

DYMEX

Model Builder

VERSION 3



User's Guide

G.F. Maywald, D.J. Kriticos, R.W. Sutherst and W. Bottomley



Hearne Scientific Software End-User Licence Agreement

This is a legal agreement between you (either an individual or an entity) and Hearne Scientific Software Pty Ltd. By opening the sealed Hearne Scientific Software package (herein SOFTWARE) and/or by using the SOFTWARE, you agree to be bound by the terms of this agreement. If you do not agree to the terms of this agreement, promptly return the SOFTWARE and accompanying items (including printed materials or other containers) to the place you obtained them for a full refund within thirty (30) days of the purchase date.

1. **Grant of Licence:** Hearne Scientific Software grants you a non-exclusive licence to use the SOFTWARE in accordance with the following terms.
 - a. **Single-User Licence:** If you purchased a single-user licence, Hearne Scientific Software allows one (1) designated individual, and only that individual, the right to install the software on one (1) home, one (1) work, and one (1) portable computer. The designated individual agrees that no more than one (1) installation will be in use at any given time.
 - b. **Group Licence-Limited Installation:** If you purchased a multi-user licence, you may install and use the SOFTWARE on individual computer workstations within your organisation up to the number of users specified in the purchase order. This type of licence is not portable outside your organisation (i.e. The SOFTWARE can not be installed on a computer at home or on a portable computer unless your organisation purchases a maintenance plan at the time of purchasing the group licence).
2. **Backup Copies:** Hearne Scientific Software grants you the right to make one (1) archival copy of the SOFTWARE for the sole purpose of backing up the SOFTWARE and protecting your investment from loss.
3. **Transfer:** Hearne Scientific Software further grants you the right to transfer this licence and the SOFTWARE to another party provided the following transfer conditions are met.
 - a. The other party accepts all terms of this agreement.
 - b. All copies of the SOFTWARE are transferred and you discontinue use of the SOFTWARE after transferring.
 - c. Hearne Scientific Software is promptly notified of the name and address of the other party and the serial number of the SOFTWARE.
 - d. Hearne Scientific Software is not required to supply new media.
4. **Term and Termination:** Failure to comply with any of these terms will terminate this agreement and your right to use the SOFTWARE. You may also choose to terminate the agreement at any time. Upon termination of this agreement, you must immediately destroy the SOFTWARE and all copies of it.
5. **Copyright:** The SOFTWARE (including any images, applets, photographs, animations, video, audio, music and text incorporated into the SOFTWARE) is owned by CSIRO and Hearne Scientific Software and is protected by Australian copyright laws and international treaty provisions. You agree not to modify, adapt, translate, reverse engineer, decompile, or disassemble the Software. You must treat the SOFTWARE like any other copyrighted material (e.g. a book or musical recording) except that you may either:
 - a. Make one copy of the SOFTWARE solely for backup or archival purposes; or
 - b. Transfer the SOFTWARE to a single hard disk provided you keep the original solely for backup or archival purposes.
 - c. You may only copy the printed materials accompanying the SOFTWARE for your own internal use.
6. **Limited Warranty:** Hearne Scientific Software warrants that the SOFTWARE will perform substantially in accordance with the accompanying printed materials for a period of thirty (30) days from the date of purchase. Any implied warranties on the SOFTWARE are limited to thirty (30) days. Some states/provinces do not allow limitations on duration of an implied warranty, so the above limitation may not apply to you.
7. **Customer Remedies:** Hearne Scientific Software's entire liability and your exclusive remedy shall be, at Hearne Scientific Software's option:
 - a. Return of the price paid; or
 - b. Repair or replacement of the SOFTWARE that does not meet Hearne Scientific Software's limited warranty and that is returned to Hearne Scientific Software with a copy of your receipt. This limited warranty is void if failure of the SOFTWARE or hardware has resulted from accident, abuse, or improper application. Any replacement of SOFTWARE will be warranted for a further thirty (30) days.
8. **No Other Warranties:** To the maximum extent permitted by applicable law, Hearne Scientific Software disclaims all other warranties, expressed or implied, including but not limited to implied warranties of merchantability and fitness for a particular purpose, with respect to the SOFTWARE and the accompanying written materials. You may have others, which vary from state to province to province.
9. **NO LIABILITY FOR CONSEQUENTIAL DAMAGES:** To the maximum extent permitted by applicable law, in no event shall Hearne Scientific Software or its suppliers be liable for any special, incidental, indirect, or consequential damages whatsoever (including, without limitation, damages for loss of business profits, business interruption, loss of business information, or any other pecuniary loss) arising out of the use of or inability to use the SOFTWARE product, even if Hearne Scientific Software has been advised of the possibility of such damages.
10. **Governing Law.** This Agreement will be governed by the laws in force in the State of Victoria, Australia.

