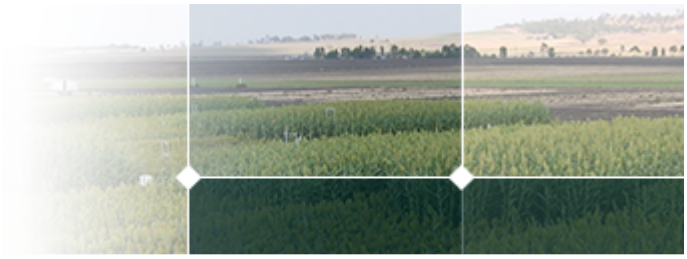


0.0.0.0

1 The APSIM Grapevine Model

ZHU, J., Brown, H.E., Parker, A.K.,Trought, M.C.T.

1.1 Building the model.

The APSIM Grapevine model has been adapted from the potato model Brown et al., 2011 and developed using the Plant Modelling Framework (PMF) of Brown et al., 2014. This new framework provides a library of plant organ and process submodels that can be coupled, at runtime, to construct a model in much the same way that models can be coupled to construct a simulation. This means that dynamic composition of lower level process and organ classes (e.g. photosynthesis, leaf) into larger constructions (e.g. maize, wheat, sorghum) can be achieved by the model developer without additional coding.

The model consists of: * a phenology model to simulate development between growth phases
* a structure model to simulate plant morphology
* a collection of organs to simulate the various plant parts
* an arbitrator to allocate resources (N, biomass) to the various plant organs

1.2 Pecularities of Grapevine

Grapevine is a challenging crop to model because it is a perennial woody plant. A budding phase was added for simulating the start of the vegetative phase. This budding phase was similar to the emergence phase for annual crop. There are an enormous number of Grapevine cultivars and trainning systems. They differ in phenology and vegetative growth.

1.3 Including a Grapevine crop in an APSIM simulation

To indluce a Grapevine crop in a simulation the grapevine plant needs to be added to the vineyard, field or zone in which it is to be grown. This can be done by right clicking on the vineyard, selecting Add from the drop down menu and then selecting Plant from the list that comes up.

1.3.1 Setting plant density and bud number (budNumber number and early canopy growth)

The Grapevine Management script sets the number of stems that will appear from plant.

1.3.2 Setting final leaf number (Crop duration)

The duration of a Grapevine crop is determined by the number of leaves that the mainstem produces and how long it takes for these to appear and seness.

2 APSIM Description

The Agricultural Production Systems sIMulator (APSIM) is a farming systems modelling framework that is being actively developed by the APSIM Initiative.

It is comprised of

1. a set of biophysical models that capture the science and management of the system being modelled,
2. a software framework that allows these models to be coupled together to facilitate data exchange between the models,
3. a set of input models that capture soil characteristics, climate variables, genotype information, field management etc,
4. a community of developers and users who work together, to share ideas, data and source code,

5. a data platform to enable this sharing and
6. a user interface to make it accessible to a broad range of users.

The literature contains numerous papers outlining the many uses of APSIM applied to diverse problem domains. In particular, Holzworth et al., 2014; Keating et al., 2003; McCown et al., 1996; McCown et al., 1995 have described earlier versions of APSIM in detail, outlining the key APSIM crop and soil process models and presented some examples of the capabilities of APSIM.

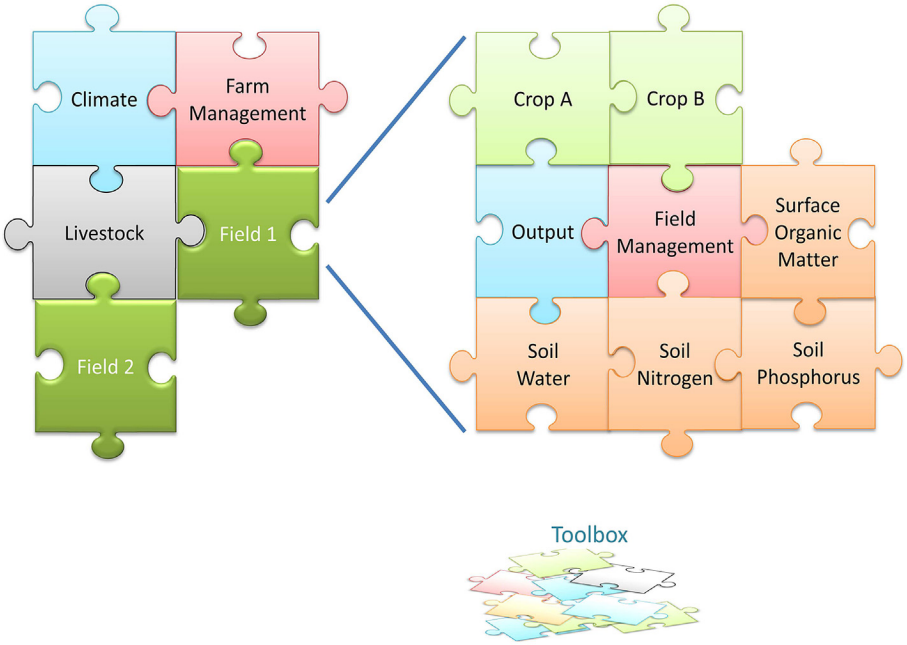


Figure 2: This conceptual representation of an APSIM simulation shows a “top level” farm (with climate, farm management and livestock) and two fields. The farm and each field are built from a combination of models found in the toolbox. The APSIM infrastructure connects all selected model pieces together to form a coherent simulation.*

The APSIM Initiative has begun developing a next generation of APSIM (APSIM Next Generation) that is written from scratch and designed to run natively on Windows, LINUX and MAC OSX. The new framework incorporates the best of the APSIM 7.x framework with an improved supporting framework. The Plant Modelling Framework (a generic collection of plant building blocks) was ported from the existing APSIM to bring a rapid development pathway for plant models. The user interface paradigm has been kept the same as the existing APSIM version, but completely rewritten to support new application domains and the newer Plant Modelling Framework. The ability to describe experiments has been added which can also be used for rapidly building factorials of simulations. The ability to write C# scripts to control farm and paddock management has been retained. Finally, all simulation outputs are written to an SQLite database to make it easier and quicker to query, filter and graph outputs.

The model described in this documentation is for APSIM Next Generation.

APSIM is freely available for non-commercial purposes. Non-commercial use of APSIM means public-good research & development and educational activities. It includes the support of policy development and/or implementation by, or on behalf of, government bodies and industry-good work where the research outcomes are to be made publicly available. For more information visit the licensing page on the APSIM web site

3 Model description

The Grapevine model is constructed from the following list of software components. Details of the implementation and model parameterisation are provided in the following sections.

List of Plant Model Components.

Component Name	Component Type
Phenology	Models.PMF.Phen.Phenology
Structure	Models.PMF.Struct.Structure

Component Name	Component Type
Leaf	Models.PMF.Organs.Leaf
Shoot	Models.PMF.Organs.GenericOrgan
Cordon	Models.PMF.Organs.GenericOrgan
Trunk	Models.PMF.Organs.GenericOrgan
Berry	Models.PMF.Organs.ReproductiveOrgan
Root	Models.PMF.Organs.Root
Arbitrator	Models.PMF.OrganArbitrator

3.1 SauvignonBlanc2C

This cultivar is defined by overriding some of the base parameters of the plant model.

SauvignonBlanc2C makes the following changes:

3.2 SauvignonBlanc4C

This cultivar is defined by overriding some of the base parameters of the plant model.

SauvignonBlanc4C makes the following changes:

3.3 Phenology

This model simulates the development of the crop through successive developmental *phases*. Each phase is bound by distinct growth *stages*. Phases often require a target to be reached to signal movement to the next phase. Differences between cultivars are specified by changing the values of the default parameters shown below.

Dormancy sequence: Paradormancy (Apical dominance), Endodormancy(Chilling requirement), Ecodormancy (forcing unit) Sarvas (1974) separates bud dormancy in two periods: the "rest" period is defined as the period when buds are dormant due to physiological conditions (endodormancy), and the "quiescence" period is when buds remain dormant due to unfavourable environmental conditions (ecodormancy). The current phenological models assume the dormancy starts at March 1st, and budburst is regulated by temperature. Ecodormancy is induced by a period with chilling temperatures, and then followed by a period with forcing temperatures till bud burst (Chuine et al., 2003).

We then simulated the time of flowering, fruitSet,veraison, leaf fall. After one cycle, bud goes to dormancy at March 1st.

Paradormancy, which is the suspension of growth caused by factors outside the meristem but within the plant. It is typically an influence of one organ over another, and includes an apical bud preventing outgrowth of a lower bud, which relates to apical dominance (see Martin 1987 for review).

List of stages and phases used in the simulation of crop phenological development

Phase Number	Phase Name	Initial Stage	Final Stage
1	EndoDormancy	Endodormancy	Ecodormancy
2	Budding	Ecodormancy	BudBurst
3	Flowering	BudBurst	Flowering
4	BerryDevelopment	Flowering	Veraison
5	LeafDeathPhase	Veraison	LeafFall
6	GotoPhase	LeafFall	EndoDormancy

3.3.1 Phenological Phases

3.3.1.1 EndoDormancy Phase

This *phase* goes from Endodormancy to Ecodormancy. It uses a *Target* to determine the duration between development *Stages*. Daily *progress* is accumulated until the *Target* is met and remaining fraction of the day is forwarded to the next phase.

The *Target* and the daily *Progression* toward Ecodormancy are described as follow:

3.3.1.1.1 Target

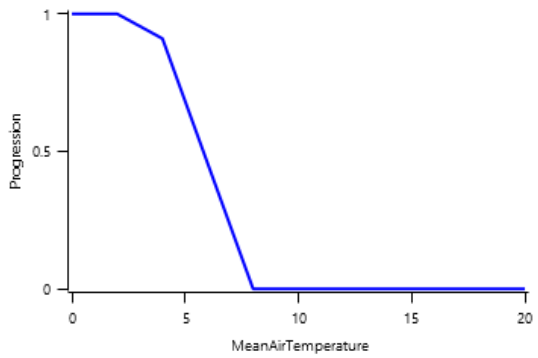
Target = BaseTarget - [Phenology].AccChillBefPrune

Where:

BaseTarget = 8 (days)

3.3.1.1.2 Progression

Progression is calculated from the mean of 3-hourly estimates of air temperature based on daily max and min temperatures.



X	Y
0	1
2	1
4	0.91
8	0
20	0

3.3.1.2 Budding Phase

This *phase* goes from Ecodormancy to BudBurst. It uses a *Target* to determine the duration between development *Stages*. Daily *progress* is accumulated until the *Target* is met and remaining fraction of the day is forwarded to the next phase.

The *Target* and the daily *Progression* toward BudBurst are described as follow:

3.3.1.2.1 Target

Target = 79 (days)

3.3.1.2.2 Progression

Progression =

3.3.1.3 Flowering Phase

This *phase* goes from BudBurst to Flowering. It uses a *Target* to determine the duration between development *Stages*. Daily *progress* is accumulated until the *Target* is met and remaining fraction of the day is forwarded to the next phase.

The *Target* and the daily *Progression* toward Flowering are described as follow:

3.3.1.3.1 Target

Target = 27

Progression = [Phenology].ThermalTime

3.3.1.4 BerryDevelopment Phase

This *phase* goes from Flowering to Veraison. It uses a *Target* to determine the duration between development *Stages*. Daily *progress* is accumulated until the *Target* is met and remaining fraction of the day is forwarded to the next phase.

The *Target* and the daily *Progression* toward Veraison are described as follow:

3.3.1.4.1 Target

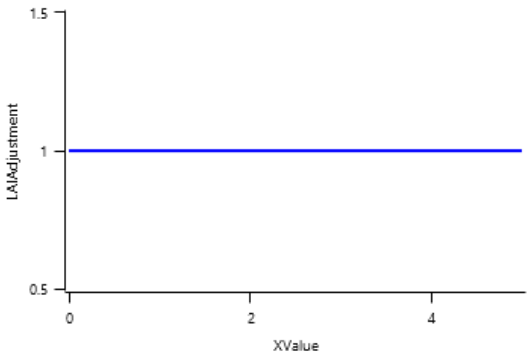
Target = Base × LAIAdjustment

Where:

Base = 40

3.3.1.4.1.1 LAIAdjustment

LAIAdjustment is calculated using linear interpolation



X	Y
0	1
1	1
2	1
3	1
4	1
5	1

XValue = [Grapevine].Leaf.LAI

Progression = [Phenology].ThermalTime

It proceeds until the last leaf on the main-stem has fully senessced. Therefore its duration depends on the number of main-stem leaves that are produced and the rate at which they seness following final leaf appearance.

ThermalTime = [Phenology].ThermalTime

3.3.1.5 GotoPhase Phase

This is a special phase, at LeafFall the phenology is reset to the EndoDormancy phase.

3.3.2 ThermalTime

ThermalTime =

3.3.3 BudBurstDOY

A function is used to provide flowering date as days after sowing(DAS).

Before BudBurst

PreEventValue = 0

On BudBurst the value is set to:

PostEventValue = [Weather].DaysSinceWinterSolstice

3.3.4 FloweringDOY

A function is used to provide flowering date as days after sowing(DAS).

Before Flowering

PreEventValue = 0

On Flowering the value is set to:

PostEventValue = [Weather].DaysSinceWinterSolstice

3.3.5 VeraisonDOY

A function is used to provide flowering date as days after sowing(DAS).

Before Veraison

PreEventValue = 0

On Veraison the value is set to:

PostEventValue = [Weather].DaysSinceWinterSolstice

3.3.6 CurrentSeason

A function is used to capture the year of the current season as in the south hemisphere the growing season extends to the second year.

Before BudBurst

PreEventValue = 0

On BudBurst the value is set to:

PostEventValue = [Clock].Today.Year

3.3.7 AccChillBefPrune

AccChillBefPrune is a daily accumulation of the values of functions listed below between the Veraison and LeafFall stages. Function values added to the accumulate total each day are:

3.3.7.1 LessThanFunction

IF [Weather].DaysSinceWinterSolstice < PruningTime THEN

ChillingUnit = [Phenology].EndoDormancy.Progression

ELSE

Zero = 0

3.4 Structure

The structure model simulates morphological development of the plant to inform the Leaf class when and how many leaves appear and to provide a hight estimate for use in calculating potential transpiration.

3.4.1 Plant and Main-Stem Population

The *Plant.Population* is set at sowing with information sent from a manager script in the Sow method. The *PrimaryBudNumber* is also sent with the Sow method and the main-stem population (*MainStemPopn*) for the crop is calculated as: *MainStemPopn* = *Plant.Population* x *PrimaryBudNumber* Primary bud number is > 1 for crops like potato and grape vine where there are more than one main-stem per plant

3.4.2 Main-Stem leaf appearance

Each day the number of main-stem leaf tips appeared (*LeafTipsAppeared*) is calculated as: *LeafTipsAppeared* += *DeltaTips* Where *DeltaTips* is calculated as: *DeltaTips* = *ThermalTime/Phyllochron* Where *Phyllochron* is the thermal time duration between the appearance of leaf tipx given by:

3.4.2.1 Phyllochron

1.62 between BudBurst and Veraison and a value of zero outside of this period

and *ThermalTime* is given by:

3.4.2.2 ThermalTime

ThermalTime =

LeafTipsAppeared continues to increase until *FinalLeafNumber* is reached where *FinalLeafNumber* is calculated as:

3.4.2.3 FinalLeafNumber

FinalLeafNumber = 25 (number)

3.4.3 Branching and Branch Mortality

The total population of stems (*TotalStemPopn*) is calculated as: *TotalStemPopn = MainStemPopn + NewBranches - NewlyDeadBranches* Where *NewBranches = MainStemPopn* x *BranchingRate* and *BranchingRate* is given by:

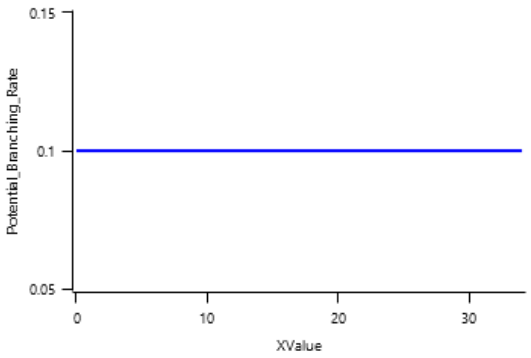
3.4.3.1 BranchingRate

BranchingRate = Potential_Branching_Rate × LinearInterpolationFunction

Where:

3.4.3.1.1 Potential_Branching_Rate

Potential_Branching_Rate is calculated using linear interpolation

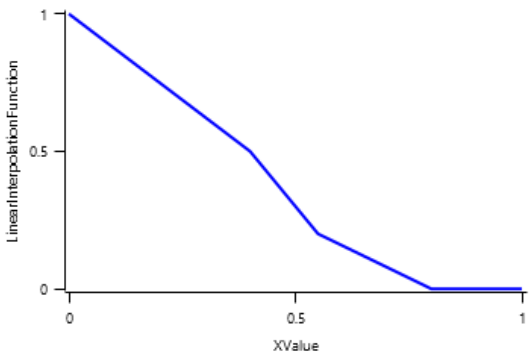


X	Y
0	0.1
7	0.1
8	0.1
13	0.1
14	0.1
34	0.1

XValue = [Structure].LeafTipsAppeared

3.4.3.1.2 LinearInterpolationFunction

LinearInterpolationFunction is calculated using linear interpolation



X	Y
0	1
0.4	0.5
0.55	0.2
0.8	0
1	0

XValue = [Leaf].CoverGreen

NewlyDeadBranches is calcualted as: *NewlyDeadBranches* = (*TotalStemPopn - MainStemPopn*) x *BranchMortality* where *BranchMortality* is given by:

3.4.3.2 BranchMortality

BranchMortality = 0

3.4.4 Height

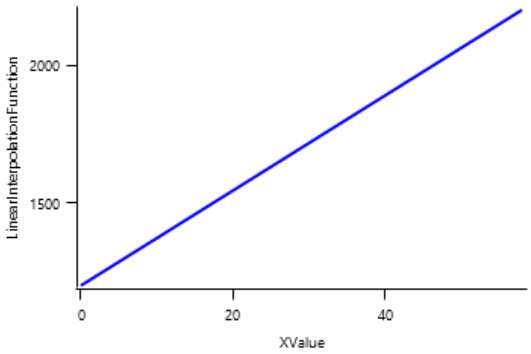
The Height of the crop is calculated by the *HeightModel*:

3.4.4.1 HeightModel

HeightModel is the same as LinearInterpolationFunction until it reaches Veraison stage when it fixes its value

3.4.4.1.1 LinearInterpolationFunction

LinearInterpolationFunction is calculated using linear interpolation



X	Y
0	1200
58	2200

XValue = [Grapevine].Phenology.AccumulatedEmergedTT

3.4.5 MainStemPrimordialnitiationRate

1.8 between BudBurst and Veraison and a value of zero outside of this period

3.4.6 DroughtInducedBranchMortality

DroughtInducedBranchMortality = 0

3.4.7 StemSenescenceAge

StemSenescenceAge = 0

3.4.8 BudNumber

Sets the number of buds on each mains stem to the valud of it child on the BudBurst

3.4.8.1 FractionOfBudBurst

FractionOfBudBurst = BudNumberEffect × CordonDiameterEffect

Where:

3.4.8.1.1 BudNumberEffect

a sigmoid function of the form y = Xmax * 1 / 1 + $e^{-(XValue - Xo) / b}$

XValue = [Structure].PrimaryBudNo

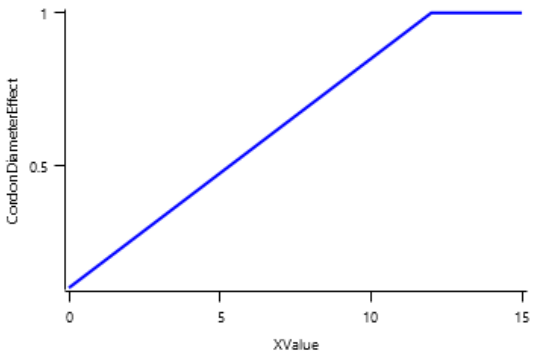
Ymax = 1.4

Xo = 100

b = -54.8977

3.4.8.1.2 CordonDiameterEffect

CordonDiameterEffect is calculated using linear interpolation



X	Y
0	0.1
12	1
15	1

XValue = [Grapevine].Cordon.CordonDiameter

3.5 Leaf

The leaves are modelled as a set of leaf cohorts and the properties of each of these cohorts are summed to give overall values for the leaf organ. A cohort represents all the leaves of a given main- stem node position including all of the branch leaves appearing at the same time as the given main-stem leaf (Lawless et al., 2005). The number of leaves in each cohort is the product of the number of plants per m^2 and the number of branches per plant. The *Structure* class models the appearance of main-stem leaves and branches. Once cohorts are initiated the *Leaf* class models the area and biomass dynamics of each. It is assumed all the leaves in each cohort have the same size and biomass properties. The modelling of the status and function of individual cohorts is delegated to *LeafCohort* classes.

3.5.1 Dry Matter Fixation

The most important DM supply from leaf is the photosynthetic fixation supply. Radiation interception is calculated from LAI using an extinction coefficient of:

3.5.1.1 ExtinctionCoeff

ExtinctionCoeff = Constant

Where:

Constant = 0.5

3.5.1.2 Photosynthesis

Biomass fixation is modelled as the product of intercepted radiation and its conversion efficiency, the radiation use efficiency (RUE) (Monteith et al., 1977). This approach simulates net photosynthesis rather than providing separate estimates of growth and respiration. The potential photosynthesis calculated using RUE is then adjusted according to stress factors, these account for plant nutrition (FN), air temperature (FT), vapour pressure deficit (FVPD), water supply (FW) and atmospheric CO_2 concentration (FCO2). NOTE: RUE in this model is expressed as g/MJ for a whole plant basis, including both above and below ground growth.

3.5.1.2.1 RUE

3.5.1.2.1.1 Juvenile

The value of RUE from BudBurst to Flowering is calculated as follows:

Constant = 1.2 (g/MJ)

3.5.1.2.1.2 Vegetative

The value of RUE from Flowering to Veraison is calculated as follows:

Constant = 1.05 (g/MJ)

3.5.1.2.1.3 Reproductive

The value of RUE from Veraison to LeafFall is calculated as follows:

Constant = 1.5 (g/MJ)

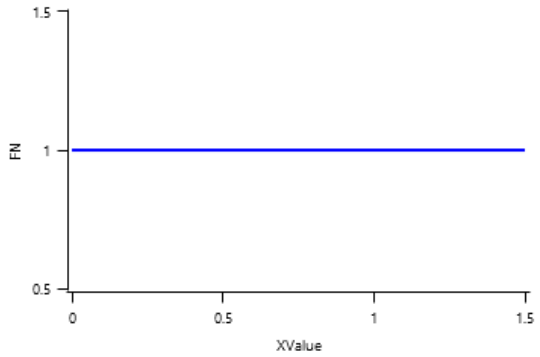
RUE has a value of zero for phases not specified above

3.5.1.2.2 FCO2

This model calculates the CO_2 impact on RUE using the approach of Reyenga et al., 1999.

3.5.1.2.3 FN

FN is calculated using linear interpolation



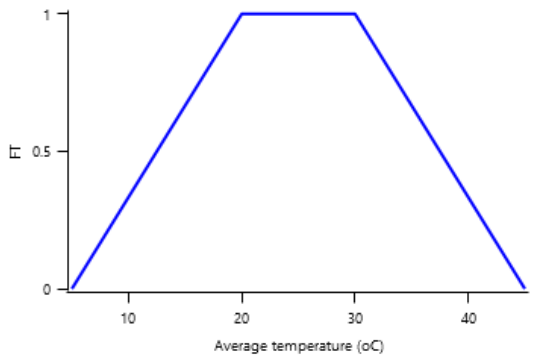
X	Y
0	1
1	1
1.5	1

XValue = [Leaf].Fn

3.5.1.2.4 FT

FT is calculated as a function of daily min and max temperatures, these are weighted toward max temperature according to the specified MaximumTemperatureWeighting factor. A value equal to 1.0 means it will use max temperature, a value of 0.5 means average temperature.

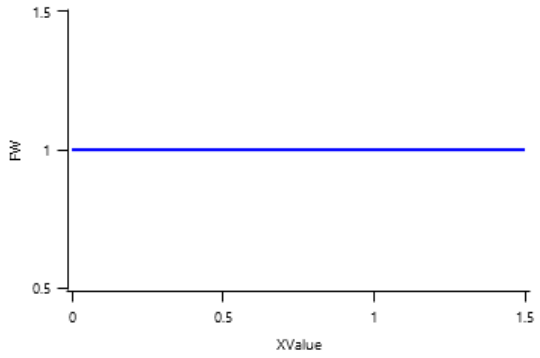
MaximumTemperatureWeighting = 0.75



X	Y
5	0
20	1
30	1
45	0

3.5.1.2.5 FW

FW is calculated using linear interpolation

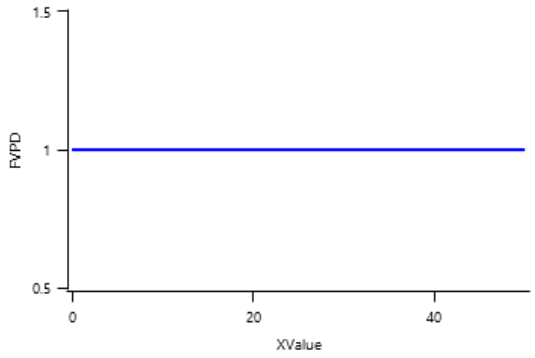


X	Y
0	1
1	1
1.5	1

XValue = [Leaf].Fw

3.5.1.2.6 FVPD

FVPD is calculated using linear interpolation



X	Y
0	1
10	1
50	1

XValue = [Leaf].Photosynthesis.VPD

RadnInt = [Leaf].RadIntTot

ThermalTime = [Structure].ThermalTime

Area = 1000

Area = 0

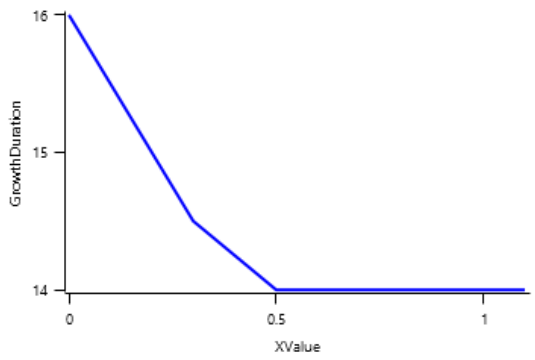
Area = 0

3.5.2 Potential Leaf Area index

Leaf area index is calculated as the sum of the area of each cohort of leaves The appearance of a new cohort of leaves occurs each time Structure.LeafTipsAppeared increases by one. From tip appearance the area of each cohort will increase for a certian number of degree days defined by the *GrowthDuration*

3.5.2.1 GrowthDuration

GrowthDuration is calculated using linear interpolation



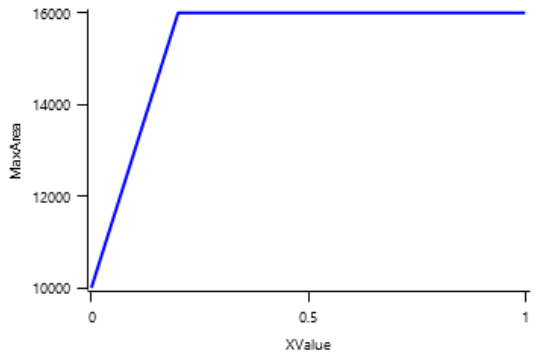
X	Y
0	16
0.3	14.5
0.5	14
1	14
1.1	14

XValue = [Structure].RelativeNodeApperance

If no stress occurs the leaves will reach a Maximum area (*MaxArea*) at the end of the *GrowthDuration*. The *MaxArea* is defined by:

3.5.2.2 MaxArea

MaxArea is calculated using linear interpolation



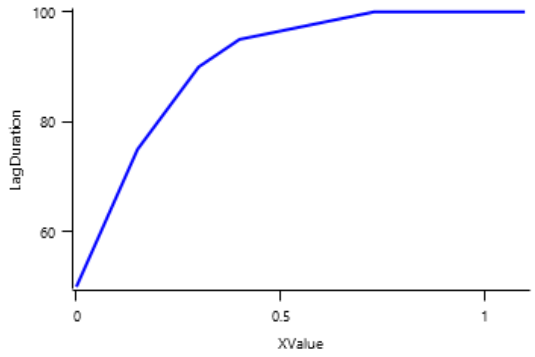
X	Y
0	10000
0.2	16000
1	16000

XValue = [Structure].RelativeNodeApperance

In the absence of stress the leaf will remain at *MaxArea* for a number of degree days set by the *LagDuration* and then area will senesce to zero at the end of the *SenescenceDuration*

3.5.2.3 LagDuration

LagDuration is calculated using linear interpolation

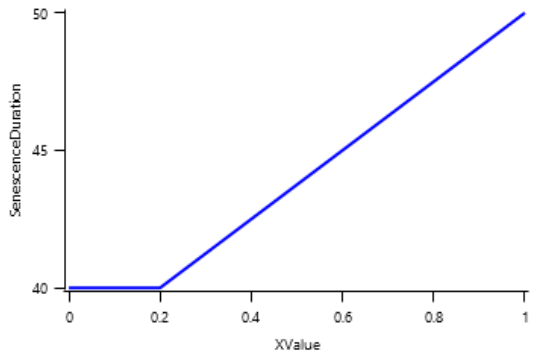


X	Y
0	50
0.15	75
0.3	90
0.4	95
0.73	100
1	100
1.1	100

XValue = [Structure].RelativeNodeApperance

3.5.2.4 SenescenceDuration

SenescenceDuration is calculated using linear interpolation



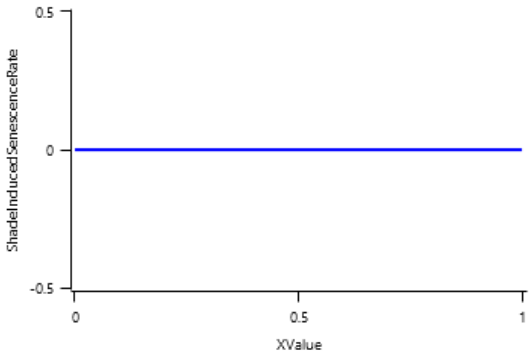
X	Y
0	40
0.2	40
1	50

XValue = [Structure].RelativeNodeApperance

Mutual shading can cause premature senescence of cohorts if the leaf area above them becomes too great. Each cohort models the proportion of its area that is lost to shade induced senescence each day as:

3.5.2.5 ShadeInducedSenescenceRate

ShadeInducedSenescenceRate is calculated using linear interpolation



X	Y
0	0
0.5	0
0.8	0
1	0

XValue = [Leaf].CohortCurrentRankCoverAbove

3.5.3 Stress effects on Leaf Area Index

Stress reduces leaf area in a number of ways. Firstly, stress occuring prior to the appearance of the cohort can reduce cell division, so reducing the maximum leaf size. Leaf captures this by multiplying the *MaxSize* of each cohort by a *CellDivisionStress* factor which is calculated as:

3.5.3.1 CellDivisionStress

CellDivisionStress = 1 (0-1)

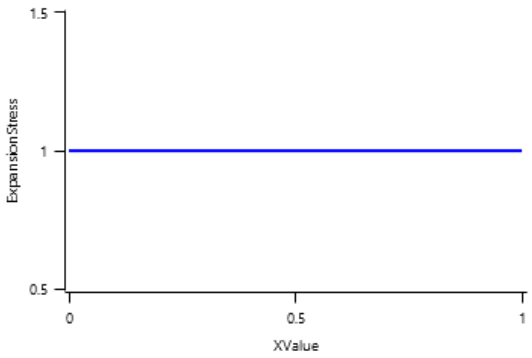
Leaf.FN quantifys the N stress status of the plant and represents the concentration of metabolic N relative the maximum potentil metabolic N content of the leaf calculated as (*Leaf.NConc - MinimumNConc*)/(*CriticalNConc - MinimumNConc*).

