground litter organic C pool, real number, units g C.

k = 2 : aboveground litter organic N pool, real number, units g N.

k = 3 : available inorganic NH4 pool, real number, units g NH4-N. [[16]](#footnote-16)

k = 4 : available inorganic NO3 pool, real number, units g NO3-N.

k = 5 : MSOM (mineral soil organic matter) C pool. Conceptually not litter, and below the "zero" depth point. Real number, units g C.

k = 6 : MSOM (mineral soil organic matter) N pool. Conceptually not litter, and below the "zero" depth point. Real number, units g N.

k = 7 : belowground litter organic C pool, real number, units g C.

k = 8 : belowground litter organic N pool, real number, units g N.

k = 9 : MUCK organic matter C pool, real number, units g C.

k = 10 : MUCK organic matter organic N pool, real number, units g N.

(future additions could include detrital P, available P pools, or DOC, DON)

## Arrays that store mass-weighted first-order k for litter decomposition

gckcalcs(i, j, m) : array that stores information for calculating grid cell litter mass-weighted first-order k values

i = grid index of x location, integer

j = grid index of y location, integer

m = 1 : current year litter C mass that has entered grid cell

m = 2 : current year litter C mass \* k value as it entered grid cell

Mass-weighted k value in the grid cell can be calculated by dividing (m=1) by (m=2) value

kL(i, j) : array that stores grid cell litter first-order k values

## Regional nutrient pools

RNP(i) = Regional nutrient pool sizes.

i = various pools.

i = 1 : available NH4-N pool in regional nutrient pool, units g NH4-N.

i = 2 : available NO3-N pool in regional nutrient pool, units g NO3-N.

(future additions could include available P pools, or DOC, DON)

## Logical Flow Analysis

LFASpp(i, j) = array that stores "Logical Flow Analysis" by species.

This array tracks the logical flow in the model, to see which various conditions are limiting the growth or reproduction of individuals each year.

i = species

j = logical flow counters, as integers. Incremented each time a certain condition occurs for one individual. Can occur numerous times per year for each individual. All are reset to zero at the start of each year. So all are annual counters.

1 = uptake N demand was scaled back due to limitation of available inorganic N in grid cell

2 = there is enough rhiz C, but not enough rhiz N, to est an AG stem without parental subsidy

3 = there is enough rhiz N, but not enough rhiz C, to est an AG stem without parental subsidy

4 = there is both enough rhiz C and N at same time to estab AG stem, with and without parental subsidy (i.e. total number of new aboveground stems created)

5 = aboveground growth limited because not enough N is in belowground part to send up to stem

6 = not used

7 = number of times that a living individual was completely submerged (counter that increments every time an individual of this species, is submerged, including all dates for same individual)

8 = number of aboveground stems in this species that were killed because they were submerged long enough

9 = number of times a first new daughter rhizome node is created by a terminal node

10 = number of times a terminal branch & node are created by a terminal node

11 = number of times a lateral branch & node are created by a non-terminal node

1. Beginning with v 4.2, inorganic N was separated into NH4-N and NO3-N. In v 4.1 and earlier, these were combined into a single pool of inorganic N. [↑](#footnote-ref-1)
2. The reason for allowing an attempted new stem initiation only every 5th day was both to speed model operation, and to limit the rate of rhizome reproduction to a realistic rate under high-nutrient conditions. [↑](#footnote-ref-2)
3. When simulating individual plants, units used in the model code are absolute g C and g N, not normalized per unit area. In printed model results, in most cases these are aggregated over an area and divided by the appropriate area to express results in g C / m2 or g N / m2. [↑](#footnote-ref-3)
4. This description of light competition in Mondrian derives largely from Martina et al. (2016). [↑](#footnote-ref-4)
5. This description of competition for N derives largely from Currie et al. (2014), with some greater detail and slight modifications reflecting model updates since that paper was published. [↑](#footnote-ref-5)
6. Note that this is one important way in which the assumption of N limitation is hard-wired into model processes. Note also that because growth is formulated this way, a higher C:N ratio in aboveground tissue will make a plant species be more productive for a given amount of N uptake. Mondrian currently does not include any tradeoff in which lower foliar N concentrations could result in lower photosynthesis. [↑](#footnote-ref-6)
7. If you have version 4.0, clonal branching is always OFF and is not included as a switch in the parameter input file. Clonal branching is still in beta-testing by the development team and will be released in version 4.1. [↑](#footnote-ref-7)
8. The species-specific parameters for both terminal branching and lateral branching are set by the user in the Paramater input file. [↑](#footnote-ref-8)
9. The thresholds, both for having enough C and N to pass to the new daughter node, and for having enough C and N for the parent node to retain, are expressed as a proportion of maximum belowground size and are set by the user in the Parameter input file. [↑](#footnote-ref-9)
10. The reason for having the parameter ***maxindivs*** is simply to allocate adequate computer memory resources at the start of a model run and to speed model operation. If Mondrian tries to create an additional individual beyond this number, it will not be created, an error message window will appear, and an error will be logged. Set this parameter high enough that it is unlikely to be reached, but not so high that too much memory is set aside. [↑](#footnote-ref-10)
11. In the parameter input file, species-specific paramters like these are listed as ***cidist1, cidist2, cidist3***, and so on for species 1, 2, 3, up to 5 species. When these are read into the model from the input file and used in the code, all species-specific parameters like these are stored in a one-dimensional array, in this case cidist(i) where i = 1 to 5. [↑](#footnote-ref-11)
12. The version that is freely distributed from the website may be 1 to 2 versions behind the version currently in development and testing. If you would like the most recent version, contact the development team and we will be happy to make it available if we feel it has been tested sufficiently. [↑](#footnote-ref-12)
13. Important caveat: Mondrian is a very complex individual-based model. It would be difficult for someone who was not involved in model development to make self-consistent changes to the code because many parts of the model depend on other parts in subtle but important ways. [↑](#footnote-ref-13)
14. Previous to version 4.3, results files were ASCII text files (not csv format). [↑](#footnote-ref-14)
15. At various points in model development, additional output files were created for detailed testing. Some of these are still available but seldom used when the model is being used for ecological research or applied management purposes. For more information, contact the development team. [↑](#footnote-ref-15)
16. Previous to version 4.2 there was only one pool of available N, stored in k = 3. [↑](#footnote-ref-16)