Should you have any questions concerning this agreement please contact Hearne Scientific Software.

Hearne Scientific Software Pty Ltd
Level 6, 552 Lonsdale St
Melbourne 3000, Australia
Ph +61 3 9602 5088
Fx +61 3 9602 5050
www.hearne.com.au

Table of Contents

1.	INTRODUCTION.....	1
1.1	WHAT IS DYMEX ?	2
1.2	WHAT IS THE <i>BUILDER</i> ?	3
1.3	MINIMUM REQUIREMENTS TO RUN DYMEX	3
1.4	INSTALLING AND RUNNING DYMEX	3
1.5	CHANGES FROM VERSION 1	4
1.6	CHANGES FROM VERSION 2	5
2.	AN OVERVIEW OF DYMEX MODELS	6
2.1	VARIABLES	6
2.2	MODULES	7
2.3	FUNCTIONS, PROCESSES AND PARAMETERS.....	11
3.	BUILDER FUNDAMENTALS.....	12
3.1	APPEARANCE	12
3.2	MODULE SYMBOLS	13
3.3	MENU COMMANDS	14
3.4	THE VARIABLES WINDOW	15
4.	BUILDING OR MODIFYING A MODEL	17
4.1	GENERAL CONSIDERATIONS	18
4.2	TIMESTEP.....	19
4.3	SUB-POPULATION STRUCTURE.....	20
4.4	MODULE ORDER	21
4.5	NAMING VARIABLES AND MODULES	23
4.6	MODULE INPUTS	23
4.7	MODULE OUTPUTS	25
4.8	MODULE FACTORS.....	26
4.9	MODULE SETTINGS.....	27
4.10	MODULE DESCRIPTION	27
4.11	PARAMETER SET PROPERTIES	27
5.	THE TIMER MODULE.....	29
6.	THE LIFECYCLE MODULE.....	31
6.1	THE LIFESTAGE AND ITS COMPONENTS	34

6.2 LIFECYCLE AND LIFESTAGE TYPES	35
6.2.1 <i>Branching Lifestages</i>	36
6.2.2 <i>Endostages</i>	37
6.3 THE LIFESTAGE WINDOW	39
6.3.1 <i>Lifestage Window Operations</i>	41
6.4 WHAT IS A COHORT IN DYMEX?	43
6.5 COHORTS AND SUB-POPULATIONS	46
6.6 WHAT IS AN ENVIRONMENT IN DYMEX?	46
6.7 WHAT IS A LIFESTAGE PROCESS?	49
6.7.1 <i>Process Rates and the Model Timestep</i>	50
6.8 MODIFYING LIFESTAGE PROCESSES.....	51
6.9 PROCESS COMPONENT DIALOG BOXES.....	52
6.10 FUNCTION DIALOG BOXES.....	53
6.10.1 <i>Parameter Properties Dialog Boxes</i>	54
6.10.2 <i>Advanced Function Properties (Function modifiers) Dialog Box</i>	56
6.11 THE COMBINATION RULE	57
6.12 THE DEVELOPMENT PROCESS	59
6.12.1 <i>Development Update Method</i>	61
6.12.2 <i>Insect Development</i>	62
6.12.3 <i>Plant Growth and Development</i>	63
6.13 THE REPRODUCTION PROCESS	64
6.13.1 <i>Timing of Reproduction</i>	67
6.13.2 <i>Genetic Mixing</i>	68
6.13.3 <i>Insect Reproduction</i>	68
6.13.4 <i>Plant Reproduction</i>	69
6.13.5 <i>Vertebrate Reproduction</i>	70
6.14 THE MORTALITY PROCESS.....	71
6.14.1 <i>Mortality Update Method</i>	75
6.14.2 <i>Mortality Timing</i>	75
6.14.3 <i>Insect Mortality</i>	76
6.14.4 <i>Plant mortality</i>	76
6.14.5 <i>Vertebrate Mortality</i>	78
6.15 THE TRANSFER PROCESS	78
6.15.1 <i>Transfer Update Method</i>	79
6.15.2 <i>Transfer Timing</i>	79