Leaf.FW quantifies water stress and is calculated as *Leaf.Transpiration*/*Leaf.WaterDemand*, where *Leaf.Transpiration* is the minimum of *Leaf.WaterDemand* and *Root.WaterUptake*

Stress during the *GrowthDuration* of the cohort reduces the size increase of the cohort by multiplying the potential increase by a *ExpansionStress* factor:

3.5.3.2 ExpansionStress

ExpansionStress is calculated using linear interpolation



X	Y
0	1
1	1

XValue = [Grapevine].Leaf.Fw

Stresses can also acellerate the onset and rate of senescence in a number of ways. Nitrogen shortage will cause N to be retranslocated out of lower order leaves to support the expansion of higher order leaves and other organs When this happens the lower order cohorts will have their area reduced in proportion to the amount of N that is remobilised out of them.

Water stress hastens senescence by increasing the rate of thermal time accumulation in the lag and senescence phases. This is done by multiplying thermal time accumulation by *DroughtInducedLagAcceleration* and *DroughtInducedSenescenceAcceleration* factors, respectively:

3.5.3.3 DroughtInducedLagAcceleration

DroughtInducedLagAcceleration = 1

3.5.3.4 DroughtInducedSenAcceleration

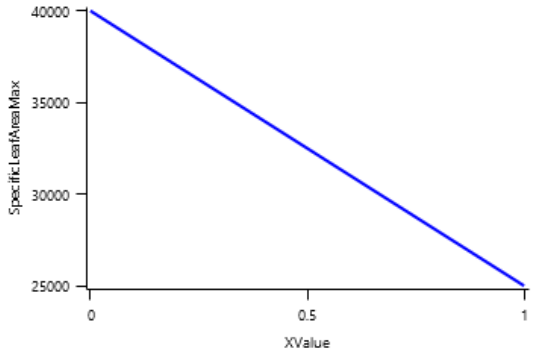
DroughtInducedSenAcceleration = 1

3.5.4 Dry matter Demand

Leaf calculates the DM demand from each cohort as a function of the potential size increment (DeltaPotentialArea) an specific leaf area bounds. Under non stressed conditions the demand for non-storage DM is calculated as *DeltaPotentialArea* divided by the mean of *SpecificLeafAreaMax* and *SpecificLeafAreaMin*. Under stressed conditions it is calculated as *DeltaWaterConstrainedArea* divided by *SpecificLeafAreaMin*.

3.5.4.1 SpecificLeafAreaMax

SpecificLeafAreaMax is calculated using linear interpolation



X	Y
0	40000
1	25000

XValue = [Structure].RelativeNodeApperance

3.5.4.2 SpecificLeafAreaMin

SpecificLeafAreaMin = 20000 (mm2/g)

Non-storage DM Demand is then seperated into structural and metabolic DM demands using the *StructuralFraction*:

3.5.4.3 StructuralFraction

StructuralFraction = 0.8 (percentage)

The storage DM demand is calculated from the sum of metabolic and structural DM (including todays demands) multiplied by a *NonStructuralFraction*:

Unknown child name: NonStructuralFraction

3.5.5 Nitrogen Demand

Leaf calculates the N demand from each cohort as a function of the potential DM increment and N concentration bounds. Structural N demand = *PotentialStructuralDMAllocation* * *MinimumNConc* where:

3.5.5.1 MinimumNConc

MinimumNConc = 0.01 (g/g)

Metabolic N demand is calculated as *PotentialMetabolicDMAllocation* * (*CriticalNConc* - *MinimumNConc*) where:

3.5.5.2 CriticalNConc

CriticalNConc = 0.03 (g/g)

Storage N demand is calculated as the sum of metabolic and structural wt (including todays demands) multiplied by *LuxaryNconc* (*MaximumNConc* - *CriticalNConc*) less the amount of storage N already present. *MaximumNConc* is given by:

3.5.5.3 MaximumNConc

MaximumNConc = 0.05 (g/g)

3.5.6 Drymatter supply

In additon to photosynthesis, the leaf can also supply DM by reallocation of senescing DM and retranslocation of storgage DM: Reallocation supply is a proportion of the metabolic and non-structural DM that would be senesced each day where the proportion is set by:

3.5.6.1 DMReallocationFactor

DMReallocationFactor = 0.1 (percentage)

Retranslocation supply is calculated as a proportion of the amount of storage DM in each cohort where the proportion is set by :

3.5.6.2 DMRetranslocationFactor

DMRetranslocationFactor = 0 (percentage)

3.5.7 Nitrogen supply

Nitrogen supply from the leaf comes from the reallocation of metabolic and storage N in senescing material and the retranslocation of metabolic and storage N. Reallocation supply is a proportion of the Metabolic and Storage DM that would be senesced each day where the proportion is set by:

3.5.7.1 NReallocationFactor

NReallocationFactor = 0 (percentage)

Retranslocation supply is calculated as a proportion of the amount of storage and metabolic N in each cohort where the proportion is set by :

3.5.7.2 NRetranslocationFactor

NRetranslocationFactor = 0 (percentage)

3.5.7.3 DetachmentLagDuration

DetachmentLagDuration = 10 (thermalDay)

3.5.7.4 DetachmentDuration

DetachmentDuration = 200 (degreeDay)

3.5.7.5 InitialNConc

InitialNConc = 0.04 (g/g)

3.5.7.6 StorageFraction

StorageFraction = 0.2 (percentage)

3.5.7.7 SenessingLeafRelativeSize

SenessingLeafRelativeSize = 1 (0-1)

3.5.7.8 LeafSizeShapeParameter

LeafSizeShapeParameter = 0.01 (parameter controls the logistic growth of the leaf in growth stage)

3.5.7.9 MaintenanceRespirationFunction

MaintenanceRespirationFunction = 0 (g/g)

3.5.7.10 LagDurationAgeMultiplier

LagDurationAgeMultiplier = 1 1 1

3.5.7.11 SenescenceDurationAgeMultiplier

SenescenceDurationAgeMultiplier = 1 1 1

3.5.7.12 LeafSizeAgeMultiplier

LeafSizeAgeMultiplier = 1 1 1 1 1 1 1 1 1 1 1

3.5.7.13 RemobilisationCost

RemobilisationCost = 0

3.5.7.14 CarbonConcentration

CarbonConcentration = 0.4

3.5.8 FRGRFunction

FRGRFunction = minimum (RUE_FT, Others)

Where:

RUE_FT = [Leaf].Photosynthesis.FT

3.5.8.1 Others

Others = minimum (RUE_FN, RUE_FVPD)

Where:

RUE_FN = [Leaf].Photosynthesis.FN

RUE_FVPD = [Leaf].Photosynthesis.FVPD

3.5.9 Total Biomass

This is a composite biomass class, representing the sum of 1 or more biomass objects.

Total summarises the following biomass objects:

- [Leaf].Live
- [Leaf].Dead

3.5.10 BiomassRemovalDefaults

This organ will respond to certain management actions by either removing some of its biomass from the system or transferring some of its biomass to the soil surface residues. The following table describes the default proportions of live and dead biomass that are transferred out of the simulation using "Removed" or to soil surface residue using "To Residue" for a range of management actions. The total percentage removed for live or dead must not exceed 100%. The difference between the total and 100% gives the biomass remaining on the plant. These can be changed during a simulation using a manager script.

Method	% Live Removed	% Dead Removed	% Live To Residue	% Dead To Residue
Harvest	0	0	0	0
Cut	0	0	0	0
Prune	0	0	0	0
Graze	0	0	0	0

3.5.11 FrostFraction

0 between Veraison and LeafFall and a value of zero outside of this period

3.5.12 DMConversionEfficiency

DMConversionEfficiency = 1

3.5.13 RemobilisationCost

RemobilisationCost = 0

3.5.14 CarbonConcentration

CarbonConcentration = 0.4

3.5.15 StructuralFraction

StructuralFraction = 0.8 (g/g)

3.5.16 StomatalConductanceCO2Modifier

StomatalConductanceCO2Modifier = 1

3.6 Shoot

This organ is simulated using a GenericOrgan type. It is parameterised to calculate the growth, senescence, and detachment of any organ that does not have specific functions.

3.6.1 Dry Matter Demand

The dry matter demand for the organ is calculated as defined in DMDemands, based on the DMDemandFunction and partition fractions for each biomass pool.

3.6.1.1 DMDemands

This is the collection of functions for calculating the demands for each of the biomass pools (Structural, Metabolic, and Storage).

3.6.1.1.1 Structural

Structural = DMDemandFunction × StructuralFraction

Where:

3.6.1.1.1.1 DMDemandFunction

DMDemandFunction = PartitionFraction [Arbitrator].DM.TotalFixationSupply

Where:

PartitionFraction = 0.2

StructuralFraction = 1 (g/g)

3.6.1.1.2 Metabolic

Metabolic = 0 (g/m2)

3.6.1.1.3 Storage

The partitioning of daily growth to storage biomass is based on a storage fraction.

3.6.1.1.3.1 StorageFraction

StorageFraction = 1 - [Shoot].DMDemands.Structural.StructuralFraction

3.6.2 Nitrogen Demand

The N demand is calculated as defined in NDemands, based on DM demand the N concentration of each biomass pool.

3.6.2.1 NDemands

This is the collection of functions for calculating the demands for each of the biomass pools (Structural, Metabolic, and Storage).

3.6.2.1.1 Structural

Structural = [Shoot].minimumNconc × [Shoot].potentialDMAllocation.Structural

3.6.2.1.2 Metabolic

Metabolic = MetabolicNconc × [Shoot].potentialDMAllocation.Structural

Where:

3.6.2.1.2.1 MetabolicNconc

MetabolicNconc = [Shoot].criticalNConc - [Shoot].minimumNconc

3.6.2.1.3 Storage

The partitioning of daily growth to storage biomass attempts to bring the organ's N content to the maximum concentration.

Storage = [Shoot].maximumNconc × ([Shoot].Live.Wt + potentialAllocationWt) - [Shoot].Live.N

The demand for storage N is further reduced by a factor specified by the [Shoot].NitrogenDemandSwitch.

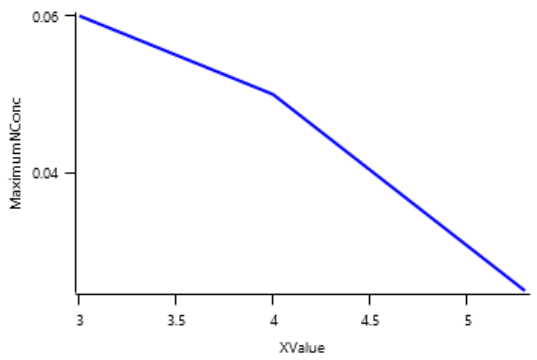
3.6.2.2 MinimumNConc

MinimumNConc = 0.006 (g/g)

CriticalNConc = [Shoot].MinimumNConc

3.6.2.3 MaximumNConc

MaximumNConc is calculated using linear interpolation



X	Y
3	0.06
4	0.05
5.3	0.025

XValue = [Phenology].Stage

The demand for N is reduced by a factor specified by the NitrogenDemandSwitch.

3.6.2.4 NitrogenDemandSwitch

A value of 1 is returned if phenology is between BudBurst and LeafFall phases, otherwise a value of 0 is returned.

3.6.3 Dry Matter Supply

Shoot does not reallocate DM when senescence of the organ occurs.

Shoot does not retranslocate non-structural DM.

3.6.4 Nitrogen Supply

Shoot will reallocate 20% of N that senesces each day.

Shoot will retranslocate 5% of non-structural N each day.

3.6.5 Senescence and Detachment

Shoot has senescence parameterised to zero so all biomass in this organ will remain alive.

Shoot has detachment parameterised to zero so all biomass in this organ will remain with the plant until a defoliation or harvest event occurs.

3.6.6 BiomassRemovalDefaults

This organ will respond to certain management actions by either removing some of its biomass from the system or transferring some of its biomass to the soil surface residues. The following table describes the default proportions of live and dead biomass that are transferred out of the simulation using "Removed" or to soil surface residue using "To Residue" for a range of management actions. The total percentage removed for live or dead must not exceed 100%. The difference between the total and 100% gives the biomass remaining on the plant. These can be changed during a simulation using a manager script.

Method	% Live Removed	% Dead Removed	% Live To Residue	% Dead To Residue
Harvest	0	0	0	0
Cut	0	0	0	0
Prune	0	0	0	0
Graze	0	0	0	0

3.7 Cordon

This organ is simulated using a GenericOrgan type. It is parameterised to calculate the growth, senescence, and detachment of any organ that does not have specific functions.

3.7.1 Dry Matter Demand

The dry matter demand for the organ is calculated as defined in DMDemands, based on the DMDemandFunction and partition fractions for each biomass pool.

3.7.1.1 DMDemands

This is the collection of functions for calculating the demands for each of the biomass pools (Structural, Metabolic, and Storage).

3.7.1.1.1 Structural

Structural = DMDemandFunction × StructuralFraction

Where:

3.7.1.1.1.1 DMDemandFunction

DMDemandFunction = PartitionFraction [Arbitrator].DM.TotalFixationSupply

Where:

PartitionFraction = 0.01

StructuralFraction = 0.8 (percentage)

3.7.1.1.2 Metabolic

Metabolic = 0 (g/m2)

3.7.1.1.3 Storage

The partitioning of daily growth to storage biomass is based on a storage fraction.

3.7.1.1.3.1 StorageFraction

StorageFraction = 1 - [Cordon].DMDemands.Structural.StructuralFraction

3.7.2 Nitrogen Demand

The N demand is calculated as defined in NDemands, based on DM demand the N concentration of each biomass pool.

3.7.2.1 NDemands

This is the collection of functions for calculating the demands for each of the biomass pools (Structural, Metabolic, and Storage).

3.7.2.1.1 Structural

Structural = [Cordon].minimumNconc × [Cordon].potentialDMAllocation.Structural

3.7.2.1.2 Metabolic

Metabolic = MetabolicNconc × [Cordon].potentialDMAllocation.Structural

Where:

3.7.2.1.2.1 MetabolicNconc

MetabolicNconc = [Cordon].criticalNConc - [Cordon].minimumNconc

3.7.2.1.3 Storage

The partitioning of daily growth to storage biomass attempts to bring the organ's N content to the maximum concentration.

Storage = [Cordon].maximumNconc × ([Cordon].Live.Wt + potentialAllocationWt) - [Cordon].Live.N

The demand for storage N is further reduced by a factor specified by the [Cordon].NitrogenDemandSwitch.

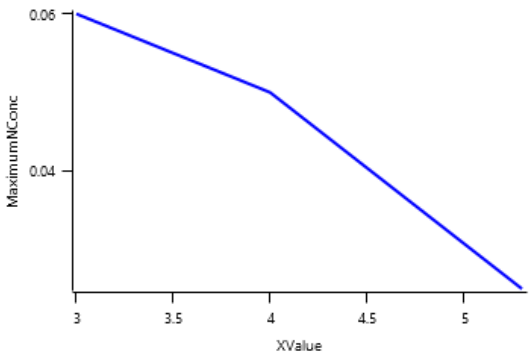
3.7.2.2 MinimumNConc

MinimumNConc = 0.006 (g/g)

CriticalNConc = [Cordon].MinimumNConc

3.7.2.3 MaximumNConc

MaximumNConc is calculated using linear interpolation



X	Y
3	0.06
4	0.05
5.3	0.025

XValue = [Phenology].Stage

The demand for N is reduced by a factor specified by the NitrogenDemandSwitch.

3.7.2.4 NitrogenDemandSwitch

A value of 1 is returned if phenology is between BudBurst and LeafFall phases, otherwise a value of 0 is returned.

3.7.3 Dry Matter Supply

Cordon will reallocate 20% of DM that senesces each day.

Cordon does not retranslocate non-structural DM.

3.7.4 Nitrogen Supply

Cordon does not reallocate N when senescence of the organ occurs.

Cordon will retranslocate 5% of non-structural N each day.

3.7.5 Senescence and Detachment

Cordon has senescence parameterised to zero so all biomass in this organ will remain alive.

Cordon has detachment parameterised to zero so all biomass in this organ will remain with the plant until a defoliation or harvest event occurs.

3.7.6 BiomassRemovalDefaults

This organ will respond to certain management actions by either removing some of its biomass from the system or transferring some of its biomass to the soil surface residues. The following table describes the default proportions of live and dead biomass that are transferred out of the simulation using "Removed" or to soil surface residue using "To Residue" for a range of management actions. The total percentage removed for live or dead must not exceed 100%. The difference between the total and 100% gives the biomass remaining on the plant. These can be changed during a simulation using a manager script.

Method	% Live Removed	% Dead Removed	% Live To Residue	% Dead To Residue
Harvest	0	0	0	0
Cut	0	0	0	0
Prune	0	0	0	0
Graze	0	0	0	0

3.8 Trunk

This organ is simulated using a GenericOrgan type. It is parameterised to calculate the growth, senescence, and detachment of any organ that does not have specific functions.

3.8.1 Dry Matter Demand

The dry matter demand for the organ is calculated as defined in DMDemands, based on the DMDemandFunction and partition fractions for each biomass pool.

3.8.1.1 DMDemands

This is the collection of functions for calculating the demands for each of the biomass pools (Structural, Metabolic, and Storage).

3.8.1.1.1 Structural

Structural = DMDemandFunction × StructuralFraction

Where:

3.8.1.1.1.1 DMDemandFunction

DMDemandFunction = PartitionFraction [Arbitrator].DM.TotalFixationSupply

Where:

PartitionFraction = 0.05

StructuralFraction = 0.8 (g/g)

3.8.1.1.2 Metabolic

Metabolic = 0 (g/m2)

3.8.1.1.3 Storage

The partitioning of daily growth to storage biomass is based on a storage fraction.

3.8.1.1.3.1 StorageFraction

StorageFraction = 1 - [Trunk].DMDemands.Structural.StructuralFraction

3.8.2 Nitrogen Demand

The N demand is calculated as defined in NDemands, based on DM demand the N concentration of each biomass pool.

3.8.2.1 NDemands

This is the collection of functions for calculating the demands for each of the biomass pools (Structural, Metabolic, and Storage).

3.8.2.1.1 Structural

Structural = [Trunk].minimumNconc × [Trunk].potentialDMAllocation.Structural

3.8.2.1.2 Metabolic

Metabolic = MetabolicNconc × [Trunk].potentialDMAllocation.Structural

Where:

3.8.2.1.2.1 MetabolicNconc

MetabolicNconc = [Trunk].criticalNConc - [Trunk].minimumNconc

3.8.2.1.3 Storage

The partitioning of daily growth to storage biomass attempts to bring the organ's N content to the maximum concentration.

Storage = [Trunk].maximumNconc × ([Trunk].Live.Wt + potentialAllocationWt) - [Trunk].Live.N

The demand for storage N is further reduced by a factor specified by the [Trunk].NitrogenDemandSwitch.

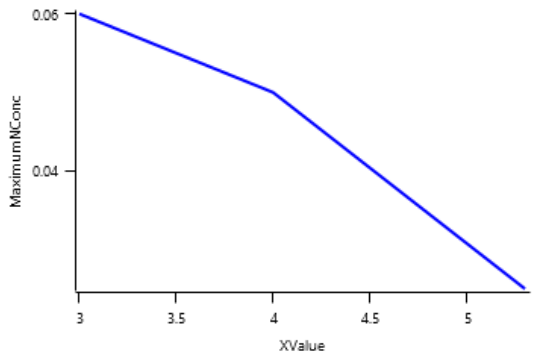
3.8.2.2 MinimumNConc

MinimumNConc = 0.006

CriticalNConc = [Trunk].MinimumNConc

3.8.2.3 MaximumNConc

MaximumNConc is calculated using linear interpolation



X	Y
3	0.06
4	0.05
5.3	0.025

XValue = [Phenology].Stage

The demand for N is reduced by a factor specified by the NitrogenDemandSwitch.

3.8.2.4 NitrogenDemandSwitch

A value of 1 is returned if phenology is between BudBurst and LeafFall phases, otherwise a value of 0 is returned.

3.8.3 Dry Matter Supply

Trunk will reallocate 20% of DM that senesces each day.

Trunk will retranslocate 10% of non-structural DM each day.

3.8.4 Nitrogen Supply

Trunk does not reallocate N when senescence of the organ occurs.

Trunk will retranslocate 5% of non-structural N each day.

3.8.5 Senescence and Detachment

Trunk has senescence parameterised to zero so all biomass in this organ will remain alive.

Trunk has detachment parameterised to zero so all biomass in this organ will remain with the plant until a defoliation or harvest event occurs.

3.8.6 BiomassRemovalDefaults

This organ will respond to certain management actions by either removing some of its biomass from the system or transferring some of its biomass to the soil surface residues. The following table describes the default proportions of live and dead biomass that are transferred out of the simulation using "Removed" or to soil surface residue using "To Residue" for a range of management actions. The total percentage removed for live or dead must not exceed 100%. The difference between the total and 100% gives the biomass remaining on the plant. These can be changed during a simulation using a manager script.

Method	% Live Removed	% Dead Removed	% Live To Residue	% Dead To Residue
Harvest	0	0	0	0
Cut	0	0	0	0
Prune	0	0	0	0
Graze	0	0	0	0

3.9 Berry

This organ uses a generic model for plant reproductive components. Yield is calculated from its components in terms of organ number and size (for example, grain number and grain size).

The final yield was calculated by vines per hectare, shoots per vine, bunches per shoot, flower number per bunch, berries per bunch,and mean berry weight. The effects of temperature and carbon status were or will be considered on bunches per shoot, flower number per bunch, berries per bunch,and mean berry weight.

3.9.1 MetFactors

The final yield was calculated by vines per hectare, shoots per vine, bunches per shoot, flower number per bunch, berries per bunch,and mean berry weight. The effects of temperature and carbon status were or will be considered on bunches per shoot, flower number per bunch, berries per bunch,and mean berry weight.

3.9.1.1 BUN_TmaxIni

BUN_TmaxIni = TmaxIni / Time

Where:

A function that accumulates values from child functions

Tmax = [Weather].MaxT

A function that accumulates values from child functions

Days = 1 (days/day)

3.9.1.2 BUN_RadIni

BUN_RadIni = RadIni / Time

Where:

A function that accumulates values from child functions

Rad = [Weather].Radn

A function that accumulates values from child functions

Days = 1 (days/day)

3.9.1.3 BEN_TmeanFlow

BEN_TmeanFlow = TmeanFlow / Time

Where:

A function that accumulates values from child functions

Tmean = [Weather].MeanT

A function that accumulates values from child functions

Days = 1 (days/day)

3.9.1.4 BEN_RainTotFlow

BEN_RainTotFlow = RainTotFlow / Constant

Where:

A function that accumulates values from child functions

RainTot = [Weather].Rain

Constant = 1

3.9.1.5 BEN_TmaxIni

BEN_TmaxIni = TmaxIni / Time

Where:

A function that accumulates values from child functions

Tmax = [Weather].MaxT

A function that accumulates values from child functions

Days = 1 (days/day)

3.9.1.6 BM_TmeanFlow

BM_TmeanFlow = TmeanFlow / Time

Where:

A function that accumulates values from child functions

Tmean = [Weather].MeanT

A function that accumulates values from child functions

Days = 1 (days/day)

3.9.1.7 BM_RainTotFlow

BM_RainTotFlow = RainTotFlow / Constant

Where:

A function that accumulates values from child functions

RainTot = [Weather].Rain

Constant = 1

3.9.1.8 BM_RadFlow

BM_RadFlow = RadFlow / Time

Where:

A function that accumulates values from child functions

Rad = [Weather].Radn

A function that accumulates values from child functions

Days = 1 (days/day)

3.9.1.9 BM_RainTotVer

BM_RainTotVer = RainTotVer / Constant

Where:

A function that accumulates values from child functions

RainTot = [Weather].Rain

Constant = 1

3.9.1.10 BUN_TmaxIni1

Before Veraison

PreEventValue = 21

On Veraison the value is set to:

PostEventValue = [Berry].MetFactors.BUN_TmaxIni

3.9.1.11 BUN_RadIni1

Before Veraison

PreEventValue = 21

On Veraison the value is set to:

PostEventValue = [Berry].MetFactors.BUN_RadIni

3.9.1.12 BEN_TmaxIni1

Before Veraison

PreEventValue = 21

On Veraison the value is set to:

PostEventValue = [Berry].MetFactors.BEN_TmaxIni

3.9.2 YieldComponent

3.9.2.1 BunchesPerVine

here we use a trigger function for bunches per shoot as the value of mean flowering temperature Y0 is updating itself during the growing season, which would cause inconvience if we directly correlates the bunches per shoot and mean flowering temperature Y0.

Numbers were guessed based on Eltom 2013 Fig. 7.7. y=0.0981*Ta + 0.35

Bunches per shoot is affected by the temperature and CHO during initiation, it is further affected by the CHO during bud burst. The later decided whether the potential initiated inflorescence can grow out successfully or been realized.

For simiplicity, right now, the CHO effect during bud burst is expressed as a function of number of buds retained.

Before BudBurst

PreEventValue = 68

On BudBurst the value is set to:

3.9.2.1.1 PostEventValue

PostEventValue = minimum (BunchFun)

Where:

3.9.2.1.1.1 BunchFun

BunchFun = maximum (MetFun, zero)

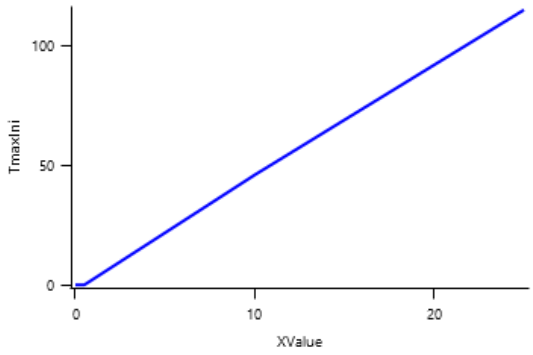
Where:

MetFun = Constant + TmaxIni + RadIni

Where:

Constant = -43.43

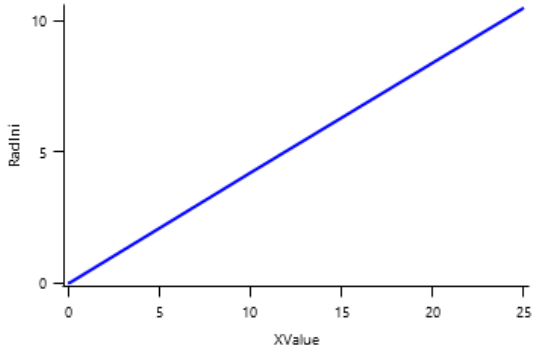
TmaxIni is calculated using linear interpolation



X	Y
0	0
0.5	0
10	46
20	92
25	115

XValue = [Berry].MetFactors.BUN_TmaxIni1

RadIni is calculated using linear interpolation



X	Y
0	0
0.1	0
10	4.2
20	8.4
25	10.5

XValue = [Berry].MetFactors.BUN_RadIni1

zero = 0

3.9.2.2 BerryNum

here we use a trigger function for bunches per shoot as the value of mean flowering temperature Y0 is updating itself during the growing season, which would cause inconvience if we directly correlates the bunches per shoot and mean flowering temperature Y0.

Numbers were guessed based on Eltom 2013 Fig. 7.7. y=0.0981*Ta + 0.35

Bunches per shoot is affected by the temperature and CHO during initiation, it is further affected by the CHO during bud burst. The later decided whether the potential initiated inflorescence can grow out successfully or been realized.

For simiplicity, right now, the CHO effect during bud burst is expressed as a function of number of buds retained.

Before Flowering

PreEventValue = 65

On Flowering the value is set to:

3.9.2.2.1 PostEventValue

PostEventValue = minimum (BerryNumFun, UpperLimit)

Where:

3.9.2.2.1.1 BerryNumFun

BerryNumFun = maximum (MetFun, zero)

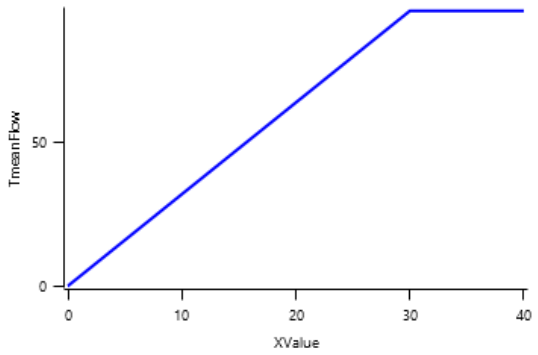
Where:

MetFun = Constant + TmeanFlow + RainTotFlow + TmaxIni + TmeanRainTot

Where:

Constant = -40

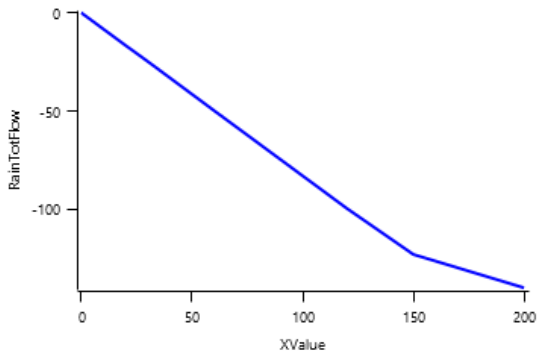
TmeanFlow is calculated using linear interpolation



X	Y
0	0
0.01	0
10	31.8
20	63.6
25	79.5
30	95.4
40	95.4

XValue = [Berry].MetFactors.BEN_TmeanFlow

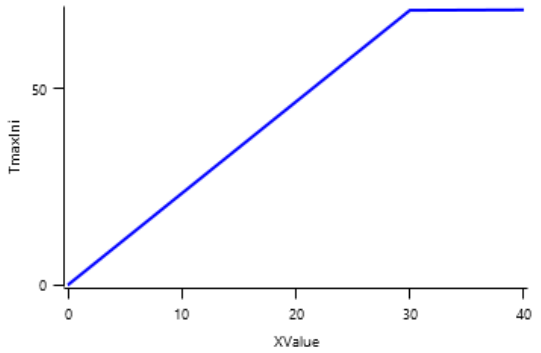
RainTotFlow is calculated using linear interpolation



X	Y
0	0
0.01	0
10	-8.3
20	-16.6
30	-24.75
60	-49.8
100	-83
120	-99.6
150	-123
200	-140

XValue = [Berry].MetFactors.BEN_RainTotFlow

TmaxIni is calculated using linear interpolation



X	Y
0	0
0.01	0
10	23.3
20	46.6
25	58.25
30	69.9
40	70

XValue = [Berry].MetFactors.BEN_TmaxIni1

TmeanRainTot = minimum (MultiplyFunction, Constant)

Where:

MultiplyFunction = Constant × [Berry].MetFactors.BEN_TmeanFlow × [Berry].MetFactors.BEN_RainTotFlow

Where:

Constant = 0.04812

Constant = 150

zero = 0

UpperLimit = 150

3.9.2.3 BerryMass

here we use a trigger function for bunches per shoot as the value of mean flowering temperature Y0 is updating itself during the growing season, which would cause inconvience if we directly correlates the bunches per shoot and mean flowering temperature Y0.

Numbers were guessed based on Eltom 2013 Fig. 7.7. y=0.0981*Ta + 0.35

Bunches per shoot is affected by the temperature and CHO during initiation, it is further affected by the CHO during bud burst. The later decided whether the potential initiated inflorescence can grow out successfully or been realized.

For simiplicity, right now, the CHO effect during bud burst is expressed as a function of number of buds retained.

Before Veraison

PreEventValue = 2.3

On Veraison the value is set to:

3.9.2.3.1 PostEventValue

PostEventValue = minimum (BerryMassFun, UpperLimit)

Where:

3.9.2.3.1.1 BerryMassFun

BerryMassFun = maximum (MetFun, zero)

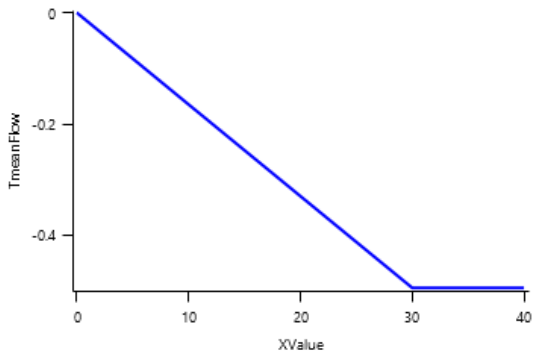
Where:

MetFun = Constant + TmeanFlow + RainTotFlow + RadFlow + RainTotVer + TmeanRainTotFlow + RadRainTotVer

Where:

Constant = 1.4

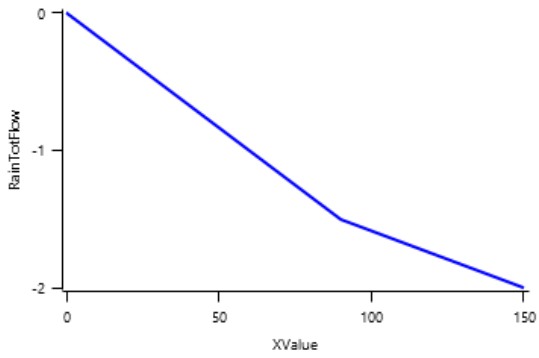
TmeanFlow is calculated using linear interpolation



X	Y
0	0
0.01	0
10	-0.165
20	-0.33
25	-0.4125
30	-0.495
40	-0.495

XValue = [Berry].MetFactors.BM_TmeanFlow

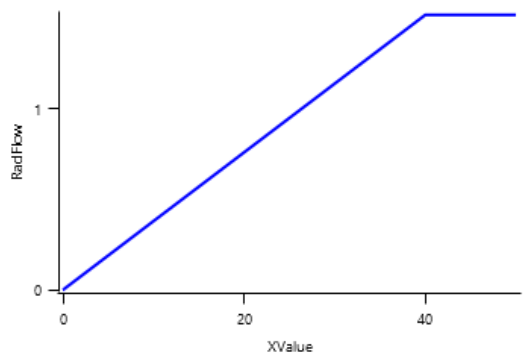
RainTotFlow is calculated using linear interpolation



X	Y
0	0
0.01	0
10	-0.167
20	-0.334
30	-0.501
60	-1.002
90	-1.503
150	-2

XValue = [Berry].MetFactors.BM_RainTotFlow

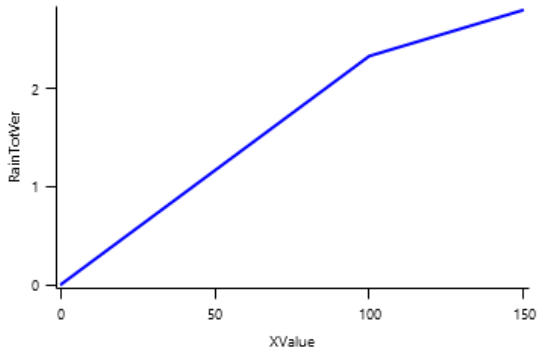
RadFlow is calculated using linear interpolation



X	Y
0	0
0.01	0
10	0.38
20	0.76
25	0.95
30	1.14
40	1.52
50	1.52

XValue = [Berry].MetFactors.BM_RadFlow

RainTotVer is calculated using linear interpolation



X	Y
0	0
0.01	0
10	0.233
20	0.466
30	0.699
60	1.398
90	2.097
100	2.33
150	2.8

XValue = [Berry].MetFactors.BM_RainTotVer

TmeanRainTotFlow = minimum (MultiplyFunction, Constant)

Where:

MultiplyFunction = Constant × [Berry].MetFactors.BM_TmeanFlow × [Berry].MetFactors.BM_RainTotFlow

Where:

Constant = 0.000987

Constant = 2

RadRainTotVer = maximum (MultiplyFunction, Constant)

Where:

MultiplyFunction = Constant × [Berry].MetFactors.BM_RadFlow × [Berry].MetFactors.BM_RainTotVer

Where:

Constant = -0.000898

Constant = -2.4246

zero = 0

UpperLimit = 2.5

3.9.3 NumberFunction

NumberFunction = [Berry].YieldComponent.BunchesPerVine × [Berry].YieldComponent.BerryNum × [Grapevine].Population

3.9.4 SingleBerryFW

SingleBerryFW = BetaGrowthFunction

Where:

3.9.4.1 BetaGrowthFunction

a beta growth function of the form y = Ymax * (1 + (te - t)/(te-tm))* (t/te)^(te/(te-tm))

Ymax = [Berry].YieldComponent.BerryMass

XValue = [Berry].ThermalTimeAfterFlowering

3.9.5 TotalBerryFW

TotalBerryFW = [Berry].NumberFunction × BetaGrowthFunction

Where:

3.9.5.1 BetaGrowthFunction

a beta growth function of the form y = Ymax * (1 + (te - t)/(te-tm))* (t/te)^(te/(te-tm))

Ymax = [Berry].YieldComponent.BerryMass

XValue = [Berry].ThermalTimeAfterFlowering

3.9.6 SingleBerryDW

SingleBerryDW = BetaGrowthFunction

Where:

3.9.6.1 BetaGrowthFunction

a beta growth function of the form y = Ymax * (1 + (te - t)/(te-tm))* (t/te)^(te/(te-tm))

Ymax = 0.5529 (g)

XValue = [Berry].ThermalTimeAfterFlowering

3.9.7 WaterContent

WaterContent = Constant - AfterFlowering

Where:

Constant = 1

3.9.7.1 AfterFlowering

0 between Flowering and LeafFall and a value of zero outside of this period

3.9.8 Brix

3.9.8.1 BeforeVeraison

The value of Brix from Flowering to Veraison is calculated as follows:

Constant = 5

3.9.8.2 AfterVeraison

The value of Brix from Veraison to LeafFall is calculated as follows:

3.9.8.2.1 DivideFunction

DivideFunction = MaximumFunction / constant

Where:

3.9.8.2.1.1 MaximumFunction

MaximumFunction = maximum (SubtractFunction, zero)

Where:

SubtractFunction = constant - [Berry].WaterContent

Where:

constant = 0.944

zero = 0

constant = 0.0082

Brix has a value of zero for phases not specified above

3.9.9 TitratableAcid

3.9.9.1 BeforeVeraison

The value of TitratableAcid from Flowering to Veraison is calculated as follows:

Constant = 0

3.9.9.2 AfterVeraison

The value of TitratableAcid from Veraison to LeafFall is calculated as follows:

3.9.9.3 TitratableAcid

An exponential function

XValue = [Berry].ThermalTimeAfterVeraison

TitratableAcid has a value of zero for phases not specified above

3.9.10 DMDemandFunction

DMDemandFunction = DeltaFunction

Where:

DeltaFunction is the daily differential of

3.9.10.1 Integral

Integral = [Berry].NumberFunction × BetaGrowthFunction

Where:

3.9.10.1.1 BetaGrowthFunction

a beta growth function of the form y = Ymax * (1 + (te - t)/(te-tm))* (t/te)^(te/(te-tm))

Ymax = 0.5529 (g)

XValue = [Berry].ThermalTimeAfterFlowering

3.9.11 ThermalTimeAfterVeraison

ThermalTimeAfterVeraison is a daily accumulation of the values of functions listed below between the Veraison and LeafFall stages. Function values added to the accumulate total each day are:

ThermalTime = [Phenology].ThermalTime

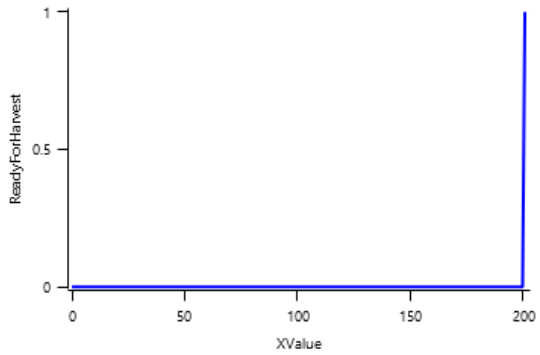
3.9.12 ThermalTimeAfterFlowering

ThermalTimeAfterFlowering is a daily accumulation of the values of functions listed below between the Flowering and LeafFall stages. Function values added to the accumulate total each day are:

ThermalTime = [Phenology].ThermalTime

3.9.13 ReadyForHarvest

ReadyForHarvest is calculated using linear interpolation



X	Y
0	0
200	0
201	1

XValue = [Berry].ThermalTimeAfterVeraison

3.9.14 MaxNConcDailyGrowth

MaxNConcDailyGrowth = 0.01 (g/g)

3.9.15 NFillingRate

NFillingRate = 0 (g/m2/d)

3.9.16 MinimumNConc

MinimumNConc = 0 (g/g)

3.9.17 MaximumNConc

MaximumNConc = 0.0001 (g/g)

3.9.18 BiomassRemovalDefaults

This organ will respond to certain management actions by either removing some of its biomass from the system or transferring some of its biomass to the soil surface residues. The following table describes the default proportions of live and dead biomass that are transferred out of the simulation using "Removed" or to soil surface residue using "To Residue" for a range of management actions. The total percentage removed for live or dead must not exceed 100%. The difference between the total and 100% gives the biomass remaining on the plant. These can be changed during a simulation using a manager script.

Method	% Live Removed	% Dead Removed	% Live To Residue	% Dead To Residue
Harvest	0	0	0	0
Cut	0	0	0	0
Prune	0	0	0	0
Graze	0	0	0	0
Thin	0	0	0	0

3.9.19 MaximumPotentialGrainSize

MaximumPotentialGrainSize = 2.5 (g fresh or dry)

3.9.20 DMConversionEfficiency

DMConversionEfficiency = 1

3.9.21 RemobilisationCost

RemobilisationCost = 0

3.9.22 CarbonConcentration

CarbonConcentration = 0.4

3.10 Root

The generic root model calculates root growth in terms of rooting depth, biomass accumulation and subsequent root length density in each soil layer.

Root Growth

Roots grow downwards through the soil profile, with initial depth determined by sowing depth and the growth rate determined by RootFrontVelocity. The RootFrontVelocity is modified by multiplying it by the soil's XF value; which represents any resistance posed by the soil to root extension. Root depth is also constrained by a maximum root depth.

Root length growth is calculated using the daily DM partitioned to roots and a specific root length. Root proliferation in layers is calculated using an approach similar to the generalised equimarginal criterion used in economics. The uptake of water and N per unit root length is used to partition new root material into layers of higher 'return on investment'.

Dry Matter Demands

A daily DM demand is provided to the organ arbitrator and a DM supply returned. By default, 100% of the dry matter (DM) demanded from the root is structural. The daily loss of roots is calculated using a SenescenceRate function. All senesced material is automatically detached and added to the soil FOM.

Nitrogen Demands

The daily structural N demand from root is the product of total DM demand and the minimum N concentration. Any N above this is considered Storage and can be used for retranslocation and/or reallocation as the respective factors are set to values other then zero.

Nitrogen Uptake

Potential N uptake by the root system is calculated for each soil layer (i) that the roots have extended into. In each layer potential uptake is calculated as the product of the mineral nitrogen in the layer, a factor controlling the rate of extraction (kNO3 or kNH4), the concentration of N form (ppm), and a soil moisture factor (NUptakeSWFactor) which typically decreases as the soil dries.

$NO3\ uptake = NO3_i \times kNO3 \times NO3_{ppm,\ i} \times NUptakeSWFactor$

$NH4\ uptake = NH4_i \times kNH4 \times NH4_{ppm,\ i} \times NUptakeSWFactor$

Nitrogen uptake demand is limited to the maximum daily potential uptake (MaxDailyNUptake) and the plants N demand. The demand for soil N is then passed to the soil arbitrator which determines how much of the N uptake demand each plant instance will be allowed to take up.

Water Uptake

Potential water uptake by the root system is calculated for each soil layer that the roots have extended into. In each layer potential uptake is calculated as the product of the available water in the layer (water above LL limit) and a factor controlling the rate of extraction (KL). The values of both LL and KL are set in the soil interface and KL may be further modified by the crop via the KLModifier function.

$SW\ uptake = (SW_i - LL_i) \times KL_i \times KLModifier$

3.10.1 NitrogenDemandSwitch

A value of 1 is returned if phenology is between BudBurst and LeafFall phases, otherwise a value of 0 is returned.

3.10.2 MinimumNConc

MinimumNConc = 0.01 (g/g)

3.10.3 MaximumNConc

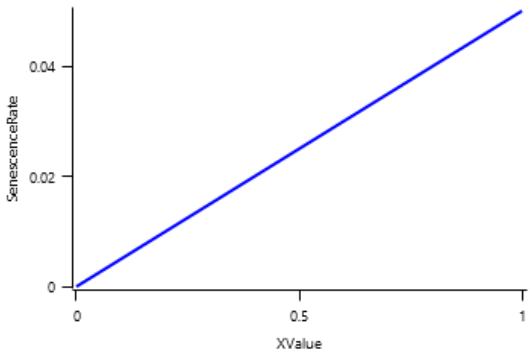
MaximumNConc = 0.01 (g/g)

3.10.4 MaximumRootDepth

MaximumRootDepth = 1000000 (mm)

3.10.5 SenescenceRate

SenescenceRate is calculated using linear interpolation



X	Y
0	0
1	0.05

XValue = [Leaf].FractionDied

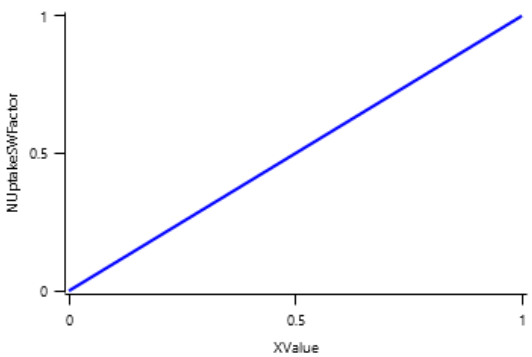
3.10.6 BiomassRemovalDefaults

This organ will respond to certain management actions by either removing some of its biomass from the system or transferring some of its biomass to the soil surface residues. The following table describes the default proportions of live and dead biomass that are transferred out of the simulation using "Removed" or to soil surface residue using "To Residue" for a range of management actions. The total percentage removed for live or dead must not exceed 100%. The difference between the total and 100% gives the biomass remaining on the plant. These can be changed during a simulation using a manager script.

Method	% Live Removed	% Dead Removed	% Live To Residue	% Dead To Residue
Harvest	0	0	0	0
Cut	0	0	0	0
Prune	0	0	0	0
Graze	0	0	0	0

3.10.7 NUptakeSWFactor

NUptakeSWFactor is calculated using linear interpolation



X	Y
0	0
1	1

XValue = [Root].RWC

3.10.8 InitialDM

InitialDM = 2500 (g/plant)

3.10.9 SpecificRootLength

SpecificRootLength = 40 (m/g)

3.10.10 RootFrontVelocity

RootFrontVelocity = 10 (mm/d)

3.10.11 KNO3

KNO3 = 0.02

3.10.12 KNH4

KNH4 = 0.003

3.10.13 MaxDailyNUptake

MaxDailyNUptake = 6

3.10.14 DMConversionEfficiency

DMConversionEfficiency = 1

3.10.15 MaintenanceRespirationFunction

MaintenanceRespirationFunction = 0.002

3.10.16 RemobilisationCost

RemobilisationCost = 0

3.10.17 CarbonConcentration

CarbonConcentration = 0.4

3.10.18 DMRetranslocationFactor

DMRetranslocationFactor = 0.1

3.10.19 DMDemands

This is the collection of functions for calculating the demands for each of the biomass pools (Structural, Metabolic, and Storage).

3.10.19.1 Structural

Structural = DMDemandFunction × StructuralFraction

Where:

3.10.19.1.1 DMDemandFunction

DMDemandFunction = PartitionFraction [Arbitrator].DM.TotalFixationSupply

Where:

PartitionFraction = 0.2

StructuralFraction = 0.8 (g/g)

3.10.19.2 Metabolic

Metabolic = 0 (g/m2)

3.10.19.3 Storage

The partitioning of daily growth to storage biomass is based on a storage fraction.

3.10.19.3.1 StorageFraction

StorageFraction = 1 - [Root].DMDemands.Structural.StructuralFraction

3.10.20 NDemands

This is the collection of functions for calculating the demands for each of the biomass pools (Structural, Metabolic, and Storage).

3.10.20.1 Structural

Structural = [Root].minimumNconc × [Root].potentialDMAllocation.Structural

3.10.20.2 Metabolic

Metabolic = MetabolicNconc × [Root].potentialDMAllocation.Structural

Where:

3.10.20.2.1 MetabolicNconc

MetabolicNconc = [Root].criticalNConc - [Root].minimumNconc

3.10.20.3 Storage

The partitioning of daily growth to storage biomass attempts to bring the organ's N content to the maximum concentration.

Storage = [Root].maximumNconc × ([Root].Live.Wt + potentialAllocationWt) - [Root].Live.N

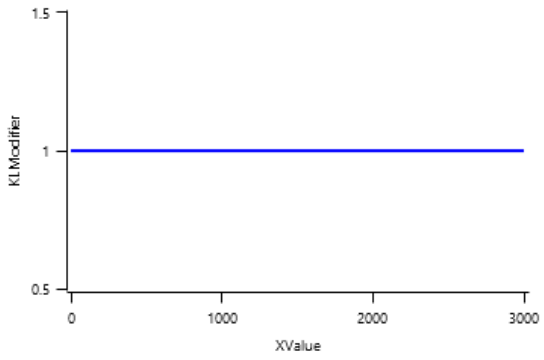
The demand for storage N is further reduced by a factor specified by the [Root].NitrogenDemandSwitch.

CriticalNConc = [Root].MinimumNConc

3.10.21 KLModifier

This is important in SLURP as it is set for each species to represent their differences in rooting patterns between crop species

KLModifier is calculated using linear interpolation



X	Y
0	1
300	1
600	1
900	1
1200	1
1500	1
1800	1
2100	1
2400	1
2700	1
3000	1

XValue = [Root].LayerMidPointDepth

3.11 Arbitrator

The Arbitrator class determines the allocation of dry matter (DM) and Nitrogen between each of the organs in the crop model. Each organ can have up to three different pools of biomass:

- **Structural biomass** which is essential for growth and remains within the organ once it is allocated there.
- **Metabolic biomass** which generally remains within an organ but is able to be re-allocated when the organ senesces and may be retranslocated when demand is high relative to supply.

- **Storage biomass** which is partitioned to organs when supply is high relative to demand and is available for retranslocation to other organs whenever supply from uptake, fixation, or re-allocation is lower than demand.

The process followed for biomass arbitration is shown in Figure 3. Arbitration calculations are triggered by a series of events (shown below) that are raised every day. For these calculations, at each step the Arbitrator exchange information with each organ, so the basic computations of demand and supply are done at the organ level, using their specific parameters.

1. **doPotentialPlantGrowth**. When this event occurs, each organ class executes code to determine their potential growth, biomass supplies and demands. In addition to demands for structural, non-structural and metabolic biomass (DM and N) each organ may have the following biomass supplies:
2. **Fixation supply**. From photosynthesis (DM) or symbiotic fixation (N)
3. **Uptake supply**. Typically uptake of N from the soil by the roots but could also be uptake by other organs (eg foliage application of N).
4. **Retranslocation supply**. Storage biomass that may be moved from organs to meet demands of other organs.
5. **Reallocation supply**. Biomass that can be moved from senescing organs to meet the demands of other organs.
6. **doPotentialPlantPartitioning.** On this event the Arbitrator first executes the DoDMSetup() method to gather the DM supplies and demands from each organ, these values are computed at the organ level. It then executes the DoPotentialDMAllocation() method which works out how much biomass each organ would be allocated assuming N supply is not limiting and sends these allocations to the organs. Each organ then uses their potential DM allocation to determine their N demand (how much N is needed to produce that much DM) and the arbitrator calls DoNSetup() to gather the N supplies and demands from each organ and begin N arbitration. Firstly DoNReallocation() is called to redistribute N that the plant has available from senescing organs. After this step any unmet N demand is considered as plant demand for N uptake from the soil (N Uptake Demand).
7. **doNutrientArbitration.** When this event occurs, the soil arbitrator gets the N uptake demands from each plant (where multiple plants are growing in competition) and their potential uptake from the soil and determines how much of their demand that the soil is able to provide. This value is then passed back to each plant instance as their Nuptake and doNUptakeAllocation() is called to distribute this N between organs.
8. **doActualPlantPartitioning.** On this event the arbitrator call DoNRetranslocation() and DoNFixation() to satisfy any unmet N demands from these sources. Finally, DoActualDMAllocation is called where DM allocations to each organ are reduced if the N allocation is insufficient to achieve the organs minimum N concentration and final allocations are sent to organs.

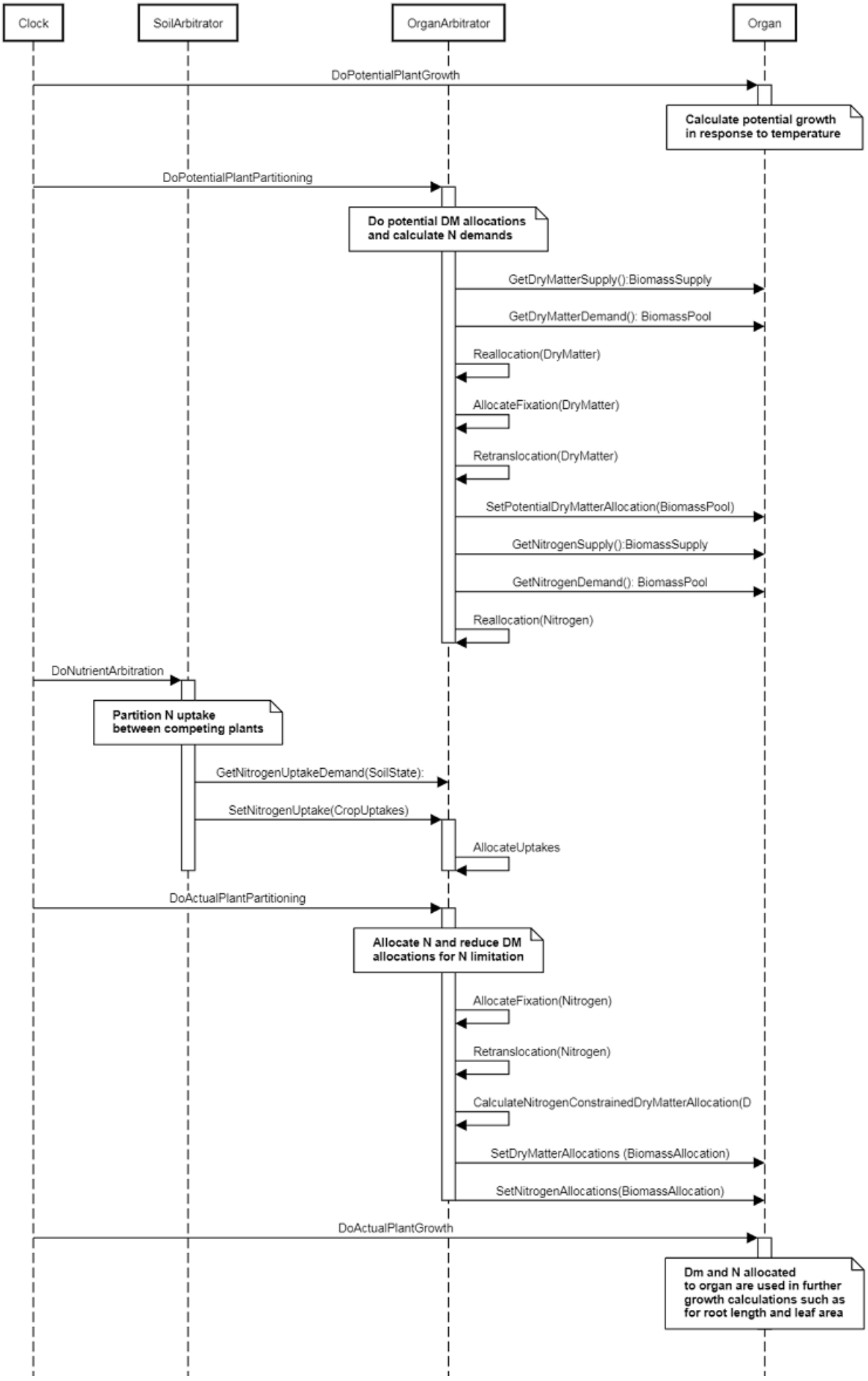


Figure 3: Schematic showing the procedure for arbitration of biomass partitioning. Pink boxes represent events that occur every day and their numbering shows the order of calculations. Blue boxes represent the methods that are called when these events occur. Orange boxes contain properties that make up the organ/arbitrator interface. Green boxes are organ specific properties.

A replacements model

The Grapevine model is constructed from the following list of software components. Details of the implementation and model parameterisation are provided in the following sections.

List of Plant Model Components.

Component Name	Component Type
Phenology	Models.PMF.Phen.Phenology
Structure	Models.PMF.Struct.Structure
Leaf	Models.PMF.Organs.Leaf
Shoot	Models.PMF.Organs.GenericOrgan
Cordon	Models.PMF.Organs.GenericOrgan
Trunk	Models.PMF.Organs.GenericOrgan
Berry	Models.PMF.Organs.ReproductiveOrgan
Root	Models.PMF.Organs.Root
Arbitrator	Models.PMF.OrganArbitrator

3.11.1 SauvignonBlanc2C

This cultivar is defined by overriding some of the base parameters of the plant model.

SauvignonBlanc2C makes the following changes:

3.11.2 SauvignonBlanc4C

This cultivar is defined by overriding some of the base parameters of the plant model.

SauvignonBlanc4C makes the following changes:

3.11.3 Phenology

This model simulates the development of the crop through successive developmental *phases*. Each phase is bound by distinct growth *stages*. Phases often require a target to be reached to signal movement to the next phase. Differences between cultivars are specified by changing the values of the default parameters shown below.

Dormancy sequence: Paradormancy (Apical dominance), Endodormancy(Chilling requirement), Ecodormancy (forcing unit) Sarvas (1974) separates bud dormancy in two periods: the "rest" period is defined as the period when buds are dormant due to physiological conditions (endodormancy), and the "quiescence" period is when buds remain dormant due to unfavourable environmental conditions (ecodormancy). The current phenological models assume the dormancy starts at March 1st, and budburst is regulated by temperature. Ecodormancy is induced by a period with chilling temperatures, and then followed by a period with forcing temperatures till bud burst (Chuine et al., 2003).

We then simulated the time of flowering, fruitSet,veraison, leaf fall. After one cycle, bud goes to dormancy at March 1st.

Paradormancy, which is the suspension of growth caused by factors outside the meristem but within the plant. It is typically an influence of one organ over another, and includes an apical bud preventing outgrowth of a lower bud, which relates to apical dominance (see Martin 1987 for review).

List of stages and phases used in the simulation of crop phenological development

Phase Number	Phase Name	Initial Stage	Final Stage
1	EndoDormancy	Endodormancy	Ecodormancy

Phase Number	Phase Name	Initial Stage	Final Stage
2	Budding	Ecodormancy	BudBurst
3	Flowering	BudBurst	Flowering
4	BerryDevelopment	Flowering	Veraison
5	LeafDeathPhase	Veraison	LeafFall
6	GotoPhase	LeafFall	EndoDormancy

3.11.3.1 Phenological Phases

3.11.3.1.1 EndoDormancy Phase

This *phase* goes from Endodormancy to Ecodormancy. It uses a *Target* to determine the duration between development *Stages*. Daily *progress* is accumulated until the *Target* is met and remaining fraction of the day is forwarded to the next phase.

The *Target* and the daily *Progression* toward Ecodormancy are described as follow:

3.11.3.1.1.1 Target

Target = BaseTarget - [Phenology].AccChillBefPrune

Where:

BaseTarget = 8 (days)

3.11.3.1.2 Budding Phase

This *phase* goes from Ecodormancy to BudBurst. It uses a *Target* to determine the duration between development *Stages*. Daily *progress* is accumulated until the *Target* is met and remaining fraction of the day is forwarded to the next phase.

The *Target* and the daily *Progression* toward BudBurst are described as follow:

3.11.3.1.2.1 Target

Target = 79 (days)

3.11.3.1.2.2 Progression

Progression =

3.11.3.1.3 Flowering Phase

This *phase* goes from BudBurst to Flowering. It uses a *Target* to determine the duration between development *Stages*. Daily *progress* is accumulated until the *Target* is met and remaining fraction of the day is forwarded to the next phase.

The *Target* and the daily *Progression* toward Flowering are described as follow:

3.11.3.1.3.1 Target

Target = 27

Progression = [Phenology].ThermalTime

3.11.3.1.4 BerryDevelopment Phase

This *phase* goes from Flowering to Veraison. It uses a *Target* to determine the duration between development *Stages*. Daily *progress* is accumulated until the *Target* is met and remaining fraction of the day is forwarded to the next phase.

The *Target* and the daily *Progression* toward Veraison are described as follow:

3.11.3.1.4.1 Target

Target = Base × LAIAdjustment

Where:

Base = 40

Progression = [Phenology].ThermalTime

It proceeds until the last leaf on the main-stem has fully senessced. Therefore its duration depends on the number of main-stem leaves that are produced and the rate at which they seness following final leaf appearance.

ThermalTime = [Phenology].ThermalTime

3.11.3.1.5 GotoPhase Phase

This is a special phase, at LeafFall the phenology is reset to the EndoDormancy phase.

3.11.3.2 ThermalTime

ThermalTime =

3.11.3.3 BudBurstDOY

A function is used to provide flowering date as days after sowing(DAS).

3.11.3.4 FloweringDOY

A function is used to provide flowering date as days after sowing(DAS).

3.11.3.5 VeraisonDOY

A function is used to provide flowering date as days after sowing(DAS).

3.11.3.6 CurrentSeason

A function is used to capture the year of the current season as in the south hemisphere the growing season extends to the second year.

3.11.3.7 AccChillBefPrune

AccChillBefPrune is a daily accumulation of the values of functions listed below between the Veraison and LeafFall stages. Function values added to the accumulate total each day are:

3.11.3.7.1 LessThanFunction

IF [Weather].DaysSinceWinterSolstice < PruningTime THEN

ChillingUnit = [Phenology].EndoDormancy.Progression

ELSE

Zero = 0

3.11.4 Structure

The structure model simulates morphological development of the plant to inform the Leaf class when and how many leaves appear and to provide a hight estimate for use in calculating potential transpiration.