6.15.3 Branching Transfer	80
6.15.4 Cohort and Cohort Variable Transfer	81
6.15.5 Insect Stage Transfer.....	82
6.15.6 Plant Stage Transfer.....	82
6.16 THE IMMIGRATION PROCESS.....	83
6.17 DISPERSAL AND RELATED PROCESSES	83
6.17.1 The Dispersal-timing Process	85
6.17.2 The Dispersal/Mixing Process	85
6.17.3 Dispersal and Cohort Grouping	88
6.18 USER-DEFINED COHORT VARIABLE PROCESSES	90
6.19 THE RESOURCE VARIABLE AND DENSITY CALCULATIONS	96
6.20 LIFESTAGE OUTPUT VARIABLES	97
6.21 LIFECYCLE INPUT	99
6.22 LIFECYCLE FACTORS	100
6.23 OTHER USES OF THE LIFECYCLE MODULE.....	101
7. MODEL INPUT MODULES.....	102
7.1 THE QUERYUSER MODULE.....	102
7.2 THE QUERYUSER/DISCRETE MODULE.....	103
7.3 THE QUERYFILE MODULE	104
7.4 THE DATAFILE MODULE	105
7.5 THE METEOROLOGICAL DATA FILE READER MODULE (METBASE).....	107
7.6 THE CLIMATE DATABASE (METMANAGER) MODULE	109
8. DATA MANIPULATION MODULES.....	111
8.1 THE EXPRESSION MODULE	111
8.2 THE EQUATION MODULE	113
8.3 THE COUNTER MODULE	115
8.4 THE FUNCTION MODULE	116
8.5 THE RUNNING MEAN MODULE.....	118
8.6 THE DIFFERENCE MODULE	119
8.7 THE STORAGE CONTAINER MODULE	119
8.8 THE SWITCH MODULE	121
9. SPECIALISED MODULES.....	123
9.1 THE CIRCADIAN MODULE.....	123
9.2 THE EVENT MODULE	127

9.3 THE DAYLENGTH MODULE.....	133
9.4 THE EVAPORATION MODULE.....	134
9.5 THE WEATHER MODULE.....	136
9.6 THE SOIL MOISTURE (1-LAYER) MODULE	139
9.7 THE DEGREE-DAY MODULE.....	141
9.8 THE CLIMATE CHANGE SCENARIO MODULE	142
9.9 THE DEMESPLITTER AND DEMESTATISTICS MODULES	146
10. FUNCTION TEMPLATES.....	148
10.1 ADDING NEW TEMPLATES.....	148
10.2 FUNCTION TEMPLATE SYNTAX	150
11. SUMMARY VARIABLES.....	154
12. WORKING WITH MODEL DESCRIPTION FILES.....	156
12.1 SAVING MODELS USING AUXILIARY FILES	159
12.2 DIRECTORY LAYOUT FOR MODEL FILES	159
13. APPENDIX I. TRANSFER FUNCTIONS AND TRANSFER RATES.....	160
♦ INDEX.....	163

1. Introduction

The dynamics of animal and plant populations are influenced by many factors, and understanding the response of a population to a multitude of external factors can be very difficult. Simulation models are a powerful means of representing such systems and allowing users to interact with them. These models help to summarize our understanding of a species' population dynamics, identify gaps in knowledge and enable rapid evaluation of management options. Building population models, however, can be expensive in time, and may require specialist programming skills. The DYMEX package is designed to overcome the bottleneck caused by inadequate computer programming resources and modelling expertise. DYMEX enables the user to build a class of ecological models referred to as mechanistic or process-based models, without the need to know a computer programming language. It is a modular modelling software package that consists of two parts: a **Builder** and a **Simulator**. The **Builder** is used to create and modify the model, while the **Simulator** is used to run a completed model, and display the results of simulations.