3.11.4.1 Plant and Main-Stem Population

The *Plant.Population* is set at sowing with information sent from a manager script in the Sow method. The *PrimaryBudNumber* is also sent with the Sow method and the main-stem population (*MainStemPopn*) for the crop is calculated as: *MainStemPopn* = *Plant.Population* x *PrimaryBudNumber* Primary bud number is > 1 for crops like potato and grape vine where there are more than one main-stem per plant

3.11.4.2 Main-Stem leaf appearance

Each day the number of main-stem leaf tips appeared (*LeafTipsAppeared*) is calculated as: *LeafTipsAppeared*

+= *DeltaTips* Where *DeltaTips* is calculated as: *DeltaTips* = *ThermalTime/Phyllochron* Where *Phyllochron* is the thermal time duration between the appearance of leaf tipx given by:

3.11.4.2.1 Phyllochron

0 between BudBurst and Veraison and a value of zero outside of this period

and *ThermalTime* is given by:

3.11.4.2.2 ThermalTime

ThermalTime =

LeafTipsAppeared continues to increase until *FinalLeafNumber* is reached where *FinalLeafNumber* is calculated as:

3.11.4.2.3 FinalLeafNumber

FinalLeafNumber = 25 (number)

3.11.4.3 Branching and Branch Mortality

The total population of stems (*TotalStemPopn*) is calculated as: *TotalStemPopn* = *MainStemPopn* + *NewBranches* - *NewlyDeadBranches* Where *NewBranches* = *MainStemPopn* x *BranchingRate* and *BranchingRate* is given by:

3.11.4.3.1 BranchingRate

BranchingRate = Potential_Branching_Rate × LinearInterpolationFunction

Where:

3.11.4.3.1.1 LinearInterpolationFunction

NewlyDeadBranches is calcualted as: *NewlyDeadBranches* = (*TotalStemPopn* - *MainStemPopn*) x *BranchMortality* where *BranchMortality* is given by:

3.11.4.3.2 BranchMortality

BranchMortality = 0

3.11.4.4 Height

The Height of the crop is calculated by the *HeightModel*:

3.11.4.5 MainStemPrimordialnitiationRate

0 between BudBurst and Veraison and a value of zero outside of this period

3.11.4.6 DroughtInducedBranchMortality

DroughtInducedBranchMortality = 0

3.11.4.7 StemSenescenceAge

StemSenescenceAge = 0

3.11.4.8 BudNumber

Sets the number of buds on each mains stem to the valud of it child on the BudBurst

3.11.4.8.1 FractionOfBudBurst

FractionOfBudBurst = BudNumberEffect × CordonDiameterEffect

Where:

3.11.4.8.1.1 BudNumberEffect

a sigmoid function of the form y = Xmax * 1 / 1 + $e^{-(XValue - Xo) / b}$

XValue = [Structure].PrimaryBudNo

Ymax = 1.4

Xo = 100

b = -54.8977

3.11.5 Leaf

The leaves are modelled as a set of leaf cohorts and the properties of each of these cohorts are summed to give overall values for the leaf organ. A cohort represents all the leaves of a given main- stem node position including all of the branch leaves appearing at the same time as the given main-stem leaf (Lawless et al., 2005). The number of leaves in each cohort is the product of the number of plants per m^2 and the number of branches per plant. The *Structure* class models the appearance of main-stem leaves and branches. Once cohorts are initiated the *Leaf* class models the area and biomass dynamics of each. It is assumed all the leaves in each cohort have the same size and biomass properties. The modelling of the status and function of individual cohorts is delegated to *LeafCohort* classes.

3.11.5.1 Dry Matter Fixation

The most important DM supply from leaf is the photosynthetic fixation supply. Radiation interception is calculated from LAI using an extinction coefficient of:

3.11.5.1.1 ExtinctionCoeff

ExtinctionCoeff = Constant

Where:

Constant = 0.5

3.11.5.2 Photosynthesis

Biomass fixation is modelled as the product of intercepted radiation and its conversion efficiency, the radiation use efficiency (RUE) (Monteith et al., 1977). This approach simulates net photosynthesis rather than providing separate estimates of growth and respiration. The potential photosynthesis calculated using RUE is then adjusted according to stress factors, these account for plant nutrition (FN), air temperature (FT), vapour pressure deficit (FVPD), water supply (FW) and atmospheric CO_2 concentration (FCO2). NOTE: RUE in this model is expressed as g/MJ for a whole plant basis, including both above and below ground growth.

3.11.5.2.1 RUE

The value of RUE from BudBurst to Flowering is calculated as follows:

Constant = 1.2 (g/MJ)

The value of RUE from Flowering to Veraison is calculated as follows:

Constant = 1.05 (g/MJ)

The value of RUE from Veraison to LeafFall is calculated as follows:

Constant = 1.5 (g/MJ)

RUE has a value of zero for phases not specified above

3.11.5.2.2 FCO2

This model calculates the CO_2 impact on RUE using the approach of Reyenga et al., 1999.

3.11.5.2.3 FVPD

RadnInt = [Leaf].RadIntTot

ThermalTime = [Structure].ThermalTime

Area = 1000

Area = 0

Area = 0

3.11.5.3 Potential Leaf Area index

Leaf area index is calculated as the sum of the area of each cohort of leaves The appearance of a new cohort of leaves occurs each time Structure.LeafTipsAppeared increases by one. From tip appearance the area of each cohort will increase for a certian number of degree days defined by the *GrowthDuration*

3.11.5.3.1 GrowthDuration

If no stress occurs the leaves will reach a Maximum area (*MaxArea*) at the end of the *GrowthDuration*. The *MaxArea* is defined by:

3.11.5.3.2 MaxArea

In the absence of stress the leaf will remain at *MaxArea* for a number of degree days set by the *LagDuration* and then area will senesce to zero at the end of the *SenescenceDuration*

3.11.5.3.3 SenescenceDuration

Mutual shading can cause premature senescence of cohorts if the leaf area above them becomes too great. Each cohort models the proportion of its area that is lost to shade induced senescence each day as:

3.11.5.4 Stress effects on Leaf Area Index

Stress reduces leaf area in a number of ways. Firstly, stress occuring prior to the appearance of the cohort can reduce cell division, so reducing the maximum leaf size. Leaf captures this by multiplying the *MaxSize* of each cohort by a *CellDivisionStress* factor which is calculated as:

3.11.5.4.1 CellDivisionStress

CellDivisionStress = 1 (0-1)

Leaf.FN quantifys the N stress status of the plant and represents the concentration of metabolic N relative the maximum potentil metabolic N content of the leaf calculated as (*Leaf.NConc - MinimumNConc*)/(*CriticalNConc - MinimumNConc*).

Leaf.FW quantifies water stress and is calculated as *Leaf.Transpiration*/*Leaf.WaterDemand*, where *Leaf.Transpiration* is the minimum of *Leaf.WaterDemand* and *Root.WaterUptake*

Stress during the *GrowthDuration* of the cohort reduces the size increase of the cohort by multiplying the potential increase by a *ExpansionStress* factor:

3.11.5.4.2 ExpansionStress

Stresses can also acellerate the onset and rate of senescence in a number of ways. Nitrogen shortage will cause N to be retranslocated out of lower order leaves to support the expansion of higher order leaves and other organs When this happens the lower order cohorts will have their area reduced in proportion to the amount of N that is remobilised out of them.

Water stress hastens senescence by increasing the rate of thermal time accumulation in the lag and senescence phases. This is done by multiplying thermal time accumulation by *DroughtInducedLagAcceleration* and *DroughtInducedSenescenceAcceleration* factors, respectively:

3.11.5.4.3 DroughtInducedLagAcceleration

DroughtInducedLagAcceleration = 1

3.11.5.4.4 DroughtInducedSenAcceleration

DroughtInducedSenAcceleration = 1

3.11.5.5 Dry matter Demand

Leaf calculates the DM demand from each cohort as a function of the potential size increment (DeltaPotentialArea) an specific leaf area bounds. Under non stressed conditions the demand for non-storage DM is calculated as *DeltaPotentialArea* divided by the mean of *SpecificLeafAreaMax* and *SpecificLeafAreaMin*. Under stressed conditions it is calculated as *DeltaWaterConstrainedArea* divided by *SpecificLeafAreaMin*.

3.11.5.5.1 SpecificLeafAreaMin

SpecificLeafAreaMin = 20000 (mm2/g)

Non-storage DM Demand is then seperated into structural and metabolic DM demands using the *StructuralFraction*:

3.11.5.5.2 StructuralFraction

StructuralFraction = 0.8 (percentage)

The storage DM demand is calculated from the sum of metabolic and structural DM (including todays demands) multiplied by a *NonStructuralFraction*:

Unknown child name: NonStructuralFraction

3.11.5.6 Nitrogen Demand

Leaf calculates the N demand from each cohort as a function of the potential DM increment and N concentration bounds. Structural N demand = *PotentialStructuralDMAllocation* * *MinimumNConc* where:

3.11.5.6.1 MinimumNConc

MinimumNConc = 0.01 (g/g)

Metabolic N demand is calculated as *PotentialMetabolicDMAllocation* * (*CriticalNConc* - *MinimumNConc*) where:

3.11.5.6.2 CriticalNConc

CriticalNConc = 0.03 (g/g)

Storage N demand is calculated as the sum of metabolic and structural wt (including todays demands) multiplied by *LuxaryNconc* (*MaximumNConc* - *CriticalNConc*) less the amount of storage N already present. *MaximumNConc* is given by:

3.11.5.6.3 MaximumNConc

MaximumNConc = 0.05 (g/g)

3.11.5.7 Drymatter supply

In additon to photosynthesis, the leaf can also supply DM by reallocation of senescing DM and retranslocation of storgage DM: Reallocation supply is a proportion of the metabolic and non-structural DM that would be senesced each day where the proportion is set by:

3.11.5.7.1 DMReallocationFactor

DMReallocationFactor = 0.1 (percentage)

Retranslocation supply is calculated as a proportion of the amount of storage DM in each cohort where the proportion is set by :

3.11.5.7.2 DMRetranslocationFactor

DMRetranslocationFactor = 0 (percentage)

3.11.5.8 Nitrogen supply

Nitrogen supply from the leaf comes from the reallocation of metabolic and storage N in senescing material and the retranslocation of metabolic and storage N. Reallocation supply is a proportion of the Metabolic and Storage DM that would be senesced each day where the proportion is set by:

3.11.5.8.1 NReallocationFactor

NReallocationFactor = 0 (percentage)

Retranslocation supply is calculated as a proportion of the amount of storage and metabolic N in each cohort where the proportion is set by :

3.11.5.8.2 NRetranslocationFactor

NRetranslocationFactor = 0 (percentage)

3.11.5.8.3 DetachmentLagDuration

DetachmentLagDuration = 10 (thermalDay)

3.11.5.8.4 DetachmentDuration

DetachmentDuration = 200 (degreeDay)

3.11.5.8.5 InitialNConc

InitialNConc = 0.04 (g/g)

3.11.5.8.6 StorageFraction

StorageFraction = 0.2 (percentage)

3.11.5.8.7 SenessingLeafRelativeSize

SenessingLeafRelativeSize = 1 (0-1)

3.11.5.8.8 LeafSizeShapeParameter

LeafSizeShapeParameter = 0.01 (parameter controls the logistic growth of the leaf in growth stage)

3.11.5.8.9 MaintenanceRespirationFunction

MaintenanceRespirationFunction = 0 (g/g)

3.11.5.8.10 LagDurationAgeMultiplier

LagDurationAgeMultiplier = 1 1 1

3.11.5.8.11 SenescenceDurationAgeMultiplier

SenescenceDurationAgeMultiplier = 1 1 1

3.11.5.8.12 LeafSizeAgeMultiplier

LeafSizeAgeMultiplier = 1 1 1 1 1 1 1 1 1 1 1

3.11.5.8.13 RemobilisationCost

RemobilisationCost = 0

3.11.5.8.14 CarbonConcentration

CarbonConcentration = 0.4

3.11.5.9 FRGRFunction

FRGRFunction = minimum (RUE_FT, Others)

Where:

RUE_FT = [Leaf].Photosynthesis.FT

3.11.5.9.1 Others

Others = minimum (RUE_FN, RUE_FVPD)

Where:

RUE_FN = [Leaf].Photosynthesis.FN

RUE_FVPD = [Leaf].Photosynthesis.FVPD

3.11.5.10 Total Biomass

This is a composite biomass class, representing the sum of 1 or more biomass objects.

Total summarises the following biomass objects:

- [Leaf].Live
- [Leaf].Dead

3.11.5.11 BiomassRemovalDefaults

This organ will respond to certain management actions by either removing some of its biomass from the system or transferring some of its biomass to the soil surface residues. The following table describes the default proportions of live and dead biomass that are transferred out of the simulation using "Removed" or to soil surface residue using "To Residue" for a range of management actions. The total percentage removed for live or dead must not exceed 100%. The difference between the total and 100% gives the biomass remaining on the plant. These can be changed during a simulation using a manager script.

Method	% Live Removed	% Dead Removed	% Live To Residue	% Dead To Residue
Harvest	0	0	0	0
Cut	0	0	0	0
Prune	0	0	0	0
Graze	0	0	0	0

3.11.5.12 FrostFraction

0 between Veraison and LeafFall and a value of zero outside of this period

3.11.5.13 DMConversionEfficiency

DMConversionEfficiency = 1

3.11.5.14 RemobilisationCost

RemobilisationCost = 0

3.11.5.15 CarbonConcentration

CarbonConcentration = 0.4

3.11.5.16 StructuralFraction

StructuralFraction = 0.8 (g/g)

3.11.5.17 StomatalConductanceCO2Modifier

StomatalConductanceCO2Modifier = 1

3.11.6 Shoot

This organ is simulated using a GenericOrgan type. It is parameterised to calculate the growth, senescence, and detachment of any organ that does not have specific functions.

3.11.6.1 Dry Matter Demand

The dry matter demand for the organ is calculated as defined in DMDemands, based on the DMDemandFunction and partition fractions for each biomass pool.

3.11.6.1.1 DMDemands

This is the collection of functions for calculating the demands for each of the biomass pools (Structural, Metabolic, and Storage).

3.11.6.1.1.1 Structural

Structural = DMDemandFunction × StructuralFraction

Where:

DMDemandFunction = PartitionFraction [Arbitrator].DM.TotalFixationSupply

Where:

PartitionFraction = 0.2

StructuralFraction = 1 (g/g)

3.11.6.1.1.2 Metabolic

Metabolic = 0 (g/m2)

3.11.6.1.1.3 Storage

The partitioning of daily growth to storage biomass is based on a storage fraction.

StorageFraction = 1 - [Shoot].DMDemands.Structural.StructuralFraction

3.11.6.2 Nitrogen Demand

The N demand is calculated as defined in NDemands, based on DM demand the N concentration of each biomass pool.

3.11.6.2.1 NDemands

This is the collection of functions for calculating the demands for each of the biomass pools (Structural, Metabolic, and Storage).

3.11.6.2.1.1 Structural

Structural = [Shoot].minimumNconc × [Shoot].potentialDMAllocation.Structural

3.11.6.2.1.2 Metabolic

Metabolic = MetabolicNconc × [Shoot].potentialDMAllocation.Structural

Where:

MetabolicNconc = [Shoot].criticalNConc - [Shoot].minimumNconc

3.11.6.2.1.3 Storage

The partitioning of daily growth to storage biomass attempts to bring the organ's N content to the maximum concentration.

Storage = [Shoot].maximumNconc × ([Shoot].Live.Wt + potentialAllocationWt) - [Shoot].Live.N

The demand for storage N is further reduced by a factor specified by the [Shoot].NitrogenDemandSwitch.

3.11.6.2.2 MinimumNConc

MinimumNConc = 0.006 (g/g)

CriticalNConc = [Shoot].MinimumNConc

3.11.6.2.3 MaximumNConc

The demand for N is reduced by a factor specified by the NitrogenDemandSwitch.

3.11.6.2.4 NitrogenDemandSwitch

A value of 1 is returned if phenology is between BudBurst and LeafFall phases, otherwise a value of 0 is returned.

3.11.6.3 Dry Matter Supply

Shoot does not reallocate DM when senescence of the organ occurs.

Shoot does not retranslocate non-structural DM.

3.11.6.4 Nitrogen Supply

Shoot will reallocate 20% of N that senesces each day.

Shoot will retranslocate 5% of non-structural N each day.

3.11.6.5 Senescence and Detachment

Shoot has senescence parameterised to zero so all biomass in this organ will remain alive.

Shoot has detachment parameterised to zero so all biomass in this organ will remain with the plant until a defoliation or harvest event occurs.

3.11.7 Cordon

This organ is simulated using a GenericOrgan type. It is parameterised to calculate the growth, senescence, and detachment of any organ that does not have specific functions.

3.11.7.1 Dry Matter Demand

The dry matter demand for the organ is calculated as defined in DMDemands, based on the DMDemandFunction and partition fractions for each biomass pool.

3.11.7.1.1 DMDemands

This is the collection of functions for calculating the demands for each of the biomass pools (Structural, Metabolic, and Storage).

3.11.7.1.1.1 Structural

Structural = DMDemandFunction × StructuralFraction

Where:

DMDemandFunction = PartitionFraction [Arbitrator].DM.TotalFixationSupply

Where:

PartitionFraction = 0.01

StructuralFraction = 0.8 (percentage)

3.11.7.1.1.2 Metabolic

Metabolic = 0 (g/m2)

3.11.7.1.1.3 Storage

The partitioning of daily growth to storage biomass is based on a storage fraction.

StorageFraction = 1 - [Cordon].DMDemands.Structural.StructuralFraction

3.11.7.2 Nitrogen Demand

The N demand is calculated as defined in NDemands, based on DM demand the N concentration of each biomass pool.

3.11.7.2.1 NDemands

This is the collection of functions for calculating the demands for each of the biomass pools (Structural, Metabolic, and Storage).

3.11.7.2.1.1 Structural

Structural = [Cordon].minimumNconc × [Cordon].potentialDMAllocation.Structural

3.11.7.2.1.2 Metabolic

Metabolic = MetabolicNconc × [Cordon].potentialDMAllocation.Structural

Where:

MetabolicNconc = [Cordon].criticalNConc - [Cordon].minimumNconc

3.11.7.2.1.3 Storage

The partitioning of daily growth to storage biomass attempts to bring the organ's N content to the maximum concentration.

Storage = [Cordon].maximumNconc × ([Cordon].Live.Wt + potentialAllocationWt) - [Cordon].Live.N

The demand for storage N is further reduced by a factor specified by the [Cordon].NitrogenDemandSwitch.

3.11.7.2.2 MinimumNConc

MinimumNConc = 0.006 (g/g)

CriticalNConc = [Cordon].MinimumNConc

3.11.7.2.3 MaximumNConc

The demand for N is reduced by a factor specified by the NitrogenDemandSwitch.

3.11.7.2.4 NitrogenDemandSwitch

A value of 1 is returned if phenology is between BudBurst and LeafFall phases, otherwise a value of 0 is returned.

3.11.7.3 Dry Matter Supply

Cordon will reallocate 20% of DM that senesces each day.

Cordon does not retranslocate non-structural DM.

3.11.7.4 Nitrogen Supply

Cordon does not reallocate N when senescence of the organ occurs.

Cordon will retranslocate 5% of non-structural N each day.

3.11.7.5 Senescence and Detachment

Cordon has senescence parameterised to zero so all biomass in this organ will remain alive.

Cordon has detachment parameterised to zero so all biomass in this organ will remain with the plant until a defoliation or harvest event occurs.

3.11.8 Trunk

This organ is simulated using a GenericOrgan type. It is parameterised to calculate the growth, senescence, and detachment of any organ that does not have specific functions.

3.11.8.1 Dry Matter Demand

The dry matter demand for the organ is calculated as defined in DMDemands, based on the DMDemandFunction and partition fractions for each biomass pool.

3.11.8.1.1 DMDemands

This is the collection of functions for calculating the demands for each of the biomass pools (Structural, Metabolic, and Storage).

3.11.8.1.1.1 Structural

Structural = DMDemandFunction × StructuralFraction

Where:

DMDemandFunction = PartitionFraction [Arbitrator].DM.TotalFixationSupply

Where:

PartitionFraction = 0.05

StructuralFraction = 0.8 (g/g)

3.11.8.1.1.2 Metabolic

Metabolic = 0 (g/m2)

3.11.8.1.1.3 Storage

The partitioning of daily growth to storage biomass is based on a storage fraction.

StorageFraction = 1 - [Trunk].DMDemands.Structural.StructuralFraction

3.11.8.2 Nitrogen Demand

The N demand is calculated as defined in NDemands, based on DM demand the N concentration of each biomass pool.

3.11.8.2.1 NDemands

This is the collection of functions for calculating the demands for each of the biomass pools (Structural, Metabolic, and Storage).

3.11.8.2.1.1 Structural

Structural = [Trunk].minimumNconc × [Trunk].potentialDMAllocation.Structural

3.11.8.2.1.2 Metabolic

Metabolic = MetabolicNconc × [Trunk].potentialDMAllocation.Structural

Where:

MetabolicNconc = [Trunk].criticalNConc - [Trunk].minimumNconc

3.11.8.2.1.3 Storage

The partitioning of daily growth to storage biomass attempts to bring the organ's N content to the maximum concentration.

Storage = [Trunk].maximumNconc × ([Trunk].Live.Wt + potentialAllocationWt) - [Trunk].Live.N

The demand for storage N is further reduced by a factor specified by the [Trunk].NitrogenDemandSwitch.

3.11.8.2.2 MinimumNConc

MinimumNConc = 0.006

CriticalNConc = [Trunk].MinimumNConc

3.11.8.2.3 MaximumNConc

The demand for N is reduced by a factor specified by the NitrogenDemandSwitch.

3.11.8.2.4 NitrogenDemandSwitch

A value of 1 is returned if phenology is between BudBurst and LeafFall phases, otherwise a value of 0 is returned.

3.11.8.3 Dry Matter Supply

Trunk will reallocate 20% of DM that senesces each day.

Trunk will retranslocate 10% of non-structural DM each day.

3.11.8.4 Nitrogen Supply

Trunk does not reallocate N when senescence of the organ occurs.

Trunk will retranslocate 5% of non-structural N each day.

3.11.8.5 Senescence and Detachment

Trunk has senescence parameterised to zero so all biomass in this organ will remain alive.

Trunk has detachment parameterised to zero so all biomass in this organ will remain with the plant until a defoliation or harvest event occurs.

3.11.9 Berry

This organ uses a generic model for plant reproductive components. Yield is calculated from its components in terms of organ number and size (for example, grain number and grain size).

The final yield was calculated by vines per hectare, shoots per vine, bunches per shoot, flower number per bunch, berries per bunch,and mean berry weight. The effects of temperature and carbon status were or will be considered on bunches per shoot, flower number per bunch, berries per bunch,and mean berry weight.

3.11.9.1 MetFactors

The final yield was calculated by vines per hectare, shoots per vine, bunches per shoot, flower number per bunch, berries per bunch,and mean berry weight. The effects of temperature and carbon status were or will be considered on bunches per shoot, flower number per bunch, berries per bunch,and mean berry weight.

3.11.9.1.1 BUN_TmaxIni

BUN_TmaxIni = TmaxIni / Time

Where:

A function that accumulates values from child functions

Tmax = [Weather].MaxT

A function that accumulates values from child functions

Days = 1 (days/day)

3.11.9.1.2 BUN_RadIni

BUN_RadIni = RadIni / Time

Where:

A function that accumulates values from child functions

Rad = [Weather].Radn

A function that accumulates values from child functions

Days = 1 (days/day)

3.11.9.1.3 BEN_TmeanFlow

BEN_TmeanFlow = TmeanFlow / Time

Where:

A function that accumulates values from child functions

Tmean = [Weather].MeanT

A function that accumulates values from child functions

Days = 1 (days/day)

3.11.9.1.4 BEN_RainTotFlow

BEN_RainTotFlow = RainTotFlow / Constant

Where:

A function that accumulates values from child functions

RainTot = [Weather].Rain

Constant = 1

3.11.9.1.5 BEN_TmaxIni

BEN_TmaxIni = TmaxIni / Time

Where:

A function that accumulates values from child functions

Tmax = [Weather].MaxT

A function that accumulates values from child functions

Days = 1 (days/day)

3.11.9.1.6 BM_TmeanFlow

BM_TmeanFlow = TmeanFlow / Time

Where:

A function that accumulates values from child functions

Tmean = [Weather].MeanT

A function that accumulates values from child functions

Days = 1 (days/day)

3.11.9.1.7 BM_RainTotFlow

BM_RainTotFlow = RainTotFlow / Constant

Where:

A function that accumulates values from child functions

RainTot = [Weather].Rain

Constant = 1

3.11.9.1.8 BM_RadFlow

BM_RadFlow = RadFlow / Time

Where:

A function that accumulates values from child functions

Rad = [Weather].Radn

A function that accumulates values from child functions

Days = 1 (days/day)

3.11.9.1.9 BM_RainTotVer

BM_RainTotVer = RainTotVer / Constant

Where:

A function that accumulates values from child functions

RainTot = [Weather].Rain

Constant = 1

3.11.9.2 YieldComponent

3.11.9.2.1 BunchesPerVine

here we use a trigger function for bunches per shoot as the value of mean flowering temperature Y0 is updating itself during the growing season, which would cause inconvience if we directly correlates the bunches per shoot and mean flowering temperature Y0.

Numbers were guessed based on Eltom 2013 Fig. 7.7. y=0.0981*Ta + 0.35

Bunches per shoot is affected by the temperature and CHO during initiation, it is further affected by the CHO during bud burst. The later decided whether the potential initiated inflorescence can grow out successfully or been realized.

For simiplicity, right now, the CHO effect during bud burst is expressed as a function of number of buds retained.

3.11.9.2.2 BerryNum

here we use a trigger function for bunches per shoot as the value of mean flowering temperature Y0 is updating

itself during the growing season, which would cause inconvience if we directly correlates the bunches per shoot and mean flowering temperature Y0.

Numbers were guessed based on Eltom 2013 Fig. 7.7. y=0.0981*Ta + 0.35

Bunches per shoot is affected by the temperature and CHO during initiation, it is further affected by the CHO during bud burst. The later decided whether the potential initiated inflorescence can grow out successfully or been realized.

For simiplicity, right now, the CHO effect during bud burst is expressed as a function of number of buds retained.

3.11.9.2.3 BerryMass

here we use a trigger function for bunches per shoot as the value of mean flowering temperature Y0 is updating itself during the growing season, which would cause inconvience if we directly correlates the bunches per shoot and mean flowering temperature Y0.

Numbers were guessed based on Eltom 2013 Fig. 7.7. y=0.0981*Ta + 0.35

Bunches per shoot is affected by the temperature and CHO during initiation, it is further affected by the CHO during bud burst. The later decided whether the potential initiated inflorescence can grow out successfully or been realized.

For simiplicity, right now, the CHO effect during bud burst is expressed as a function of number of buds retained.

3.11.9.3 NumberFunction

NumberFunction = [Berry].YieldComponent.BunchesPerVine × [Berry].YieldComponent.BerryNum × [Grapevine].Population

3.11.9.4 SingleBerryFW

SingleBerryFW = BetaGrowthFunction

Where:

3.11.9.4.1 BetaGrowthFunction

a beta growth function of the form y = Ymax * (1 + (te - t)/(te-tm))* (t/te)^(te/(te-tm))

Ymax = [Berry].YieldComponent.BerryMass

XValue = [Berry].ThermalTimeAfterFlowering

3.11.9.5 TotalBerryFW

TotalBerryFW = [Berry].NumberFunction × BetaGrowthFunction

Where:

3.11.9.5.1 BetaGrowthFunction

a beta growth function of the form y = Ymax * (1 + (te - t)/(te-tm))* (t/te)^(te/(te-tm))

Ymax = [Berry].YieldComponent.BerryMass

XValue = [Berry].ThermalTimeAfterFlowering

3.11.9.6 SingleBerryDW

SingleBerryDW = BetaGrowthFunction

Where:

3.11.9.6.1 BetaGrowthFunction

a beta growth function of the form y = Ymax * (1 + (te - t)/(te-tm))* (t/te)^(te/(te-tm))

Ymax = 0.5529 (g)

XValue = [Berry].ThermalTimeAfterFlowering

3.11.9.7 WaterContent

WaterContent = Constant - AfterFlowering

Where:

Constant = 1

3.11.9.7.1 AfterFlowering

0 between Flowering and LeafFall and a value of zero outside of this period

3.11.9.8 Brix

3.11.9.8.1 BeforeVeraison

The value of Brix from Flowering to Veraison is calculated as follows:

Constant = 5

3.11.9.8.2 AfterVeraison

The value of Brix from Veraison to LeafFall is calculated as follows:

3.11.9.8.2.1 DivideFunction

DivideFunction = MaximumFunction / constant

Where:

MaximumFunction = maximum (SubtractFunction, zero)

Where:

SubtractFunction = constant - [Berry].WaterContent

Where:

constant = 0.944

zero = 0

constant = 0.0082

Brix has a value of zero for phases not specified above

3.11.9.9 TitratableAcid

3.11.9.9.1 BeforeVeraison

The value of TitratableAcid from Flowering to Veraison is calculated as follows:

Constant = 0

3.11.9.9.2 AfterVeraison

The value of TitratableAcid from Veraison to LeafFall is calculated as follows:

3.11.9.10 TitratableAcid

An exponential function

XValue = [Berry].ThermalTimeAfterVeraison

TitratableAcid has a value of zero for phases not specified above

3.11.9.11 DMDemandFunction

DMDemandFunction = DeltaFunction

Where:

DeltaFunction is the daily differential of

3.11.9.11.1 Integral

Integral = [Berry].NumberFunction × BetaGrowthFunction

Where:

a beta growth function of the form y = Ymax * (1 + (te - t)/(te-tm))* (t/te)^(te/(te-tm))

Ymax = 0.5529 (g)

XValue = [Berry].ThermalTimeAfterFlowering

3.11.9.12 ThermalTimeAfterVeraison

ThermalTimeAfterVeraison is a daily accumulation of the values of functions listed below between the Veraison and LeafFall stages. Function values added to the accumulate total each day are:

ThermalTime = [Phenology].ThermalTime

3.11.9.13 ThermalTimeAfterFlowering

ThermalTimeAfterFlowering is a daily accumulation of the values of functions listed below between the Flowering and LeafFall stages. Function values added to the accumulate total each day are:

ThermalTime = [Phenology].ThermalTime

3.11.9.14 MaxNConcDailyGrowth

MaxNConcDailyGrowth = 0.01 (g/g)

3.11.9.15 NFillingRate

NFillingRate = 0 (g/m2/d)

3.11.9.16 MinimumNConc

MinimumNConc = 0 (g/g)

3.11.9.17 MaximumNConc

MaximumNConc = 0.0001 (g/g)

3.11.9.18 BiomassRemovalDefaults

This organ will respond to certain management actions by either removing some of its biomass from the system or transferring some of its biomass to the soil surface residues. The following table describes the default proportions of live and dead biomass that are transferred out of the simulation using "Removed" or to soil surface residue using "To Residue" for a range of management actions. The total percentage removed for live or dead must not exceed 100%. The difference between the total and 100% gives the biomass remaining on the plant. These can be changed during a simulation using a manager script.

Method	% Live Removed	% Dead Removed	% Live To Residue	% Dead To Residue
Harvest	0	0	0	0
Cut	0	0	0	0
Prune	0	0	0	0
Graze	0	0	0	0
Thin	0	0	0	0

3.11.9.19 MaximumPotentialGrainSize

MaximumPotentialGrainSize = 2.5 (g fresh or dry)

3.11.9.20 DMConversionEfficiency

DMConversionEfficiency = 1

3.11.9.21 RemobilisationCost

RemobilisationCost = 0

3.11.9.22 CarbonConcentration

CarbonConcentration = 0.4

3.11.10 Root

The generic root model calculates root growth in terms of rooting depth, biomass accumulation and subsequent root length density in each soil layer.

Root Growth

Roots grow downwards through the soil profile, with initial depth determined by sowing depth and the growth rate determined by RootFrontVelocity. The RootFrontVelocity is modified by multiplying it by the soil's XF value; which represents any resistance posed by the soil to root extension. Root depth is also constrained by a maximum root depth.

Root length growth is calculated using the daily DM partitioned to roots and a specific root length. Root proliferation in layers is calculated using an approach similar to the generalised equimarginal criterion used in economics. The uptake of water and N per unit root length is used to partition new root material into layers of higher 'return on investment'.

Dry Matter Demands

A daily DM demand is provided to the organ arbitrator and a DM supply returned. By default, 100% of the dry matter (DM) demanded from the root is structural. The daily loss of roots is calculated using a SenescenceRate function. All senesced material is automatically detached and added to the soil FOM.

Nitrogen Demands

The daily structural N demand from root is the product of total DM demand and the minimum N concentration. Any N above this is considered Storage and can be used for retranslocation and/or reallocation as the respective factors are set to values other then zero.

Nitrogen Uptake

Potential N uptake by the root system is calculated for each soil layer (i) that the roots have extended into. In each layer potential uptake is calculated as the product of the mineral nitrogen in the layer, a factor controlling the rate of extraction (kNO3 or kNH4), the concentration of N form (ppm), and a soil moisture factor (NUptakeSWFactor) which typically decreases as the soil dries.

$NO3\ uptake = NO3_i \times kNO3 \times NO3_{ppm,\ i} \times NUptakeSWFactor$

$NH4\ uptake = NH4_i \times kNH4 \times NH4_{ppm,\ i} \times NUptakeSWFactor$

Nitrogen uptake demand is limited to the maximum daily potential uptake (MaxDailyNUptake) and the plants N demand. The demand for soil N is then passed to the soil arbitrator which determines how much of the N uptake demand each plant instance will be allowed to take up.

Water Uptake

Potential water uptake by the root system is calculated for each soil layer that the roots have extended into. In each layer potential uptake is calculated as the product of the available water in the layer (water above LL limit) and a factor controlling the rate of extraction (KL). The values of both LL and KL are set in the soil interface and KL may be further modified by the crop via the KLModifier function.

$SW\ uptake = (SW_i - LL_i) \times KL_i \times KLModifier$

3.11.10.1 NitrogenDemandSwitch

A value of 1 is returned if phenology is between BudBurst and LeafFall phases, otherwise a value of 0 is returned.

3.11.10.2 MinimumNConc

MinimumNConc = 0.01 (g/g)

3.11.10.3 MaximumNConc

MaximumNConc = 0.01 (g/g)

3.11.10.4 MaximumRootDepth

MaximumRootDepth = 1000000 (mm)

3.11.10.5 BiomassRemovalDefaults

This organ will respond to certain management actions by either removing some of its biomass from the system or transferring some of its biomass to the soil surface residues. The following table describes the default proportions of live and dead biomass that are transferred out of the simulation using "Removed" or to soil surface residue using "To Residue" for a range of management actions. The total percentage removed for live or dead must not exceed 100%. The difference between the total and 100% gives the biomass remaining on the plant. These can be changed during a simulation using a manager script.

Method	% Live Removed	% Dead Removed	% Live To Residue	% Dead To Residue
Harvest	0	0	0	0
Cut	0	0	0	0
Prune	0	0	0	0
Graze	0	0	0	0

3.11.10.6 InitialDM

InitialDM = 2500 (g/plant)

3.11.10.7 SpecificRootLength

SpecificRootLength = 40 (m/g)

3.11.10.8 RootFrontVelocity

RootFrontVelocity = 10 (mm/d)

3.11.10.9 KNO3

KNO3 = 0.02

3.11.10.10 KNH4

KNH4 = 0.003

3.11.10.11 MaxDailyNUptake

MaxDailyNUptake = 6

3.11.10.12 DMConversionEfficiency

DMConversionEfficiency = 1

3.11.10.13 MaintenanceRespirationFunction

MaintenanceRespirationFunction = 0.002

3.11.10.14 RemobilisationCost

RemobilisationCost = 0

3.11.10.15 CarbonConcentration

CarbonConcentration = 0.4

3.11.10.16 DMRetranslocationFactor

DMRetranslocationFactor = 0.1

3.11.10.17 DMDemands

This is the collection of functions for calculating the demands for each of the biomass pools (Structural, Metabolic, and Storage).

3.11.10.17.1 Structural

Structural = DMDemandFunction × StructuralFraction

Where:

3.11.10.17.1.1 DMDemandFunction

DMDemandFunction = PartitionFraction [Arbitrator].DM.TotalFixationSupply

Where:

PartitionFraction = 0.2

StructuralFraction = 0.8 (g/g)

3.11.10.17.2 Metabolic

Metabolic = 0 (g/m2)

3.11.10.17.3 Storage

The partitioning of daily growth to storage biomass is based on a storage fraction.

3.11.10.17.3.1 StorageFraction

StorageFraction = 1 - [Root].DMDemands.Structural.StructuralFraction

3.11.10.18 NDemands

This is the collection of functions for calculating the demands for each of the biomass pools (Structural, Metabolic, and Storage).

3.11.10.18.1 Structural

Structural = [Root].minimumNconc × [Root].potentialDMAllocation.Structural

3.11.10.18.2 Metabolic

Metabolic = MetabolicNconc × [Root].potentialDMAllocation.Structural

Where:

3.11.10.18.2.1 MetabolicNconc

MetabolicNconc = [Root].criticalNConc - [Root].minimumNconc

3.11.10.18.3 Storage

The partitioning of daily growth to storage biomass attempts to bring the organ's N content to the maximum concentration.

Storage = [Root].maximumNconc × ([Root].Live.Wt + potentialAllocationWt) - [Root].Live.N

The demand for storage N is further reduced by a factor specified by the [Root].NitrogenDemandSwitch.

CriticalNConc = [Root].MinimumNConc

3.11.10.19 KLModifier

This is important in SLURP as it is set for each species to represent their differences in rooting patterns between crop species

3.11.11 Arbitrator

The Arbitrator class determines the allocation of dry matter (DM) and Nitrogen between each of the organs in the crop model. Each organ can have up to three different pools of biomass:

- **Structural biomass** which is essential for growth and remains within the organ once it is allocated there.
- **Metabolic biomass** which generally remains within an organ but is able to be re-allocated when the organ senesces and may be retranslocated when demand is high relative to supply.
- **Storage biomass** which is partitioned to organs when supply is high relative to demand and is available for retranslocation to other organs whenever supply from uptake, fixation, or re-allocation is lower than demand.

The process followed for biomass arbitration is shown in Figure 4. Arbitration calculations are triggered by a series of events (shown below) that are raised every day. For these calculations, at each step the Arbitrator exchange information with each organ, so the basic computations of demand and supply are done at the organ level, using their specific parameters.

1. **doPotentialPlantGrowth**. When this event occurs, each organ class executes code to determine their potential growth, biomass supplies and demands. In addition to demands for structural, non-structural and metabolic biomass (DM and N) each organ may have the following biomass supplies:
2. **Fixation supply**. From photosynthesis (DM) or symbiotic fixation (N)
3. **Uptake supply**. Typically uptake of N from the soil by the roots but could also be uptake by other organs (eg foliage application of N).
4. **Retranslocation supply**. Storage biomass that may be moved from organs to meet demands of other organs.
5. **Reallocation supply**. Biomass that can be moved from senescing organs to meet the demands of other organs.
6. **doPotentialPlantPartitioning.** On this event the Arbitrator first executes the DoDMSetup() method to gather the DM supplies and demands from each organ, these values are computed at the organ level. It then executes the DoPotentialDMAllocation() method which works out how much biomass each organ would be allocated assuming N supply is not limiting and sends these allocations to the organs. Each organ then uses their potential DM allocation to determine their N demand (how much N is needed to produce that much DM) and the arbitrator calls DoNSetup() to gather the N supplies and demands from each organ and begin N arbitration. Firstly DoNReallocation() is called to redistribute N that the plant has available from senescing organs. After this step any unmet N demand is considered as plant demand for N uptake from the soil (N Uptake Demand).
7. **doNutrientArbitration.** When this event occurs, the soil arbitrator gets the N uptake demands from each plant (where multiple plants are growing in competition) and their potential uptake from the soil and determines how much of their demand that the soil is able to provide. This value is then passed back to each plant instance as their Nuptake and doNUptakeAllocation() is called to distribute this N between organs.
8. **doActualPlantPartitioning.** On this event the arbitrator call DoNRetranslocation() and DoNFixation() to satisfy any unmet N demands from these sources. Finally, DoActualDMAllocation is called where DM allocations to each organ are reduced if the N allocation is insufficient to achieve the organs minimum N concentration and final allocations are sent to organs.

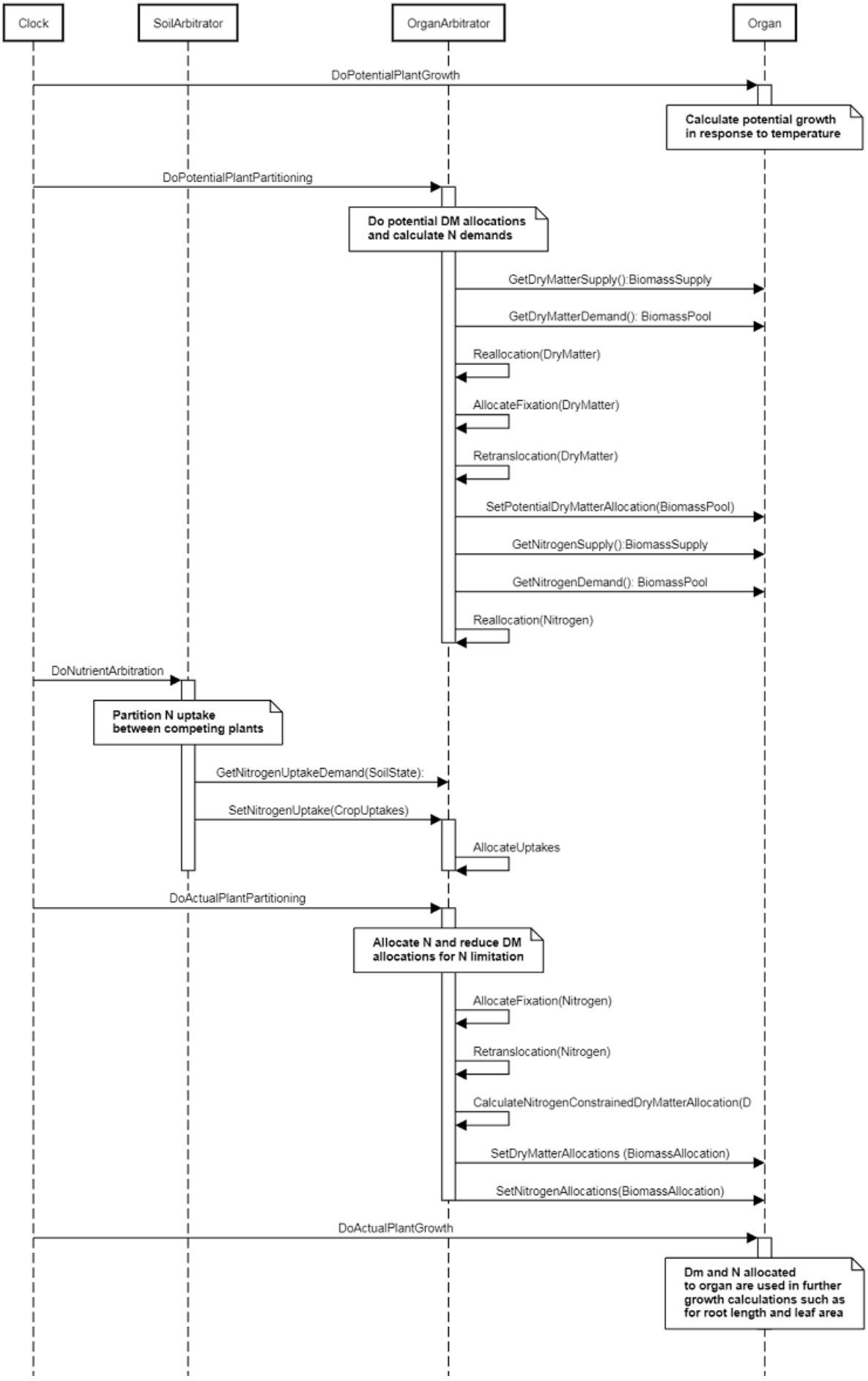


Figure 4: Schematic showing the procedure for arbitration of biomass partitioning. Pink boxes represent events that occur every day and their numbering shows the order of calculations. Blue boxes represent the methods that are called when these events occur. Orange boxes contain properties that make up the organ/arbitrator interface. Green boxes are organ specific properties.

4 DataStore

A storage service for reading and writing to/from a database.

4.1 PhenoObs

Reads the contents of a specific sheet from an EXCEL file and stores into the DataStore.

4.2 PredictedObserved

Reads the contents of a file (in apsim format) and stores into the DataStore. If the file has a column name of 'SimulationName' then this model will only input data for those rows where the data in column 'SimulationName' matches the name of the simulation under which this input model sits. If the file does NOT have a 'SimulationName' column then all data will be input.

4.2.1 PredictedObserved

Variable	n	Slope	Intercept	R2	RMSE	NSE	ME	MAE
BerryMass	47	0.435	1.428	0.158	0.313	-4.705	0.274	0.283
BerryNum	46	0.194	51.280	0.026	16.205	-1.045	1.083	13.276
BudBurstDOY	65	0.823	16.932	0.181	10.740	-2.174	-1.631	8.923
BunchesPerVine	47	0.239	49.174	0.348	7.206	0.314	0.030	5.480
f.rad.int	1337	1.064	0.146	0.499	0.234	-1.472	0.172	0.206
FloweringDOY	65	0.386	103.968	0.240	7.474	0.124	-1.338	5.738
psw	608	0.482	159.653	0.547	65.104	-0.067	49.066	54.776
transpiration	965	0.081	0.894	0.032	1.193	-0.630	-0.716	0.912
VeraisonDOY	65	0.448	129.306	0.277	8.494	0.152	-1.262	6.462

5 BaseSim

Double Guyôt-trained vines, pruned each year to retain 24 nodes were monitored. Vines were trained using vertical shoot positioning (VSP), with an exposed leaf area height of 1.2 m, and trimmed to maintain a compact canopy. Vines were planted 1.8 m apart within the row, and 2.4 m apart between rows. Average flowering date was determined by monitoring all the inflorescences on one cane on a vine in each plot on a regular basis throughout flowering. Likewise, regular berry samples (at least weekly) were taken from before the date of véraison until harvest at a soluble solids concentration value of 21.5 oBrix.

The soil class encapsulates a soil characterisation and 0 or more soil samples. the methods in this class that return double[] always return using the "Standard layer structure" i.e. the layer structure as defined by the Water child object. method. Mapping will occur to achieve this if necessary. To obtain the "raw", unmapped, values use the child classes e.g. SoilWater, Analysis and Sample.

A model for capturing physical soil parameters

A soil crop parameterization class.

The SoilWater module is a cascading water balance model that owes much to its precursors in CERES (Jones and Kiniry, 1986) and PERFECT(Littleboy et al, 1992). The algorithms for redistribution of water throughout the soil profile have been inherited from the CERES family of models.

The water characteristics of the soil are specified in terms of the lower limit (ll15), drained upper limit(dul) and saturated(sat) volumetric water contents. Water movement is described using separate algorithms for saturated or unsaturated flow. It is notable that redistribution of solutes, such as nitrate- and urea-N, is carried out in this module.

Modifications adopted from PERFECT include: * the effects of surface residues and crop cover on modifying runoff and reducing potential soil evaporation, * small rainfall events are lost as first stage evaporation rather than by the slower process of second stage evaporation, and * specification of the second stage evaporation

coefficient(cona) as an input parameter, providing more flexibility for describing differences in long term soil drying due to soil texture and environmental effects.

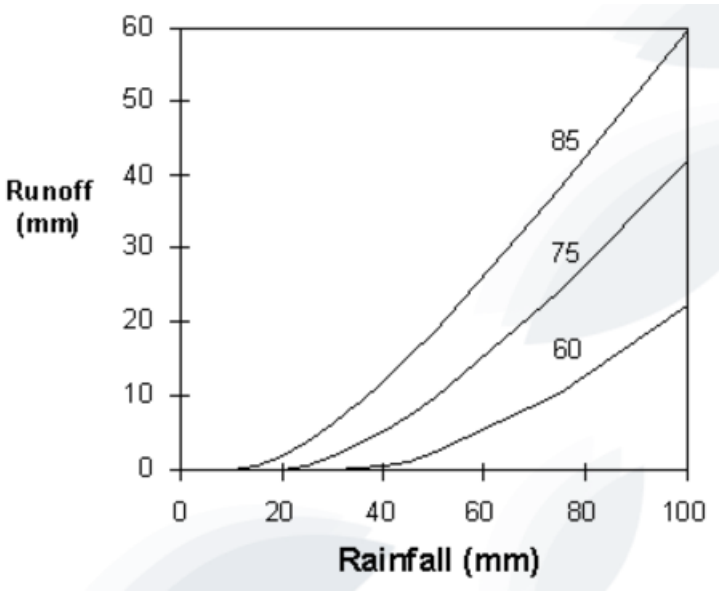
The module is interfaced with SurfaceOrganicMatter and crop modules so that simulation of the soil water balance responds to change in the status of surface residues and crop cover(via tillage, decomposition and crop growth).

Enhancements beyond CERES and PERFECT include: * the specification of swcon for each layer, being the proportion of soil water above dul that drains in one day * isolation from the code of the coefficients determining diffusivity as a function of soil water (used in calculating unsaturated flow).Choice of diffusivity coefficients more appropriate for soil type have been found to improve model performance. * unsaturated flow is permitted to move water between adjacent soil layers until some nominated gradient in soil water content is achieved, thereby accounting for the effect of gravity on the fully drained soil water profile.

SoilWater is called by APSIM on a daily basis, and typical of such models, the various processes are calculated consecutively. This contrasts with models such as SWIM that solve simultaneously a set of differential equations that describe the flow processes.

Runoff from rainfall is calculated using the USDA-Soil Conservation Service procedure known as the curve number technique. The procedure uses total precipitation from one or more storms occurring on a given day to estimate runoff. The relation excludes duration of rainfall as an explicit variable, and so rainfall intensity is ignored. When irrigation is applied it can optionally be included in the runoff calculation. This flag (willRunoff) can be set when applying irrigation.

Figure: Runoff response curves (ie runoff as a function of total daily rainfall) are specified by numbers from 0 (no runoff) to 100 (all runoff). Response curves for three runoff curve numbers for rainfall varying between 0 and 100 mm per day.



The user supplies a curve number for average antecedent rainfall conditions (CN2Bare). From this value the wet (high runoff potential) response curve and the dry (low runoff potential) response curve are calculated. The SoilWater module will then use the family of curves between these two extremes for calculation of runoff depending on the daily moisture status of the soil. The effect of soil moisture on runoff is confined to the effective hydraulic depth as specified in the module's ini file and is calculated to give extra weighting to layers closer to the soil surface. Figure: Runoff response curves (ie runoff as a function of total daily rainfall) are specified by numbers from 0 (no runoff) to 100 (all runoff).

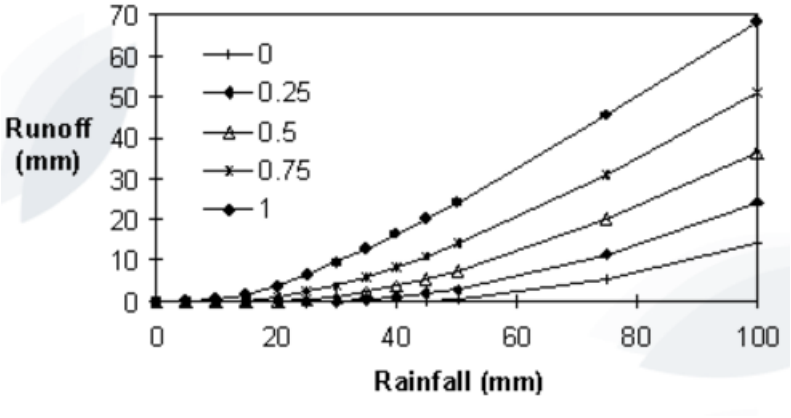
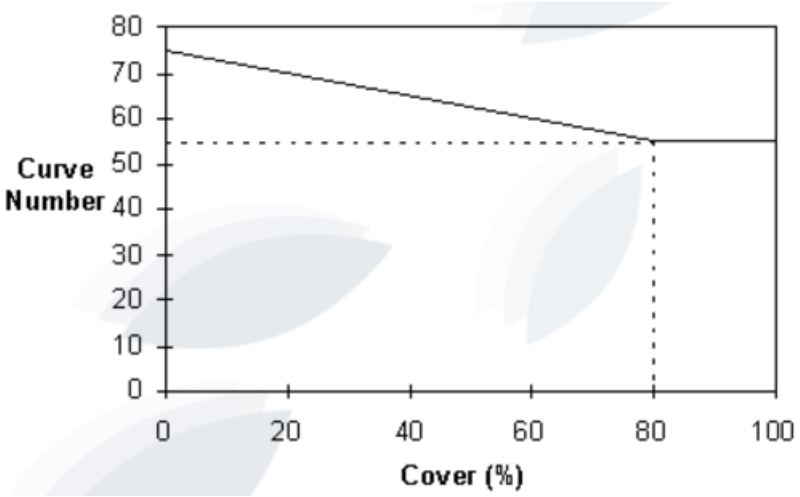


Figure: Residue cover effect on runoff curve number where bare soil curve number is 75 and total reduction in curve number is 20 at 80% cover.



Surface residues inhibit the transport of water across the soil surface during runoff events and so different families of response curves are used according to the amount of crop and residue cover.The extent of the effect on runoff is specified by a threshold surface cover (CNCov), above which there is no effect, and the corresponding curve number reduction (CNRed).

Tillage of the soil surface also reduces runoff potential, and a similar modification of Curve Number is used to represent this process. A tillage event is directed to the module, specifying cn_red, the CN reduction, and cn_rain, the rainfall amount required to remove the tillage roughness. CN2 is immediately reduced and increases linearly with cumulative rain, ie.roughness is smoothed out by rain.

Implements the curve number reduction caused by cover.

Implements the curve number reduction caused by tillage. Mark Littleboy's tillage effect on runoff (used in PERFECT v2.0) Littleboy, Cogle, Smith, Yule and Rao(1996). Soil management and production of alfisols in the SAT's I. Modelling the effects of soil management on runoff and erosion.Aust.J.Soil Res. 34: 91-102.

Soil evaporation is assumed to take place in two stages: the constant and the falling rate stages.

In the first stage the soil is sufficiently wet for water to be transported to the surface at a rate at least equal to the potential evaporation rate. Potential evapotranspiration is calculated using an equilibrium evaporation concept as modified by Priestly and Taylor(1972).

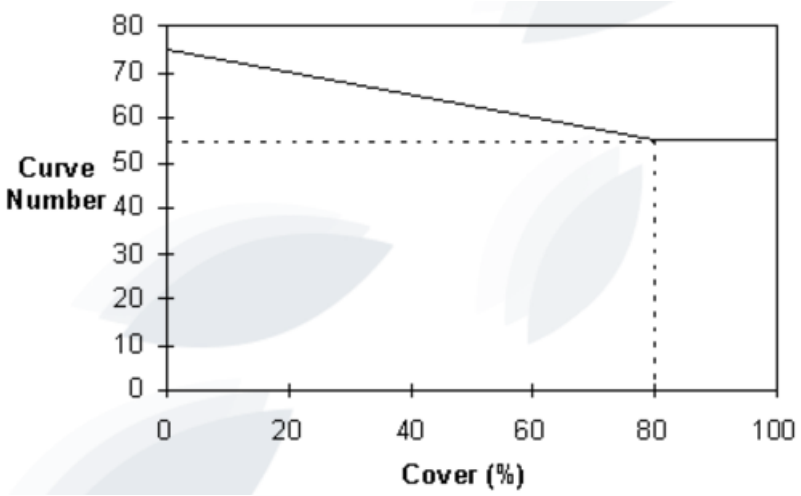
Once the water content of the soil has decreased below a threshold value the rate of supply from the soil will be less than potential evaporation (second stage evaporation). These behaviors are described in SoilWater through the use of two parameters: U and CONA.

The parameter U (as from CERES) represents the amount of cumulative evaporation before soil supply decreases below atmospheric demand. The rate of soil evaporation during the second stage is specified as a function of time since the end of first stage evaporation. The parameter CONA (from PERFECT) specifies the change in cumulative second stage evaporation against the square root of time.

i.e. Es = CONA $t^{1/2}$

Water lost by evaporation is removed from the surface layer of the soil profile thus this layer can dry below the wilting point or lower limit (LL) to a specified air-dry water content (air_dry).

Figure: Cumulative Soil Evaporation through time for U = 6 mm and CONA = 3.5.



For $t <= t_1$ Es = Eos For $t > t_1$ Es = U x t + CONA x Sqrt($t-t_1$)

Lateral movement of water is calculated from a user specified lateral inflow ('InFlow').

Lateral Outflow is the flow that occurs as a result of the soil water going above DUL and the soil being on a slope. So if there is no slope and the water goes above DUL there is no lateral outflow. KLAT is just the lateral resistance of the soil to this flow. It is a soil water conductivity.

The calculation of lateral outflow on a layer basis is now performed using the equation: Lateral flow for a layer = KLAT * d * s / $(1 + s^2)^{0.5}$ * L / A * unit conversions. Where: KLAT = lateral conductivity (mm/day) d = depth of saturation in the layer(mm) = Thickness * (SW - DUL) / (SAT - DUL) if SW > DUL. (Note this allows lateral flow in any "saturated" layer, not just those inside a water table.) s = slope(m / m) L = catchment discharge width. Basically, it's the width of the downslope boundary of the catchment. (m) A = catchment area. (m^2)

NB. with Lateral Inflow it is assumed that ALL the water goes straight into the layer. Irrespective of the layers ability to hold it. It is like an irrigation. KLAT has no effect and does not alter the amount of water coming into the layer. KLAT only alters the amount of water flowing out of the layer

When water content in any layer is below SAT but above DUL, a fraction of the water drains to the next deepest layer each day.

Flux = SWCON x (SW - DUL)

Infiltration or water movement into any layer that exceeds the saturation capacity of the layer automatically cascades to the next layer.

For water contents below DUL, movement depends upon the water content gradient between adjacent layers and the diffusivity, which is a function of the average water contents of the two layers.

Unsaturated flow may occur both towards the surface and downwards, but cannot move water out of the bottom of the deepest layer in the profile. Flow between adjacent layers ceases at a soil water gradient (gravity_gradient) specified in the SoilWater ini file.

The diffusivity is defined by two parameters set by the user (diffus_const, diffus_slope) in the SoilWater parameter set (Default values, from CERES, are 88 and 35.4, but 40 and 16 have been found to be more appropriate for describing water movement in cracking clay soils).

Diffusivity = diffus_const x exp(diffus_slope x thet_av)

where thet_av is the average of SW - LL15 across the two layers. Flow = Diffusivity x Volumetric Soil Water Gradient

Water table is the depth (in mm) below the ground surface of the first layer which is above saturation.

Computes the soil C and N processes

This class encapsulates a SoilNitrogen model NO3 solute.

This class encapsulates a SoilNitrogen model NH4 solute.

This class encapsulates a SoilNitrogen model urea solute.

This class encapsulates a SoilNitrogen model 'PlantAvailableNO3' solute.

This class encapsulates a SoilNitrogen model NH4 solute.

A model for capturing soil organic parameters

This class captures chemical soil data

Represents the simulation initial water status. There are multiple ways of specifying the starting water; 1) by a fraction of a full profile, 2) by depth of wet soil or 3) a single value of plant available water.

The class represents a soil sample.

Calculates the average soil temperature at the centre of each layer, based on the soil temperature model of EPIC (Williams et al 1984) This code was separated from old SoilN - tidied up but not updated (RCichota, sep/2012)

5.1 SurfaceOrganicMatter

The surface organic matter model.

Encapsulates a list of residue types for SurfaceOrganicMatter model

5.2 MicroClimate

The module MICROMET, described here, has been developed to allow the calculation of potential transpiration for multiple competing canopies that can be either layered or intermingled.

This model controls irrigation events, which can be triggered using the Apply() method.

The fertiliser model

The Grapevine model is constructed from the following list of software components. Details of the implementation and model parameterisation are provided in the following sections.

List of Plant Model Components.

Component Name	Component Type

This class encapsulates an operations schedule.

The manager model

A report class for writing output to the data store.

Reads in weather data and makes it available to other models.

5.1 Clock

The clock model is resonsible for controlling the daily timestep in APSIM. It keeps track of the simulation date and loops from the start date to the end date, publishing events that other models can subscribe to.

5.1.1 Pre-timestep events (in order)

Events
DoInitialSummary
StartOfSimulation
CLEMInitialiseResource
CLEMInitialiseActivity

Events
CLEMValidate
FinalInitialise
StartOfMonth
StartOfYear

5.1.2 Timestep events (in order)

Events
DoWeather
DoDailyInitialisation
StartOfDay
DoManagement
DoPestDiseaseDamage
DoEnergyArbitration
DoSoilWaterMovement
DoSoilTemperature
DoSoilOrganicMatter
DoSurfaceOrganicMatterDecomposition
DoUpdateWaterDemand
DoWaterArbitration
PrePhenology
DoPhenology
DoPotentialPlantGrowth
DoPotentialPlantPartioning
DoNutrientArbitration
DoActualPlantPartioning
DoActualPlantGrowth
DoStock
DoLifecycle
DoUpdate
DoManagementCalculations
DoReportCalculations
EndOfDay
DoReport

5.1.3 Post-timestep events (in order)

Events
EndOfSimulation

This model collects the simulation initial conditions and stores into the DataStore. It also provides an API for writing messages to the DataStore.

The APSIM farming systems model has a long history of use for simulating mixed or intercropped systems. Doing this requires methods for simulating the competition of above and below ground resources. Above ground competition for light has been calculated within APSIM assuming a mixed turbid medium using the Beer-Lambert analogue as described by Keating et al., 1993. The MicroClimate Snow et al., 2004 model now used within APSIM builds upon this by also calculating the impact of mutual shading on canopy conductance and partitions aerodynamic conductance to individual species in applying the Penman-Monteith model for calculating potential crop water use. The arbitration of below ground resources of water and nitrogen is calculated by this model.

Traditionally, below ground competition has been arbitrated using two approaches. Firstly, the early approaches Adiku et al., 1995; Carberry et al., 1996 used an alternating order of uptake calculation each day to ensure that different crops within a simulation did not benefit from precedence in daily orders of calculations. Soil water simulations using the SWIM3 model Huth et al., 2012 arbitrate individual crop uptakes as part of the simulataneous solutions of various soil water fluxes as part of its solution of the Richards' equation Richards, 1931.

The soil arbitrator operates via a simple integration of daily fluxes into crop root systems via a Runge-Kutta calculation.

If Y is any soil resource, such as water or N, and U is the uptake of that resource by one or more plant root systems, then

$Y_{t+1} = Y_t - U$

Because U will change through the time period in complex manners depending on the number and nature of demands for that resource, we use Runge-Kutta to integrate through that time period using

$Y_{t+1} = Y_t + 1/6 \times (U_1 + 2xU_2 + 2xU_3 + U_4)$

Where U_1,U_2,U_3 and U_4 are 4 estimates of the Uptake rates calculated by the crop models given a range of soil resource conditions, as follows:

$U_1 = f(Y_t)$,

$U_2 = f(Y_t - 0.5xU_1)$,

$U_3 = f(Y_t - 0.5xU_2)$,

$U_4 = f(Y_t - U_3)$.

So U_1 is the estimate based on the uptake rates at the beginning of the time interval, similar to a simple Euler method. U_2 and U_3 are estimates based on the rates somewhere near the midpoint of the time interval. U_4 is the estimate based on the rates toward the end of the time interval.

The iterative procedure allows crops to influence the uptake of other crops via various feedback mechanisms. For example, crops rapidly extracting water from near the surface will dry the soil in those layers, which will force deeper rooted crops to potentially extract water from lower layers. Uptakes can notionally be of either sign, and so trees providing hydraulic lift of water from water tables could potentially make this water available for uptake by mutplie understory species within the timestep. Crops are responsible for meeting resource demand by whatever means they prefer. And so, leguminous crops may start by taking up mineral N at the start of the day but rely on fixation later in a time period if N becomes limiting. This will reduce competition from others and change the balance dynamically throughout the integration period.

The design has been chosen to provide the following benefits:

1) The approach is numerically simple and pure.

2) The approach does not require the use of any particular uptake equation. The uptake equation is embodied within the crop model as designed by the crop model developer and tester.

3) The approach will allow any number of plant species to interact.

4) The approach will allow for arbitration between species in any zone, but also competition between species that may demand resources from multiple zones within the simulation.

5) The approach will automatically arbitrate supply of N between zones, layers, and types (nitrate vs ammonium) with the preferences of all derived by the plant model code.