This User Guide is concerned with the **Builder**, while the **Simulator** is dealt with in a separate User's Guide. The **Builder** guide covers the following topics:

- Building a new model (page 15)
- Creating a lifecycle model (page 31)
- Reading data into your model (page 102)
- Manipulating data (page 109)
- Using specialised data manipulation (page 123)

Other documentation on the DYMEX CD Rom consists of a tutorial based around an insect lifecycle, and two tutorials based on plant lifecycles (an *Annual* and a *Perennial* plant). They allow you to learn the basics of model construction in your field of interest by constructing a new model step-by-step. Several example models are provided to illustrate different aspects and usages of DYMEX features. A careful examination of these may be useful in understanding the paradigm underlying DYMEX. A comprehensive on-line Help system is also provided. It allows you to look up the meanings of terms used in DYMEX, the functions of the various DYMEX modules and available options and limitations. All the documentation is included on the CD-ROM and the manuals and tutorials can be read by installing *Acrobat Reader*, which is also included on the CD-ROM.

Some conventions used throughout this guide are as follows:



The exclamation symbols throughout the manual represent important notes.



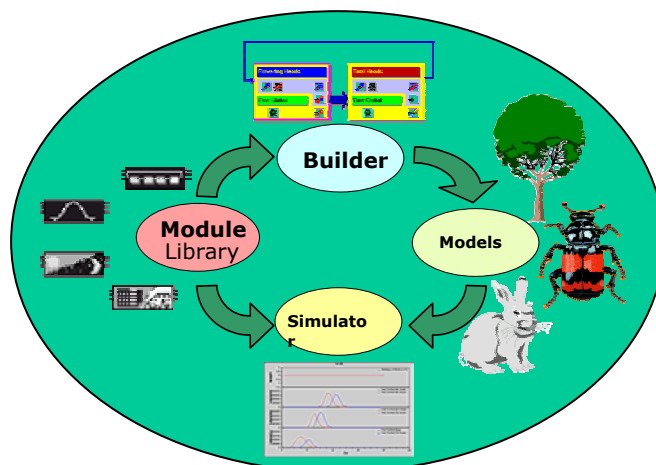
The light bulb represents examples and ideas that can be used when constructing a model.

1.1 What is DYMEX ?

DYMEX is a computer software package that enables you to interactively build and then run models of fluctuating populations of organisms in changing environments. Ecologists can create a wide range of process-based population models without the need to know a programming language. Models are structured around lifecycles, which in turn consist of the stages that individuals pass through during their life. A DYMEX lifecycle describes cohorts of individuals and the processes that affect the size, age and number of individuals in the cohort (see *What is a Cohort in DYMEX?*, page 43).

Models created within DYMEX consist of a series of modules, with each module responsible for a particular task. Modules use information from other modules as input, and supply information to other modules. DYMEX comes with a library of modules that can be incorporated into any model constructed with the **Builder** (Fig. 1-1). Each module performs a specific function (for example, **MetBase** is used to read a standard set of meteorological variables from a file). Models created in the **Builder** can be opened in the **Simulator** within which simulations can be run. The results of these simulations can be displayed in tables, graphs and maps as well as exported to other programs.

Fig. 1-1 A diagram of the function of the **Builder** and **Simulator**.



Models will normally be developed around one or more **Lifecycle** modules. Other modules provide data to the lifecycle modules, or manipulate lifecycle output in some way. Many modules have multiple uses (e.g. **Function** module) and may be used in several places in a model, while others are more specialised (e.g. the **Soil Moisture** module). Most modules receive input from another module or from an outside source. For example, the **MetBase** (Meteorological Database) module reads meteorological data from text files and provides the data as variables that can be accessed by other modules.

1.2 What is the *Builder*?

The DYMEX ***Builder*** is a program that enables the creation of a model from the components (modules, processes, functions etc.) provided in the components library. No computer programming is required in this process – modules are chosen, configured and interconnected using a graphical representation of the modules and lifecycles, and a series of dialogs. The model can be built in several stages, with work saved at any stage to a file (the *Model Description File*, with name “*filename.gmd*”). The ***Builder*** cannot run the completed model - the ***Simulator*** provided with DYMEX is used for that. Typically, a lot of switching between the ***Builder*** and the ***Simulator*** occurs in the process of creating a model. Usually the best approach to building a complex model is to build a simple model first, verify that it runs correctly within the ***Simulator***, and then iteratively extend the model further.