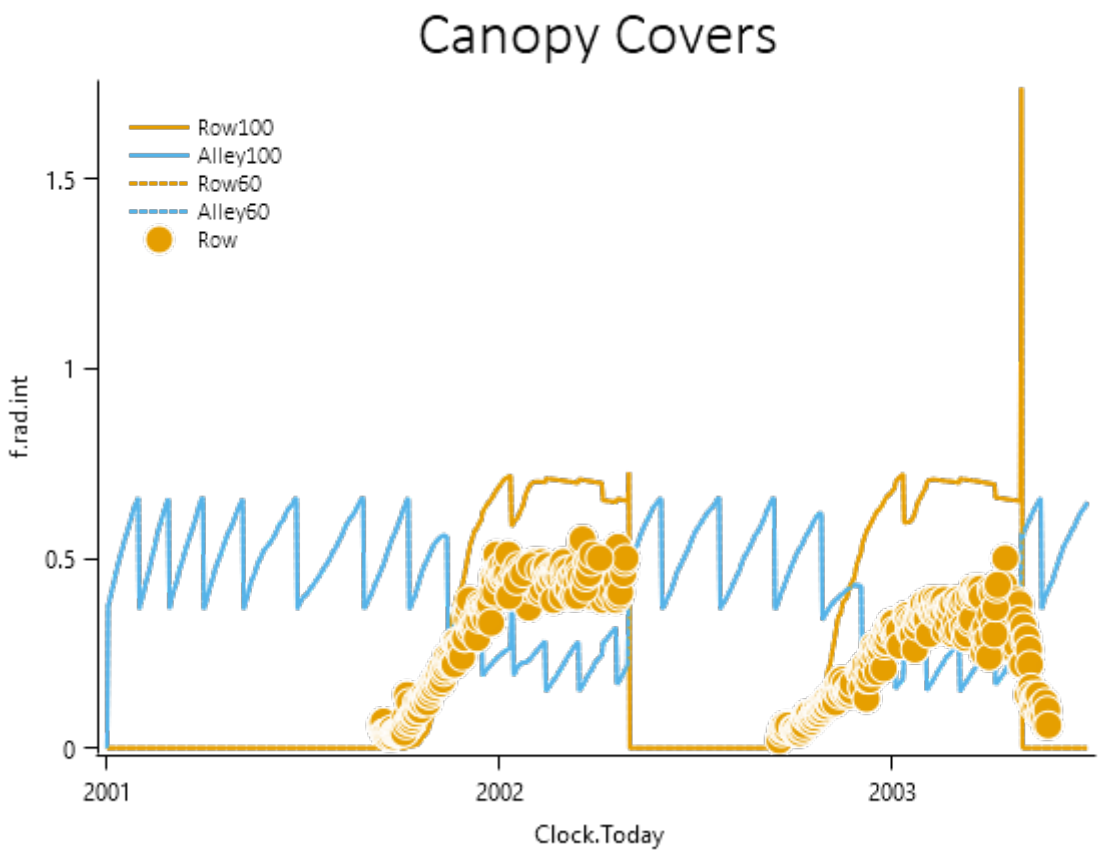
6 Validation

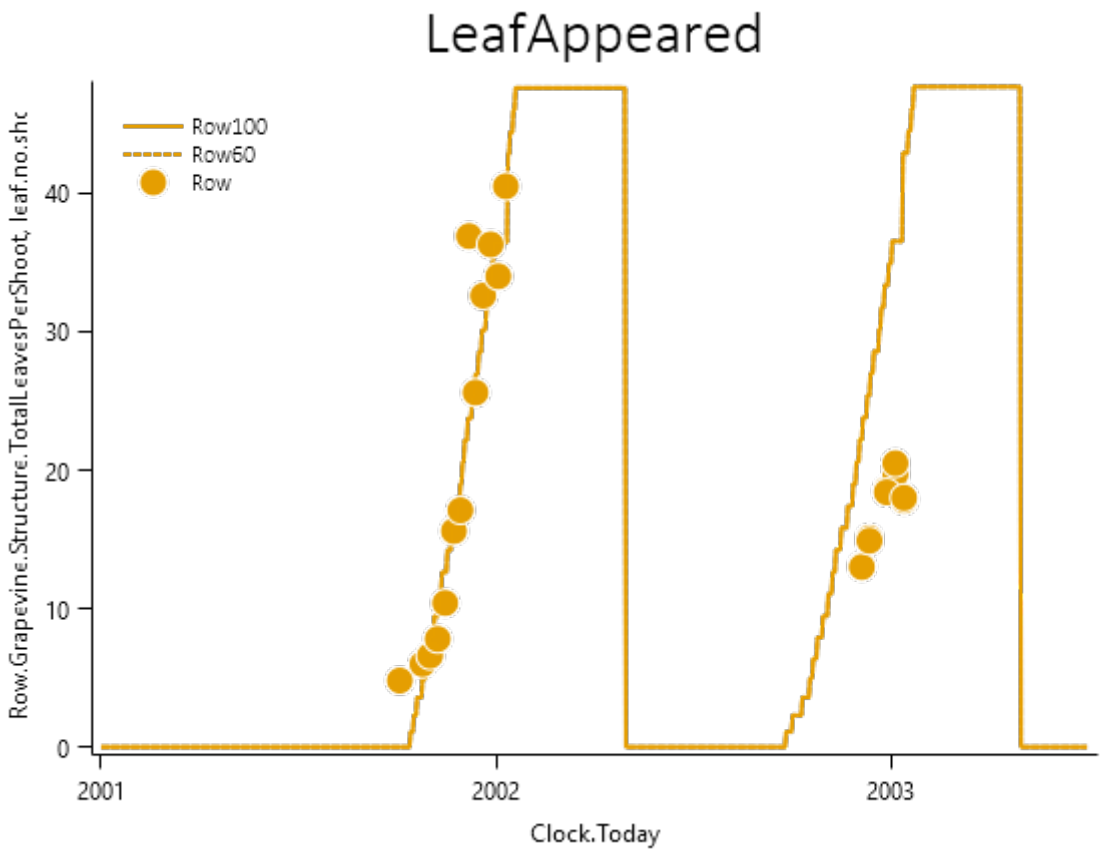
6.1 New Zealand

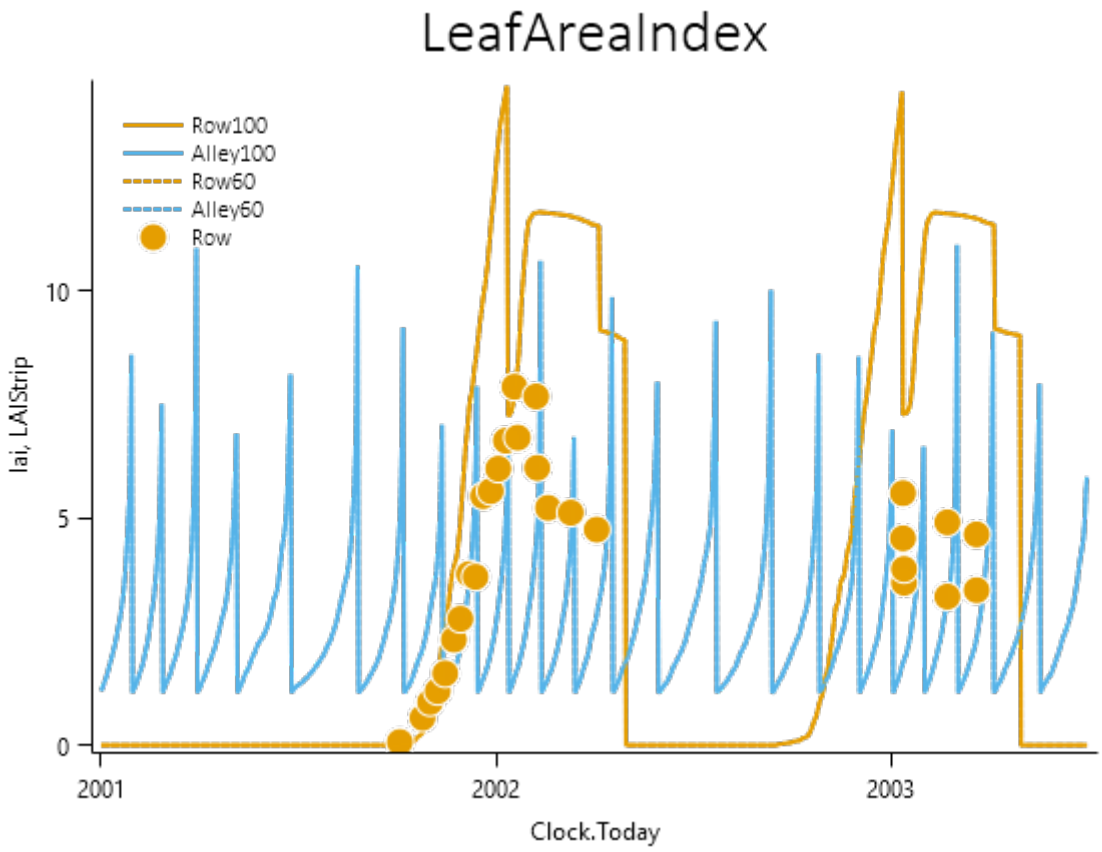
List of experiments.

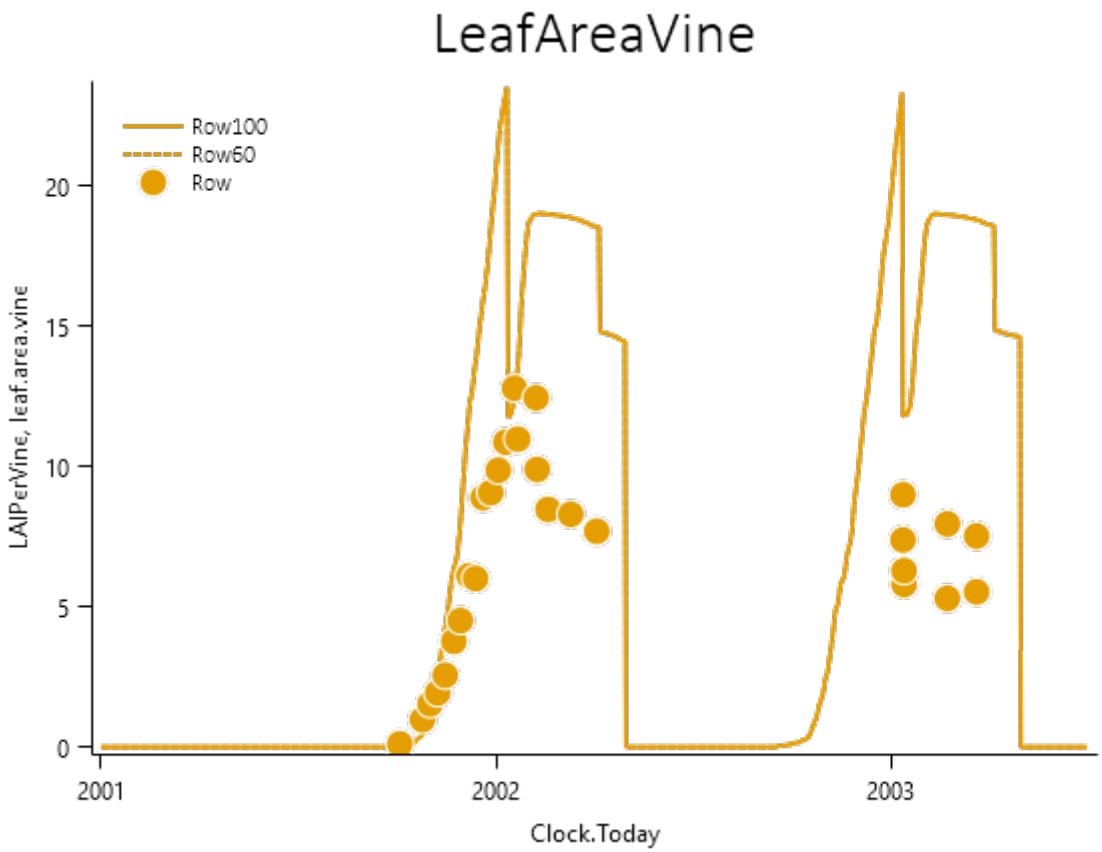
Experiment Name	Design (Number of Treatments)
Marlborough	Climate (4)
Grape_Squire_1	Irri (2)
Grape_Squire_2	Irri (3)

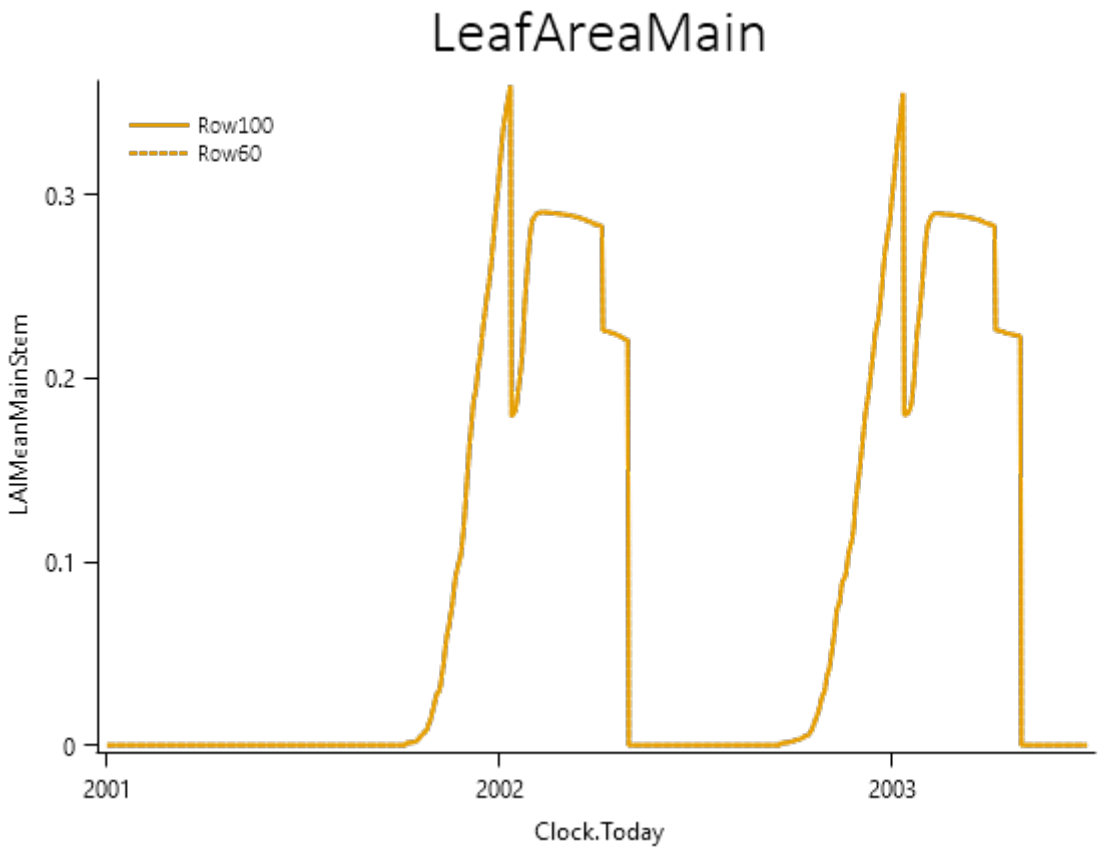
6.1.1 Grape_Squire_1

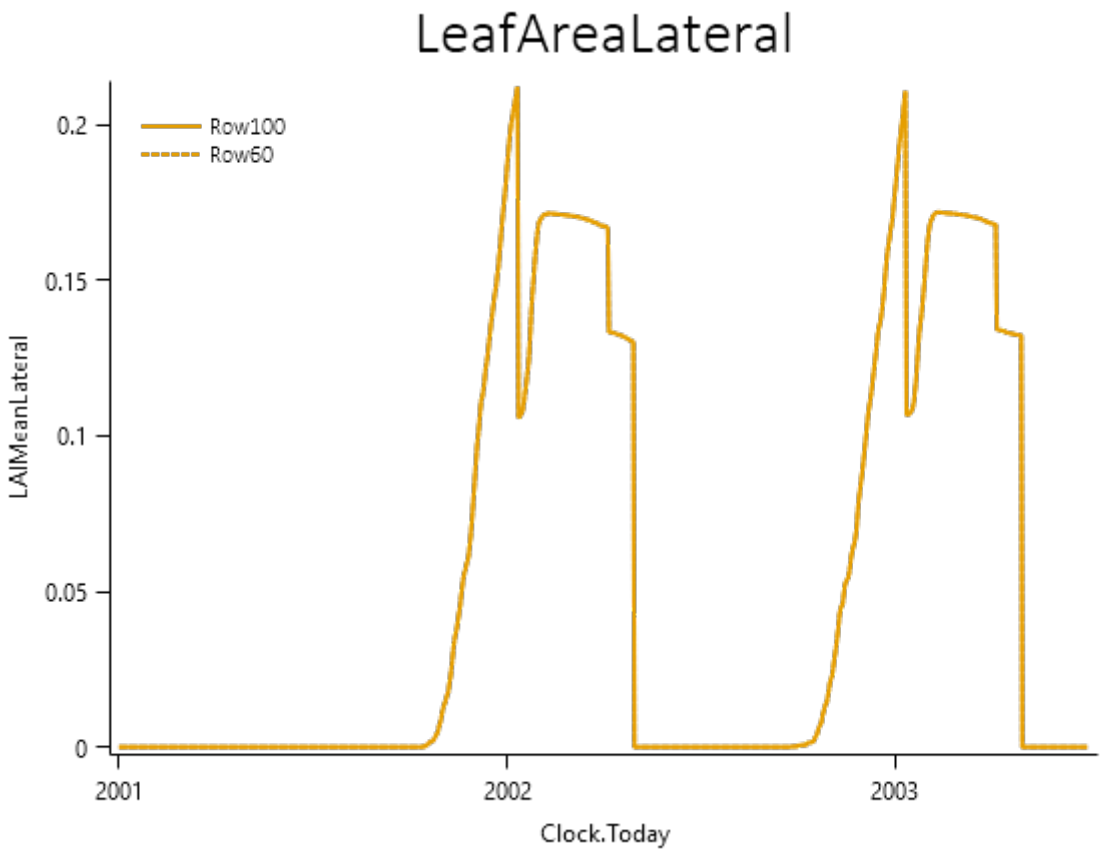


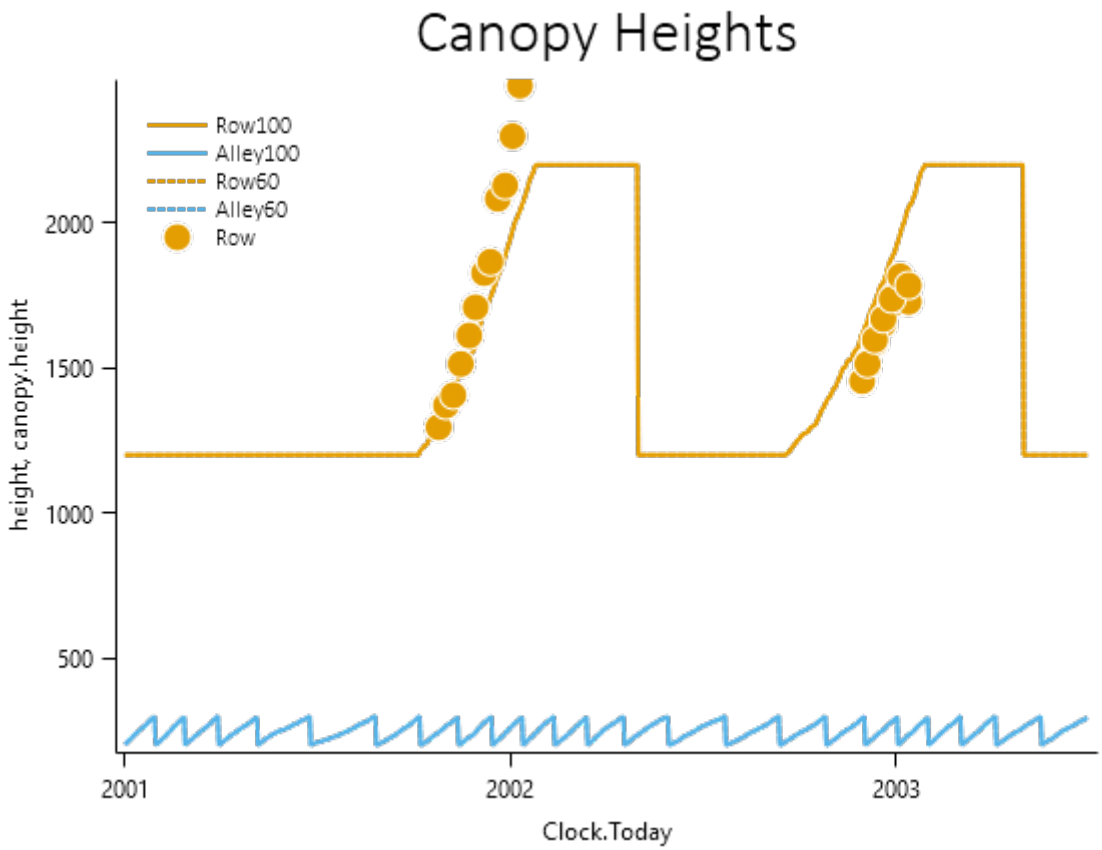

LeafAppeared
Row100
Row60
Row
Row.Grapevine.Structure.TotalLeavesPerShoot, leaf.no.shc
0
10
20
30
40
2001
2002
2003
Clock.Today


LeafAreaIndex
Row100
Alley100
Row60
Alley60
Row
lai, LAIStrip
0
5
10
2001
2002
2003
Clock.Today

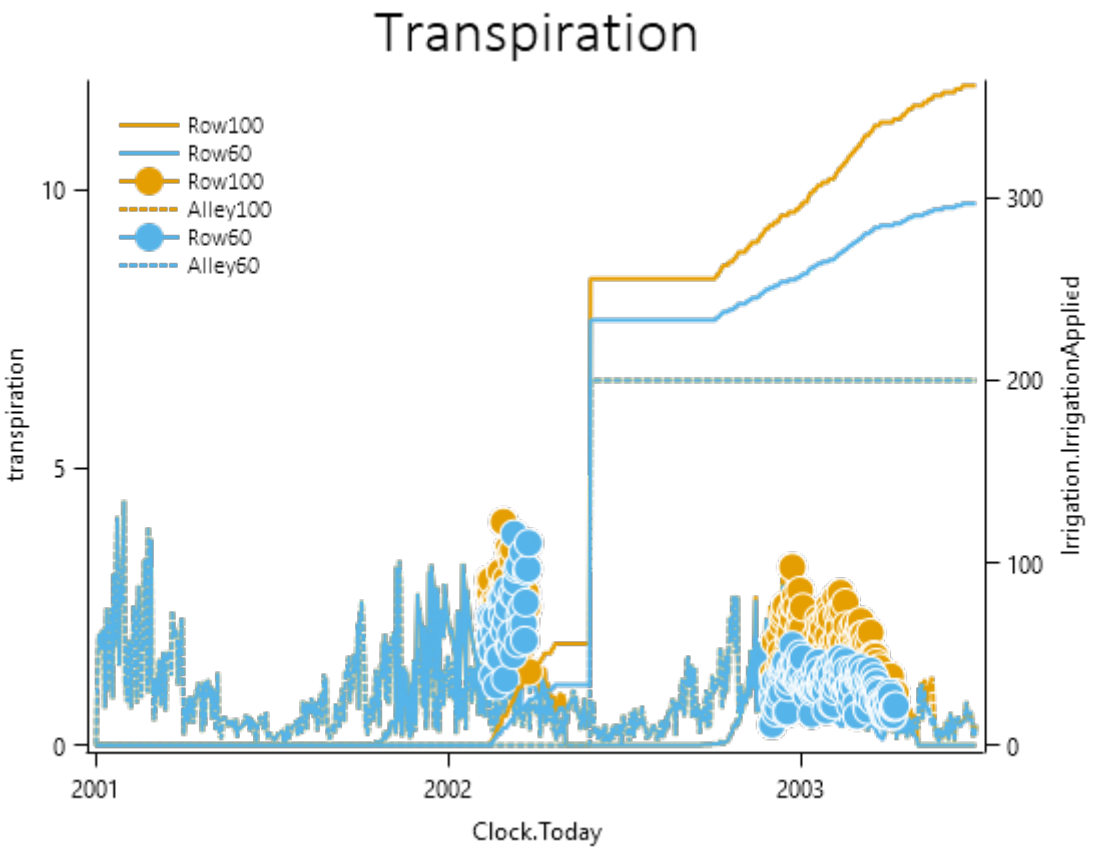

LeafAreaVine
Row100
Row60
Row
LAIPerVine, leaf.area.vine
20
15
10
5
0
2001
2002
2003
Clock.Today

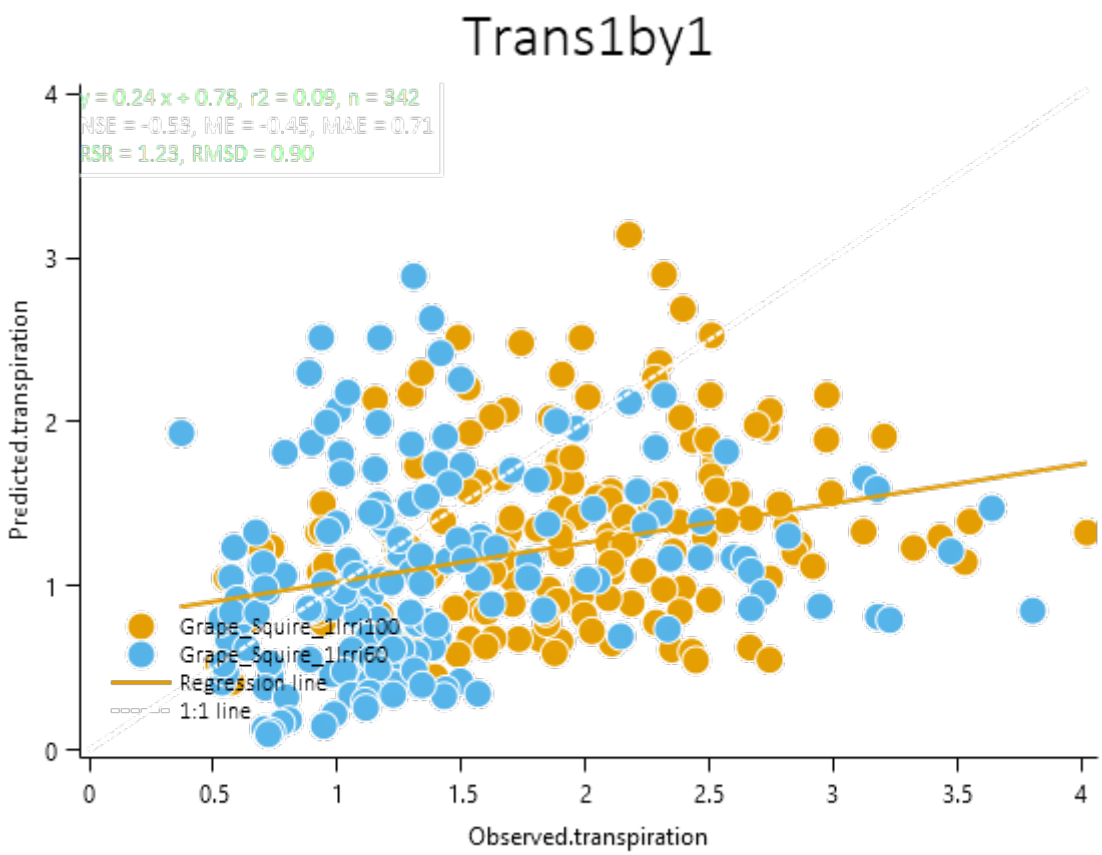

LeafAreaMain
Row100
Row60
LAIMeanMainStem
0.3
0.2
0.1
0
2001
2002
2003
Clock.Today


LeafAreaLateral
Row100
Row60
0.2
0.15
0.1
0.05
0
LAIMeanLateral
2001
2002
2003
Clock.Today

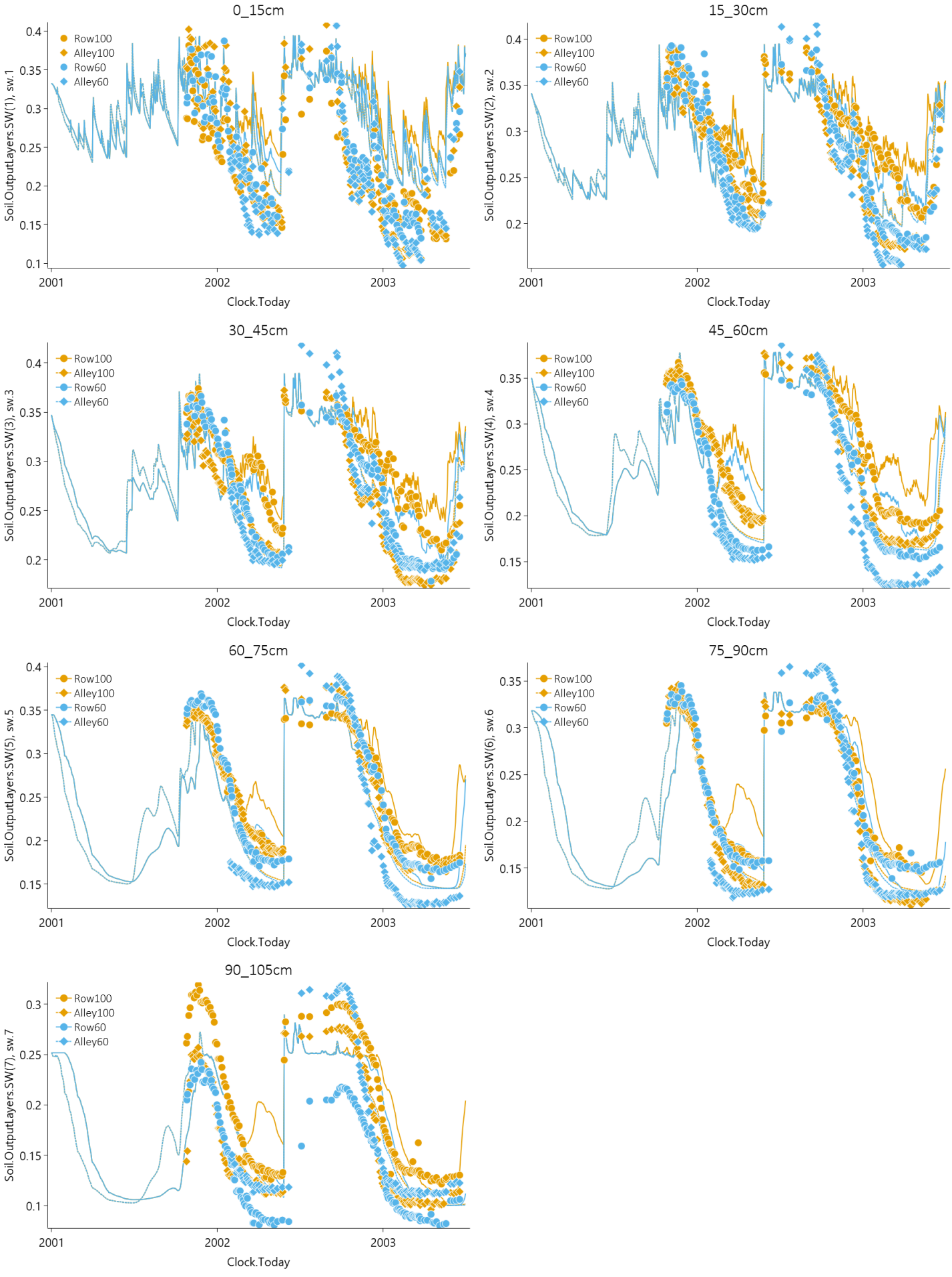

Canopy Heights
Row100
Alley100
Row60
Alley60
Row
2000
1500
1000
500
height, canopy.height
2001
2002
2003
Clock.Today

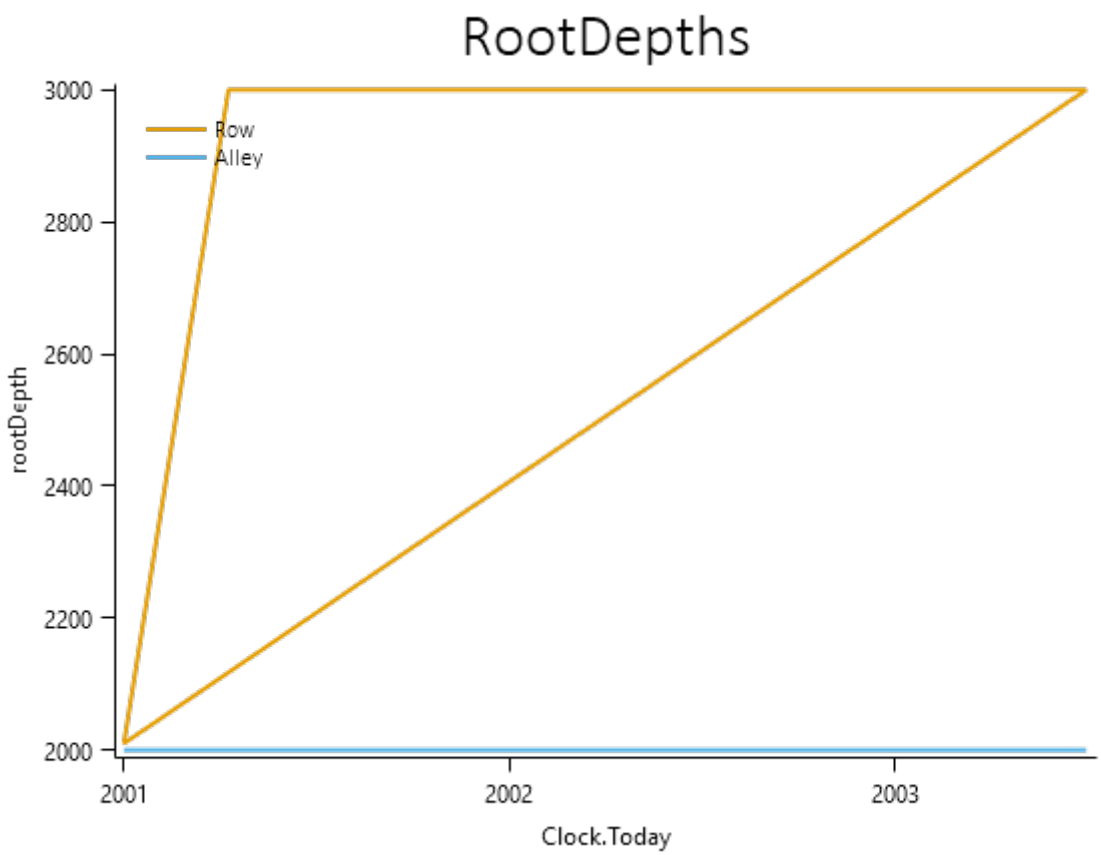

psw
Row100
Alley100
Row60
Alley60
350
300
250
200
150
psw
2001
2002
2003
Clock.Today

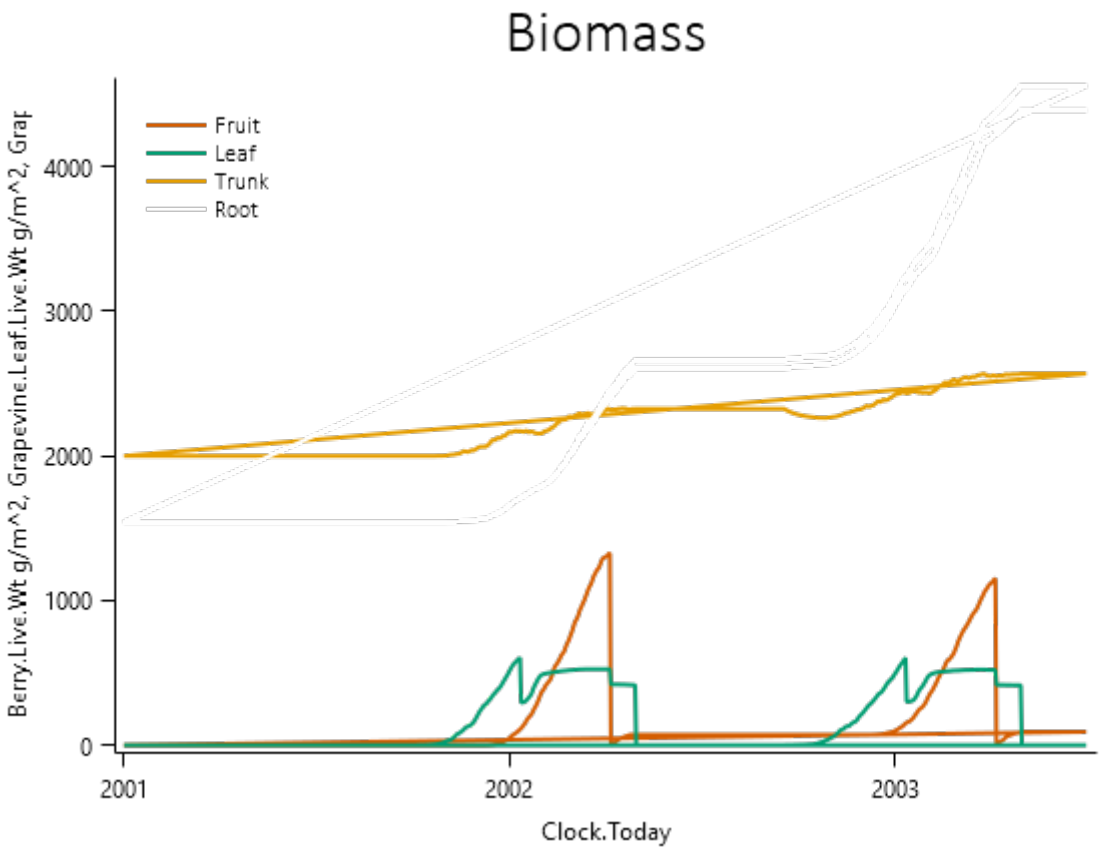

Transpiration
Row100
Row60
Row100
Alley100
Row60
Alley60
10
5
0
transpiration
300
200
100
0
Irrigation.IrrigationApplied
2001
2002
2003
Clock.Today

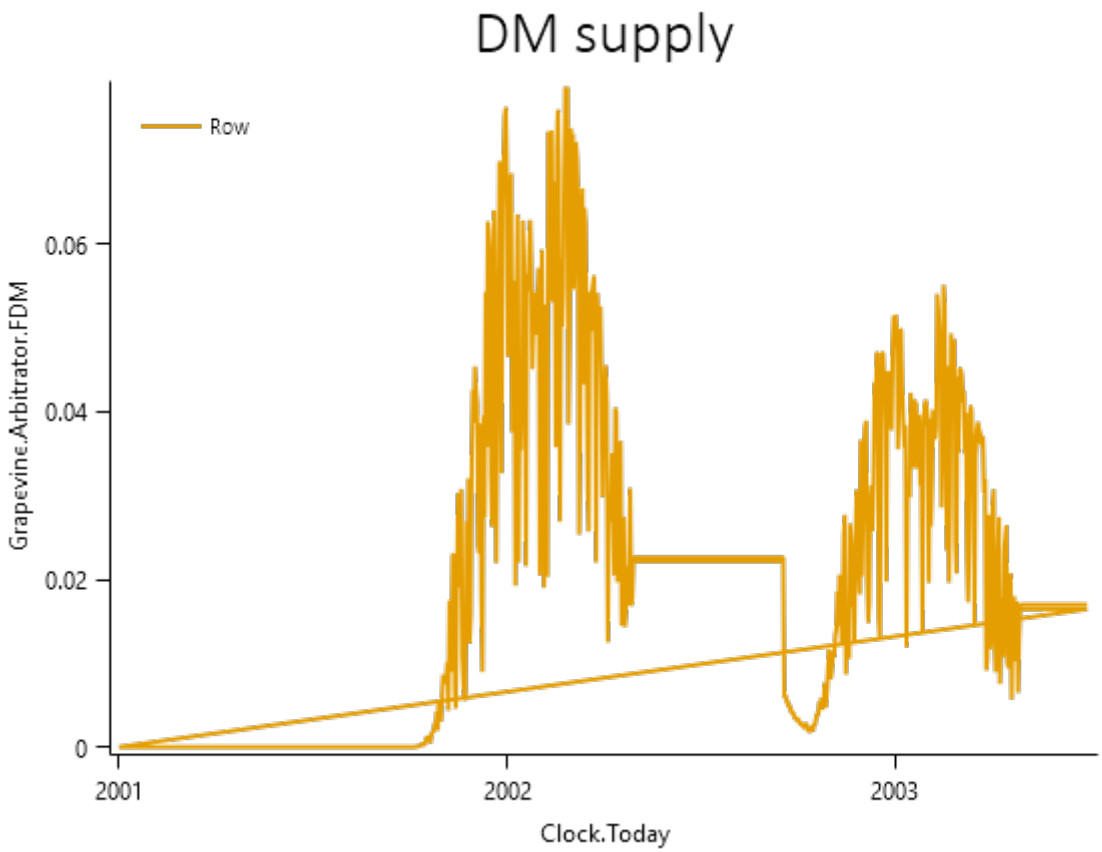

Trans1by1
y = 0.24 x + 0.78, r2 = 0.09, n = 342
NSE = -0.53, ME = -0.45, MAE = 0.71
RSR = 1.23, RMSD = 0.90
Predicted.transpiration
Observed.transpiration
Grape_Squire_1Irri100
Grape_Squire_1Irri60
Regression line
1:1 line
0
1
2
3
4
0.5
1.5
2.5
3.5

6.1.1.1 SoilWater

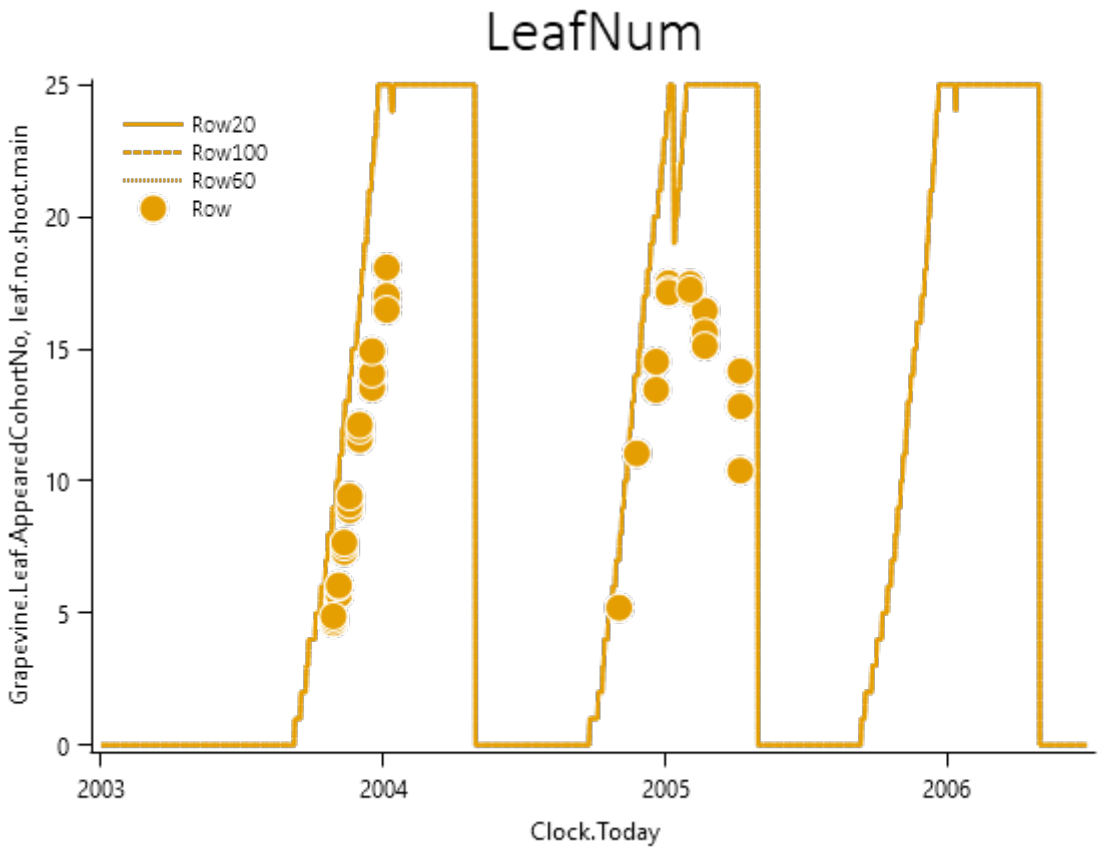


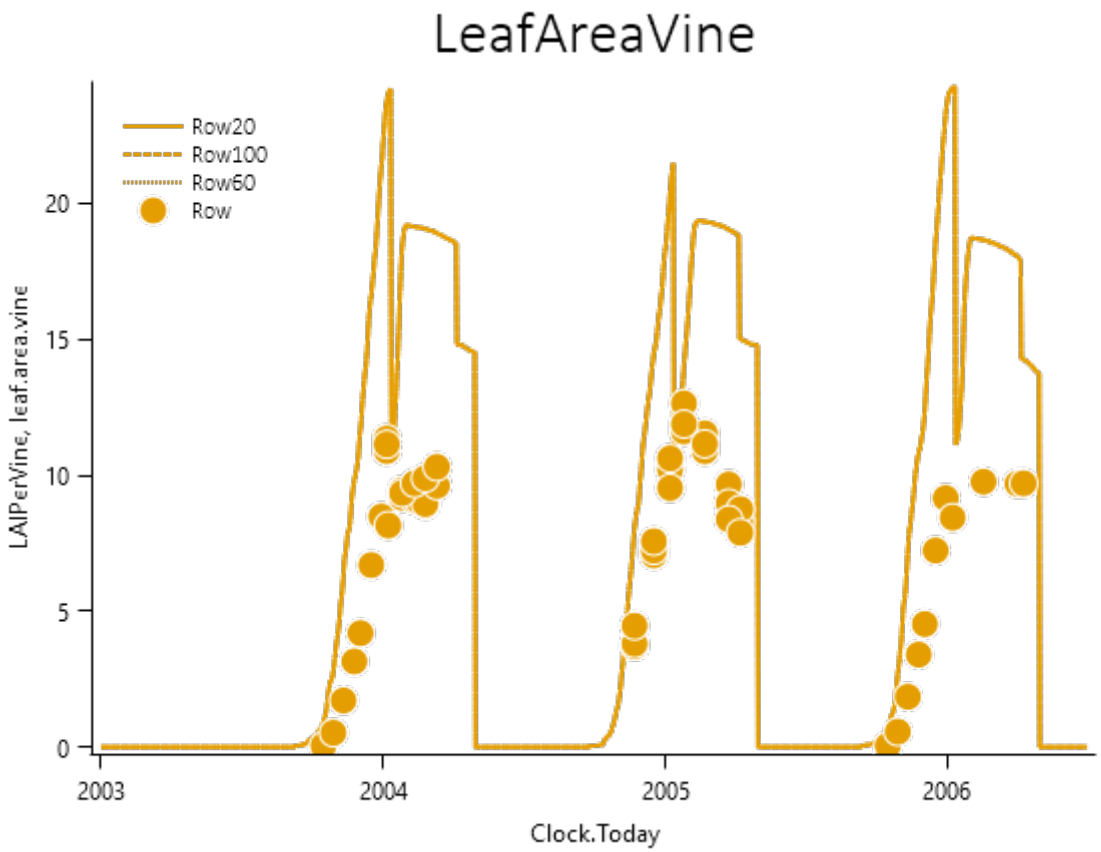

RootDepths
Row
Alley
rootDepth
3000
2800
2600
2400
2200
2000
2001
2002
2003
Clock.Today

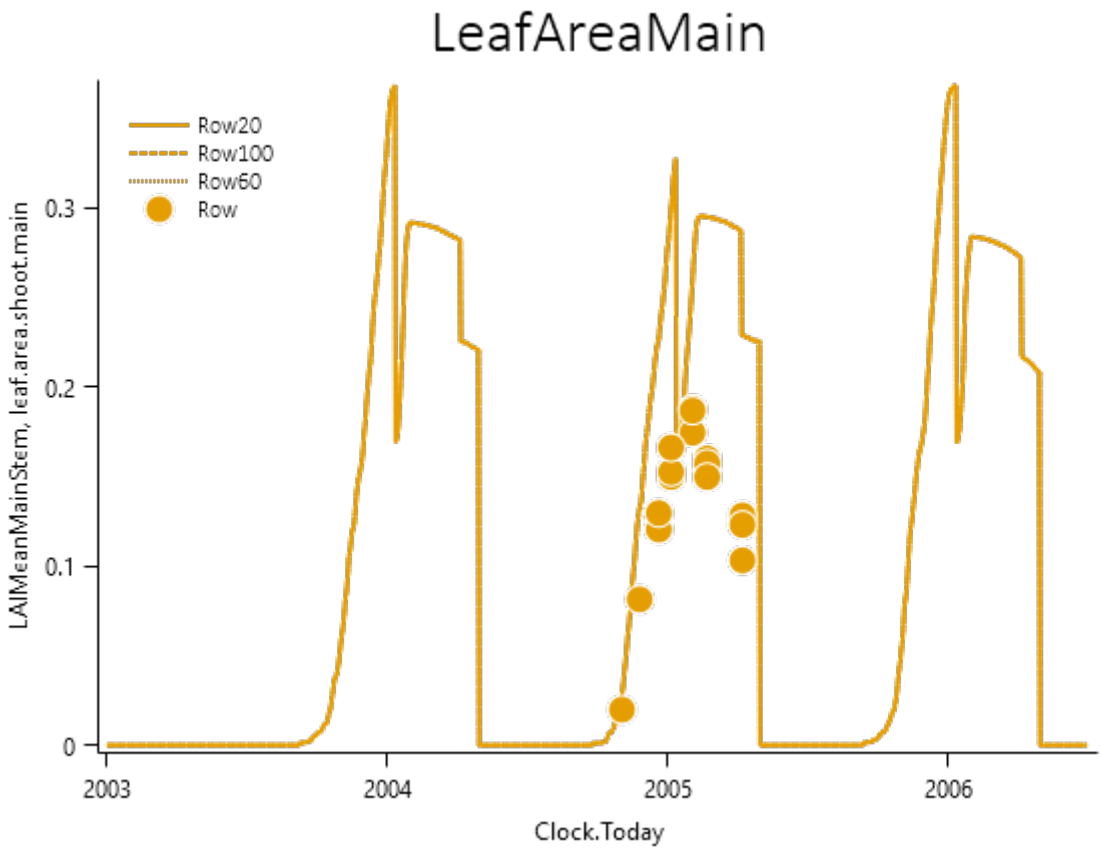

Biomass
Fruit
Leaf
Trunk
Root
Berry.Live.Wt g/m^2, Grapevine.Leaf.Live.Wt g/m^2, Grap
4000
3000
2000
1000
0
2001
2002
2003
Clock.Today

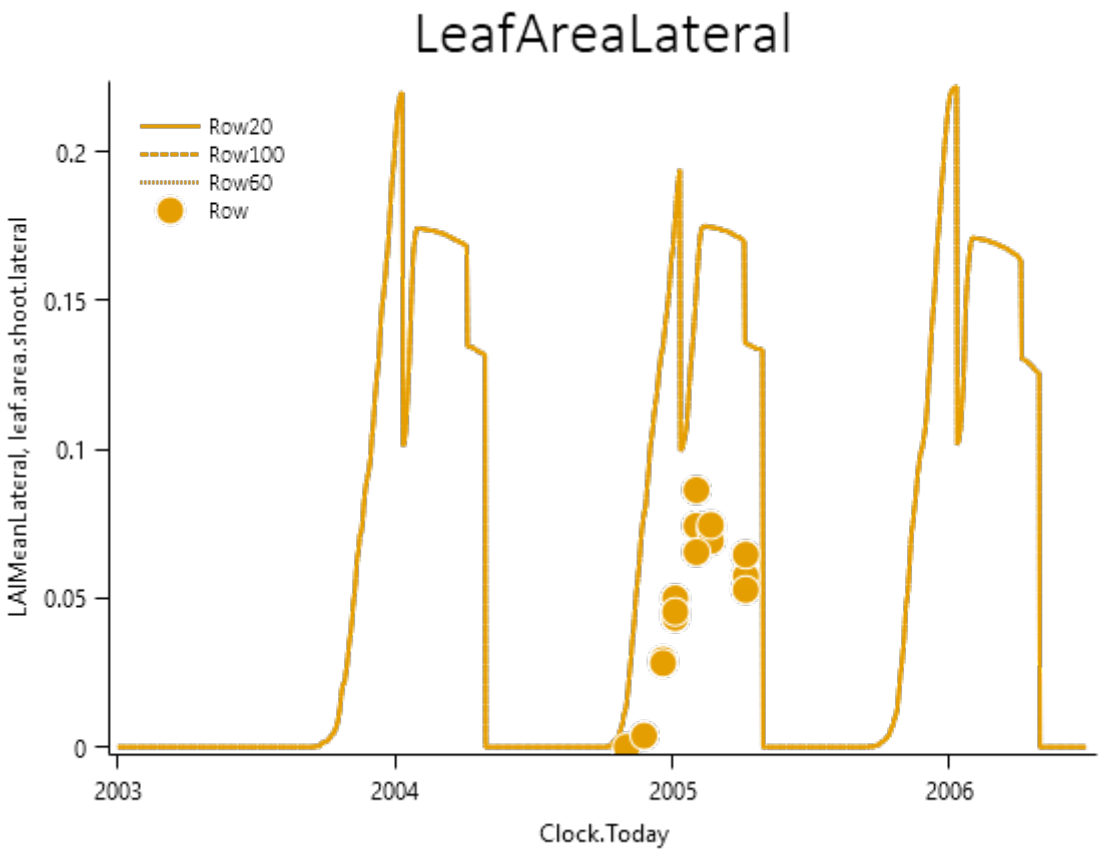


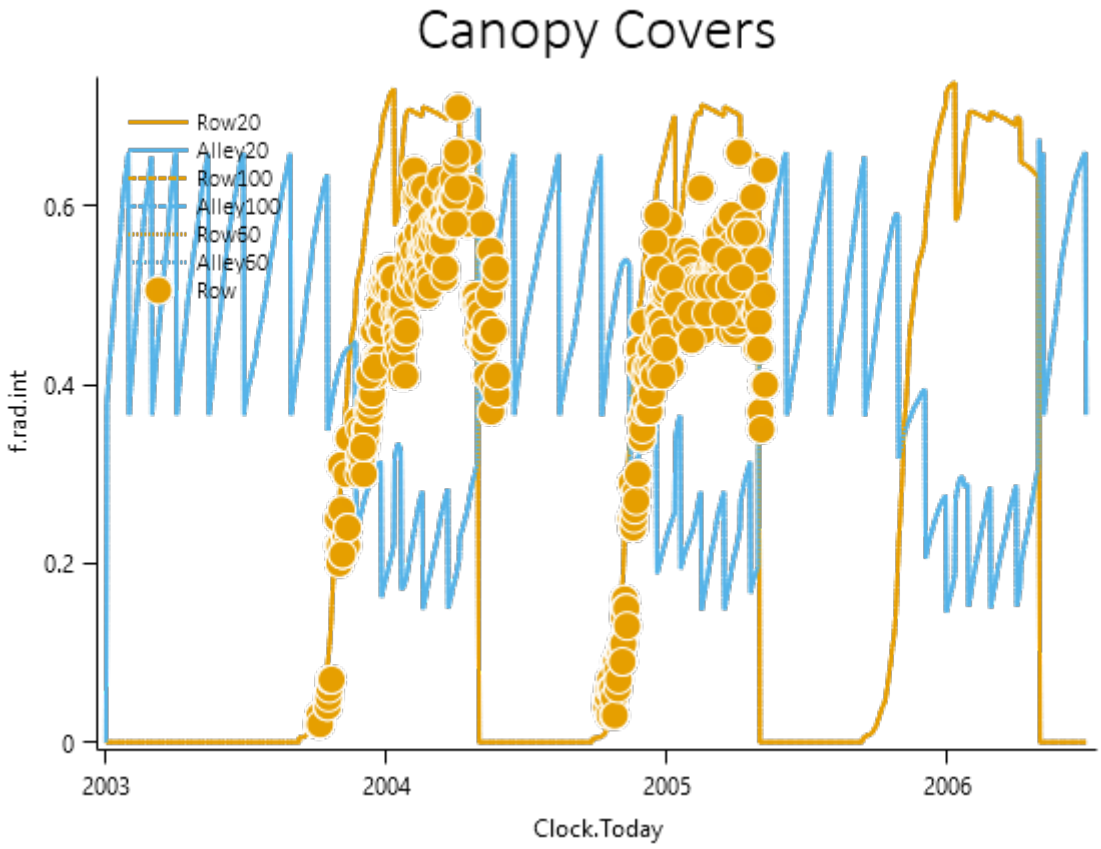
6.1.2 Grape_Squire_2

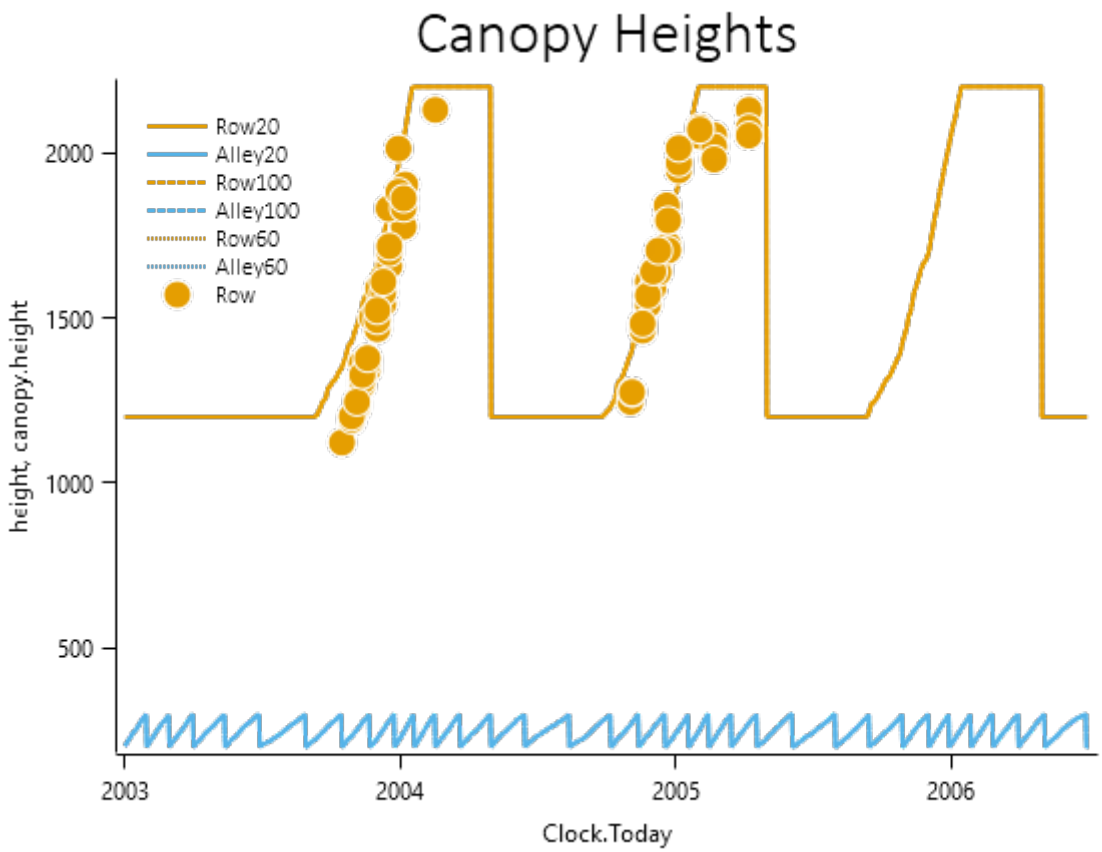


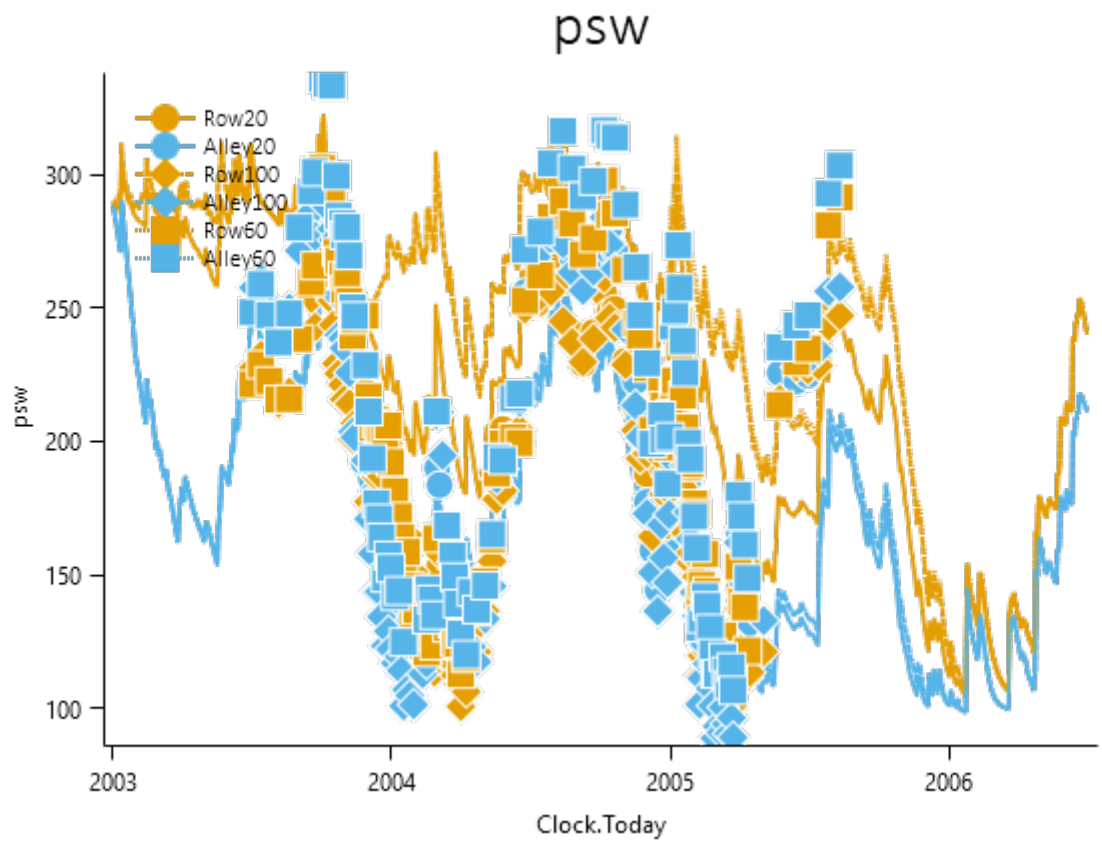

LeafAreaVine
Row20
Row100
Row60
Row
LAIPerVine, leaf.area.vine
0
5
10
15
20
2003
2004
2005
2006
Clock.Today