1.3 Minimum Requirements to Run DYMEX

The minimum requirements to run DYMEX are:

- ◆ Pentium 400 MHz or better.
- ◆ Windows NT 4, Windows 2000 or XP
- ◆ 256 megabytes of RAM.
- ◆ 200 megabytes of disk space.

1.4 Installing and Running DYMEX

➤ To install DYMEX on your machine

- 1.** Place the CD into the drive. Installation should start automatically. However, if it does not, proceed as follows.
- 2.** Select **Run...** from the **Start** menu.
- 3.** Within the **Run** dialog box type **d:\setup.exe**. If the CD-ROM drive is not the **d:** drive, then substitute the appropriate drive letter for **d**.

➤ **To run DYMEX**

- 1.** Go to the **Start** menu and find the **DYMEX** program group within the **Programs** group.
- 2.** Choose either the **Builder** or the **Simulator** to run the corresponding DYMEX component.

1.5 Changes from Version 1

The Builder has been considerably enhanced from Version 1. For users familiar with Version 1, the following list details the major differences. *Those changes that may cause models to behave differently when moved to Version 2 are shown in italics.*

- New modules are available, as follows: Adjustable Circadian, Climate Change Scenario, Daydegree, Difference, Equation, Counter, MetManager, Discrete QueryUser, Accumulator (Running Mean), Storage, Switch and Weather.
- Modules and their output variables can have descriptions associated with them.
- The Event module may now have a user-defined delay between trigger and action, a programmable off condition, as well as multiple, independent action factors.
- The Lifecycle module has been considerably enhanced, allowing branching of lifecycles, nested stages (Endostages), an immigration process, “exit” processes and many more lifestage outputs. *By default, Chronological Age is now in units of days (not timestep, as in Version 1). If you built a model with a weekly timestep using version 1 of the Builder and it includes chronological age processes, then in order to run it in version 2 of the builder you will either need to change the manner in which chronological age is updated in the lifecycle properties dialogue box to “timesteps”, or modify all the chronological age processes accordingly. Lifestage densities are calculated differently (see Section 6.19).*
- Summary Variables are now set in the Builder and can be manipulated using modules in a similar way to normal variables.
- Models may be split into several files, with a main Model Description File and a number of auxiliary files. This allows complex models such as a multi-species model to be firstly constructed as separate models, and then combined later (see Section 12).
- Parameters for a module can be placed into a separate, named Parameter File.

- All the variables used in the model can be displayed together in a Variables window along with the source module of the variables and the modules where they are used.
- Any parameter in any module may be replaced by either a function or a process. Processes can now have associated descriptions.

1.6 Changes from Version 2

The following list includes the major differences between Version 2 and Version 3. In addition, a large number of minor improvements and bug fixes have been made to the **Builder** program.

- Populations can be divided into separate sub-populations (demes) (see *Sub-population Structure*, page 20) within the model to represent, for example, genetic types or spatial units. When this is done, variables and parameters that take part in the sub-population structure have components that correspond to each sub-population.
- Operations on lifestage processes have been greatly simplified via a new **Lifestage Window** (see *The Lifestage Window*, page 39). This gives a much better overview of all the processes in the stage, allowing for a better understanding of how the model works. It also reduces the number of dialog operations that were necessary in earlier versions.
- The **Lifecycle Window** has been changed to the same format as that in the Simulator. The window can now be sized, zoomed and printed.
- The **Variables Window** has been much enhanced. Variables can be sorted and the window gives access to dialogs that allow the variables and associated modules to be edited.
- The **Lifecycle** module now can contain factors that belong to the lifecycle as a whole (i.e., they are not part of lifestage processes). This can avoid redefining the same parameter multiple times.


2. An Overview of DYMEX Models

This section describes the fundamentals of a DYMEX model. It is important that the user understands these concepts to effectively build a model using DYMEX.

Building a model in DYMEX involves choosing the required modules from the component library and interconnecting them in a way that describes the essential features of the system being modelled. Although DYMEX makes the mechanics of this process relatively easy, some planning will eliminate a considerable amount of backtracking later. It is always important to keep the aim of the modelling exercise in mind and to construct a model that addresses that aim in the simplest way. Many of the more general modelling issues are discussed elsewhere. See for example M. Gillman and R. Hails, (1997) *An introduction to ecological modelling: Putting Practice into Theory*. Blackwell Science, Oxford; or J.L. Goodenough and J.M. McKinion (eds.), (1992) *Basics of Insect Modelling* ASAE monograph number 10.