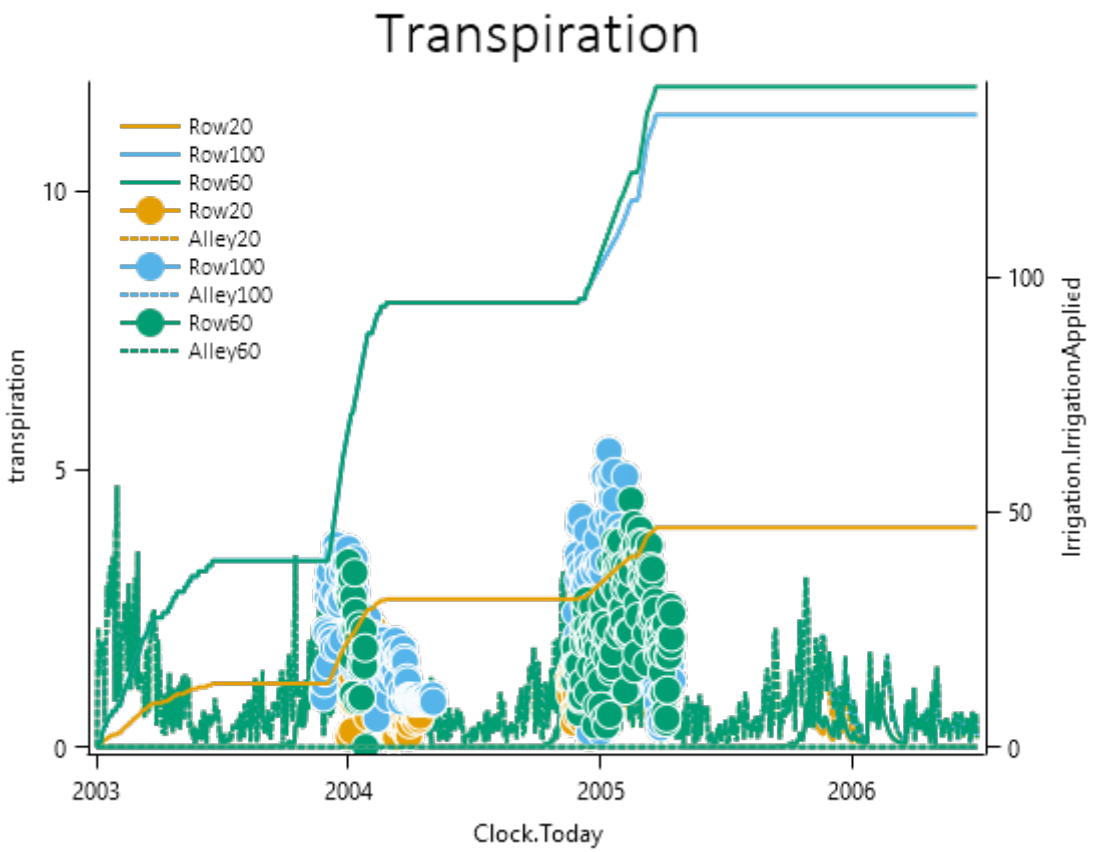

LeafAreaMain
Row20
Row100
Row60
Row
LAIMeanMainStem, leaf.area.shoot.main
0
0.1
0.2
0.3
2003
2004
2005
2006
Clock.Today

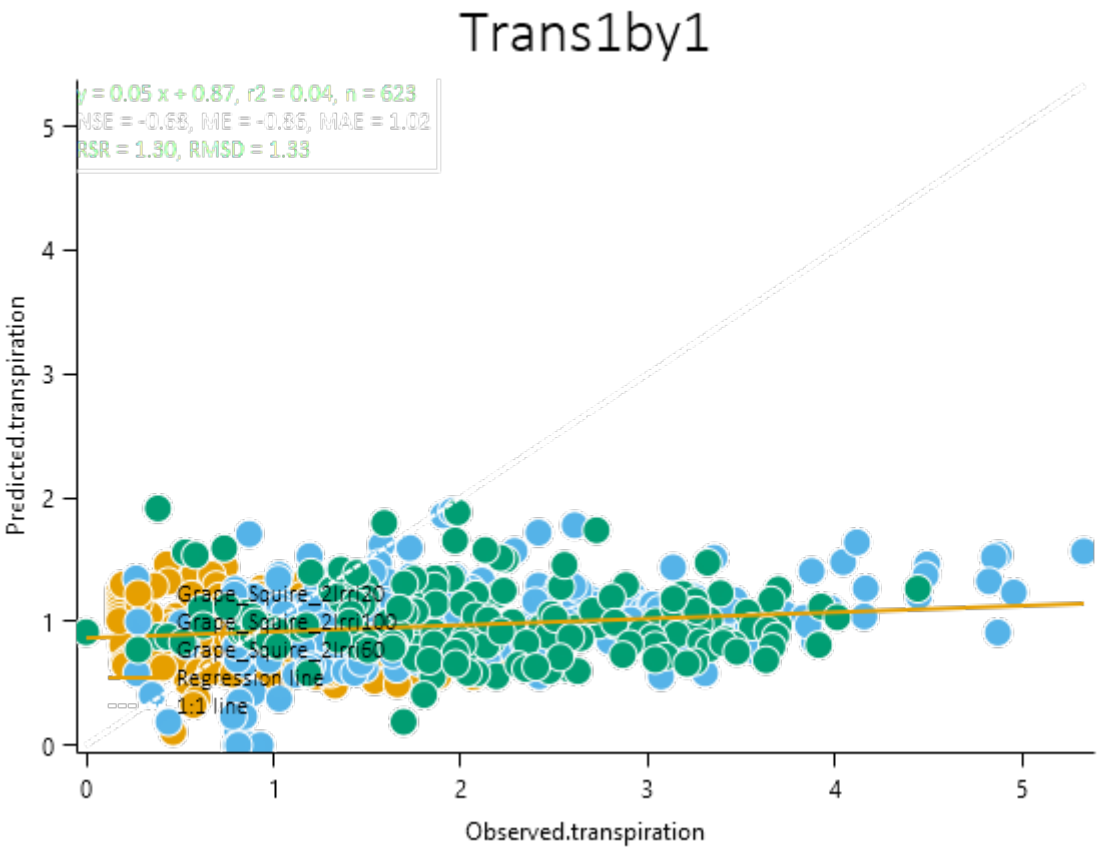

LeafAreaLateral
Row20
Row100
Row60
Row
LAIMeanLateral, leaf.area.shoot.lateral
0.2
0.15
0.1
0.05
0
2003
2004
2005
2006
Clock.Today


Canopy Covers
Row20
Alley20
Row100
Alley100
Row60
Alley60
Row
f.rad.int
0.6
0.4
0.2
0
2003
2004
2005
2006
Clock.Today

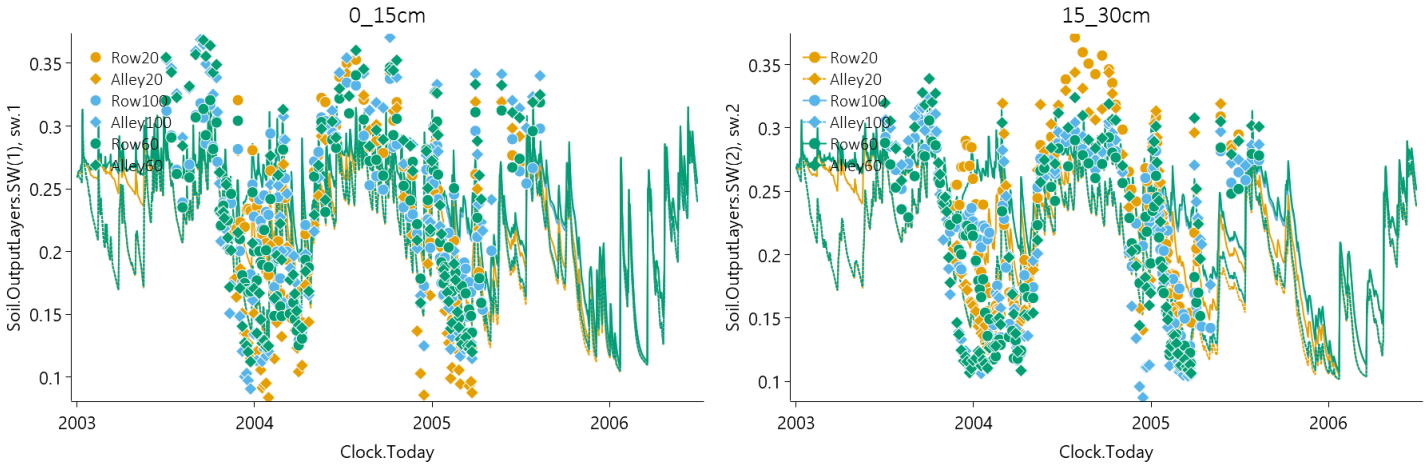

Canopy Heights
Row20
Alley20
Row100
Alley100
Row60
Alley60
Row
height, canopy.height
2000
1500
1000
500
2003
2004
2005
2006
Clock.Today

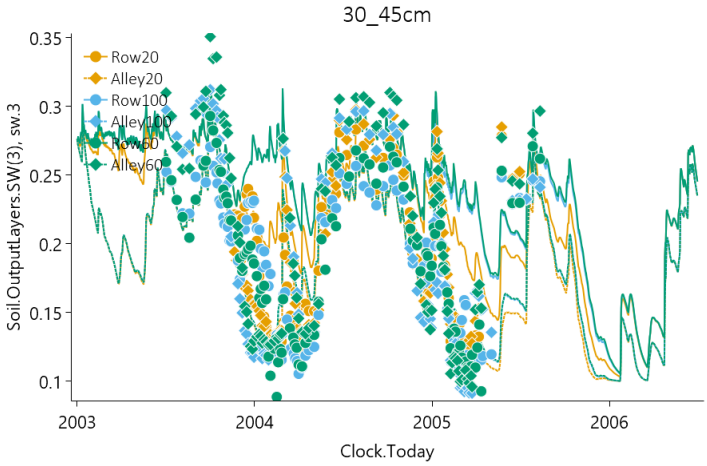

psw
Row20
Alley20
Row100
Alley100
Row60
Alley60
psw
300
250
200
150
100
2003
2004
2005
2006
Clock.Today

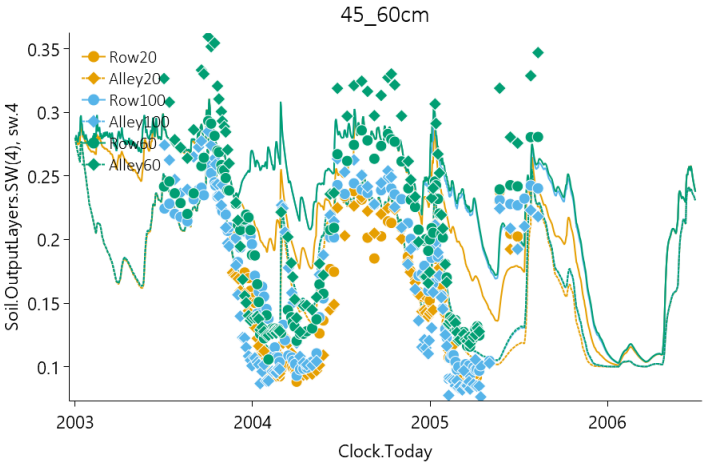

Transpiration
Row20
Row100
Row60
Row20
Alley20
Row100
Alley100
Row60
Alley60
transpiration
Irrigation.IrrigationApplied
Clock.Today
2003
2004
2005
2006
0
5
10
0
50
100

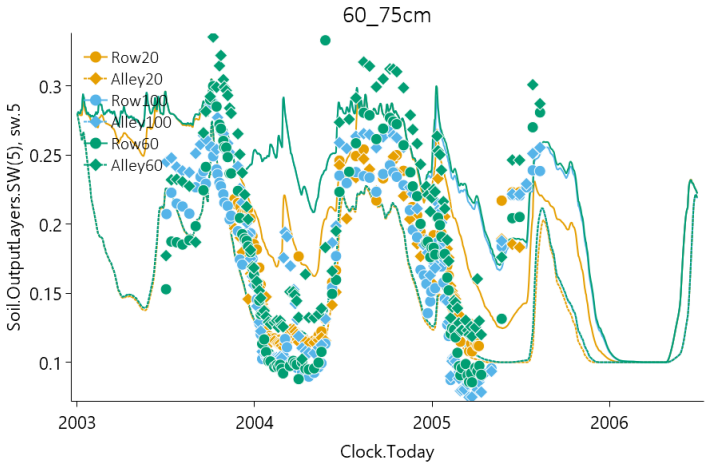

Trans1by1
y = 0.05 x + 0.87, r2 = 0.04, n = 623
NSE = -0.68, ME = -0.86, MAE = 1.02
RSR = 1.30, RMSD = 1.33
Grape_Squire_2Irr20
Grape_Squire_2Irr100
Grape_Squire_2Irr60
Regression line
1:1 line
Predicted.transpiration
Observed.transpiration
0
1
2
3
4
5

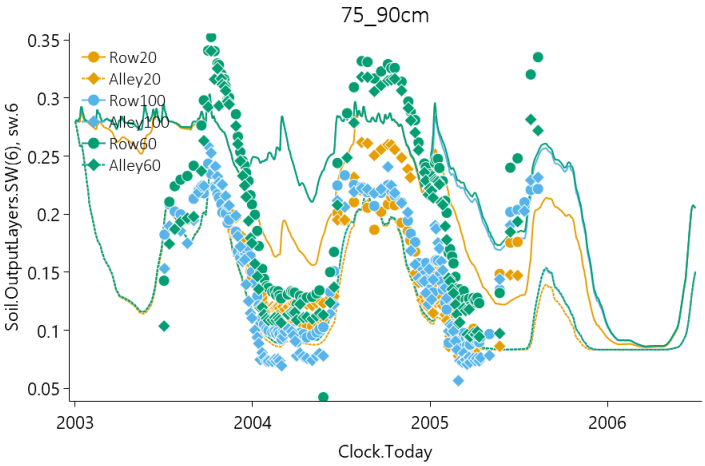
6.1.2.1 SoilWater

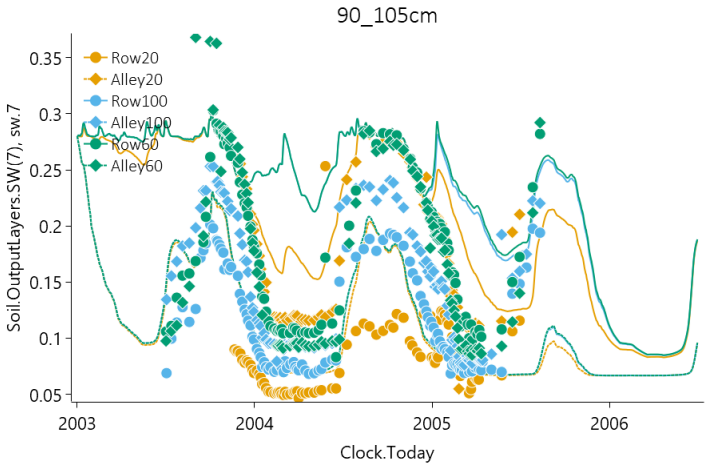


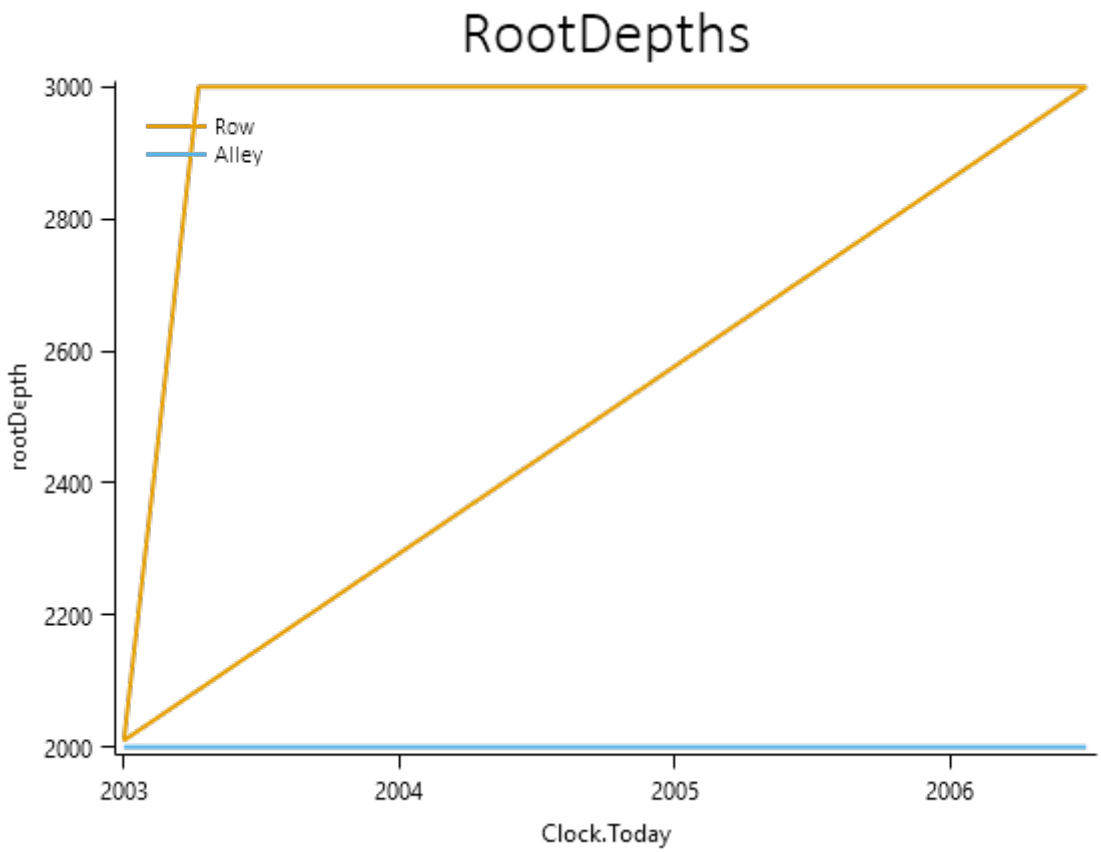


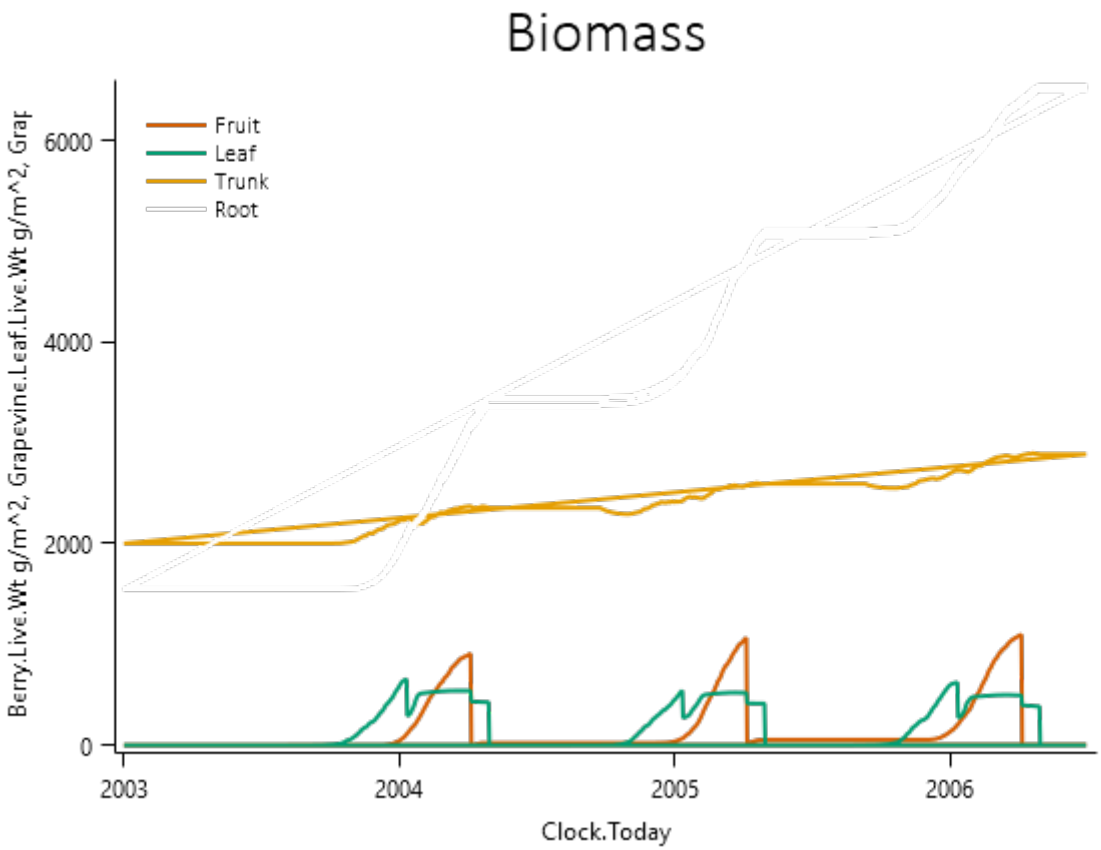


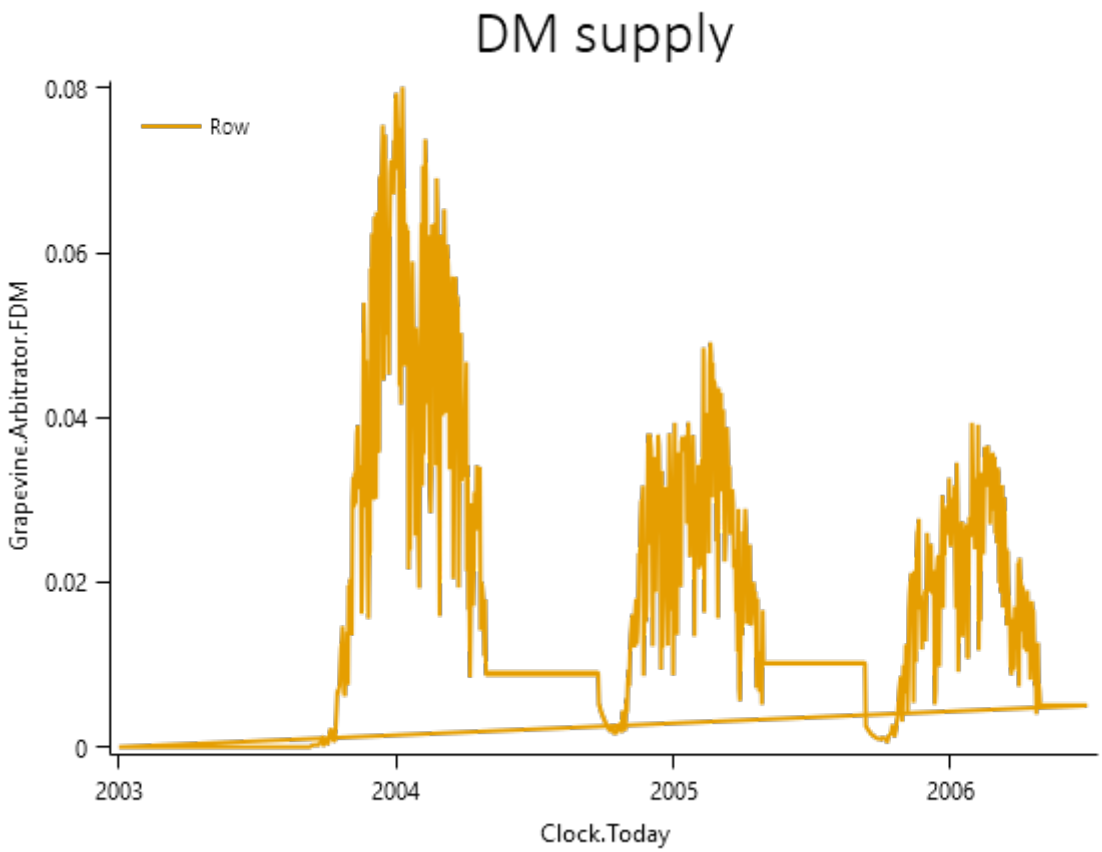


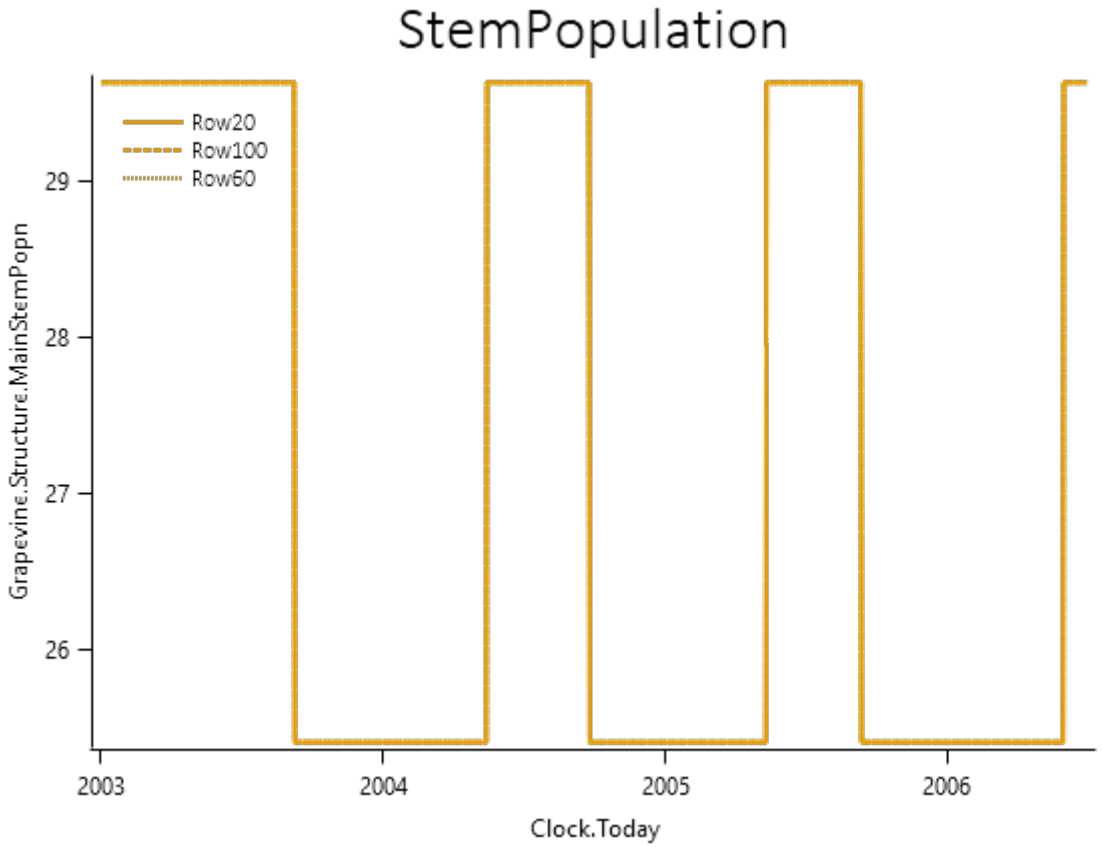





RootDepths
Row
Alley
3000
2800
2600
2400
2200
2000
rootDepth
2003
2004
2005
2006
Clock.Today


Biomass
Fruit
Leaf
Trunk
Root
6000
4000
2000
0
Berry.Live.Wt g/m^2, Grapevine.Leaf.Live.Wt g/m^2, Grap
2003
2004
2005
2006
Clock.Today





6.2 Statistics

6.2.1 PredictedObserved

Variable	n	Slope	Intercept	R2	RMSE	NSE	ME	MAE
BerryMass	47	0.435	1.428	0.158	0.313	-4.705	0.274	0.283

Variable	n	Slope	Intercept	R2	RMSE	NSE	ME	MAE
BerryNum	46	0.194	51.280	0.026	16.205	-1.045	1.083	13.276
BudBurstDOY	65	0.823	16.932	0.181	10.740	-2.174	-1.631	8.923
BunchesPerVine	47	0.239	49.174	0.348	7.206	0.314	0.030	5.480
f.rad.int	1337	1.064	0.146	0.499	0.234	-1.472	0.172	0.206
FloweringDOY	65	0.386	103.968	0.240	7.474	0.124	-1.338	5.738
psw	608	0.482	159.653	0.547	65.104	-0.067	49.066	54.776
transpiration	965	0.081	0.894	0.032	1.193	-0.630	-0.716	0.912
VeraisonDOY	65	0.448	129.306	0.277	8.494	0.152	-1.262	6.462

7 Optimization

List of experiments.

Experiment Name	Design (Number of Treatments)
ParameterOptimize	(4)

8 References

Adiku, S. G. K., Carberry, P. S., Rose, C. W., McCown, R. L., Braddock, R., 1995. A maize (zea-mays) - cowpea (vigna-unguiculata) intercrop model. Ecophysiology of Tropical Intercropping, Inst Natl Recherche Agronomique, Paris, 397-406.

Brown, H.E, Huth, N, Holzworth, D., 2011. A potato model build using the APSIM Plant.NET framework., 961-967.

Brown, Hamish E., Huth, Neil I., Holzworth, Dean P., Teixeira, Edmar I., Zyskowski, Rob F., Hargreaves, John N. G., Moot, Derrick J., 2014. Plant Modelling Framework: Software for building and running crop models on the APSIM platform. Environmental Modelling and Software 62, 385-398.

Carberry, P.S., McCown, R. L., Muchow, R. C., Dimes, J. P., Probert, M. E., 1996. Simulation of a legume ley farming system in northern Australia using the Agricultural Production Systems Simulator. Australian Journal of Experimental Agriculture 36 (8), 1037-1048.

Holzworth, Dean P., Huth, Neil I., deVoil, Peter G., Zurcher, Eric J., Herrmann, Neville I., McLean, Greg, Chenu, Karine, van Oosterom, Erik J., Snow, Val, Murphy, Chris, Moore, Andrew D., Brown, Hamish, Whish, Jeremy P. M., Verrall, Shaun, Fainges, Justin, Bell, Lindsay W., Peake, Allan S., Poulton, Perry L., Hochman, Zvi, Thorburn, Peter J., Gaydon, Donald S., Dalgliesh, Neal P., Rodriguez, Daniel, Cox, Howard, Chapman, Scott, Doherty, Alastair, Teixeira, Edmar, Sharp, Joanna, Cichota, Rogerio, Vogeler, Iris, Li, Frank Y., Wang, Enli, Hammer, Graeme L., Robertson, Michael J., Dimes, John P., Whitbread, Anthony M., Hunt, James, van Rees, Harm, McClelland, Tim, Carberry, Peter S., Hargreaves, John N. G., MacLeod, Neil, McDonald, Cam, Harsdorf, Justin, Wedgwood, Sara, Keating, Brian A., 2014. APSIM – Evolution towards a new generation of agricultural systems simulation. Environmental Modelling and Software 62, 327-350.

Huth, N.I., Bristow, K.L., Verburg, K., 2012. SWIM3: Model use, calibration, and validation. Transactions of the ASABE 55 (4), 1303-1313.

Keating, B. A., Carberry, P. S., 1993. Resource capture and use in inter cropping - solar radiation. Field Crops Research 34 (3-4), 273-301.

Keating, B. A., Carberry, P. S., Hammer, G. L., Probert, M. E., Robertson, M. J., Holzworth, D., Huth, N. I., Hargreaves, J. N. G., Meinke, H., Hochman, Z., McLean, G., Verburg, K., Snow, V., Dimes, J. P., Silburn, M., Wang, E., Brown, S., Bristow, K. L., Asseng, S., Chapman, S., McCown, R. L., Freebairn, D. M.,

Smith, C. J., 2003. An overview of APSIM, a model designed for farming systems simulation. European Journal of Agronomy 18 (3-4), 267-288.

Lawless, Conor, Semenov, MA, Jamieson, PD, 2005. A wheat canopy model linking leaf area and phenology. European Journal of Agronomy 22 (1), 19-32.

McCown, R. L., Hammer, G. L., Hargreaves, J. N. G., Holzworth, D., Huth, N. I., 1995. APSIM: an agricultural production system simulation model for operational research. Mathematics and Computers in Simulation 39 (3-4), 225-231.

McCown, R. L., Hammer, G. L., Hargreaves, J. N. G., Holzworth, D. P., Freebairn, D. M., 1996. APSIM: a Novel Software System for Model Development, Model Testing and Simulation in Agricultural Systems Research. Agricultural Systems 50 (3), 255-271.

Monteith, J. L., Moss, C. J., 1977. Climate and the Efficiency of Crop Production in Britain [and Discussion]. Philosophical Transactions of the Royal Society of London. Series B, Biological Sciences 281 (980), 277-294.

Reyenga, P.J., Howden, S. M., Meinke, H., McKeon, G.M., 1999. Modelling global change impacts on wheat cropping in south-east Queensland, Australia. Environmental Modelling & Software 14, 297-306.

Richards, Lorenzo Adolph, 1931. Capillary conduction of liquids through porous mediums. Journal of Applied Physics 1 (5), 318-333.

Snow, V. O., Huth, N. I., 2004. The APSIM MICROMET Module. HortResearch Internal Report, HortResearch, Auckland.