2.1 Variables

State Variables


As with all models, a DYMEX model has a number of state variables that in their entirety describe the system at any particular time. Examples of such variables in a particular model may be *Maximum Temperature*, *Rainfall*, *Daylength*, *Total No of Flies*, and *Egg Development Time*. Changes in the values of variables take place between one timestep and the next. The values of all DYMEX variables are updated by their associated modules. A history of values of the variables throughout a simulation run constitutes the results of that simulation, and they are retained for tabulation or charting. All variables have names, and (with the exception of a few predefined names) the modeller is responsible for naming each variable. Variables may also have a short name (mnemonic) and description, and are often represented in **Builder** windows by the symbol .

One type of state variable that is commonly encountered in a DYMEX model is used to summarise or report on some internal model processes. For example, we might have a variable that outputs the average Physiological Age of individuals in a lifestage. This type of variable may not be defined at all times during a simulation run (for example at times during the simulation when no individuals of that lifestage are present), and should only be used as input to modules or functions with caution.

Delay Variables



Delay Variables are used in DYMEX to describe intervals of time, for example the time required for an egg laid on a particular day to hatch. Such variables do not have their values assigned to them until the time period that they are monitoring has terminated (i.e., the eggs have hatched in the example). These variables are designed for use as output to graphs and tables only, and must never be used as inputs to modules or functions. They include such commonly used lifestage outputs as Development Time and Cohort Duration.

If a model uses sub-populations (demes) , some variables will contain multiple values for each timestep. These variables are termed **Demed Variables**, and are indicated in the *Builder* with the symbol . Module inputs that require a single-population input variable may not be linked to a demed variable. The converse is not true. If a single-valued variable is linked to a module input that requires a demed variable, the separate input variable components will all be set to the value of the input variable (see Fig. 2-2). Special module (**DemeSplitter** and **DemeStatistics**) can be used to extract the component values from demed variables as standard variables.

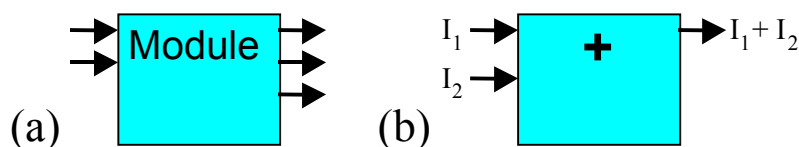
There are two pre-defined, special variable names used in DYMEX. The first, “*(none)*”, is really no variable at all – it is the name used in places where no other variable name has been specified. For example, all module inputs are initially set to this name. The other name, “*Simulation Id*”, refers to a variable that, when the model is run, will take a value equal to the sequence number of the current simulation. Hence, in single simulation runs (see *Simulator User’s Guide*, Section 18), its value will always be 1. Except for specialized purposes, there would rarely be a need to use this variable in a model.

A further type of variable (*Summary Variable*) can be used to summarise the values of other variables over a period of time. For example, a model may have a standard lifecycle output variable named “Total Plants” that supplies the total number of plants in a simulation at any time. For running and comparing multiple simulations, we may want a single number that summarises the effect of some treatment regime on that plant population. The most convenient way to do this would be to create a Summary Variable based on “Total Plants” that gives the average value for “Total Plants” over (say) the last year of the simulation. As many summary variables as required can be created, and Summary Variables can be manipulated with *Summary Modules* (see next Section and Fig. 2-3). The creation of Summary Variables is dealt with in Section 11.

2.2 Modules

Understanding the function of modules and how they interact in a DYMEX model is central to the model-building process. In its simplest form, a module is a black box that may have one or more input variables and has one or more output variables, as shown in Fig. 2-1a. The module is responsible for calculating the value of its output variables at each time step. Several different types of modules are available for use in a DYMEX model, each performing a different function. For example, an **Expression** module could have two inputs and one output, and calculate the sum of the two input variables, assigning the result to the output variable (Fig. 2-1b).

Fig. 2-1 (a) Schematic representation of a DYMEX module; (b) a DYMEX “addition” module



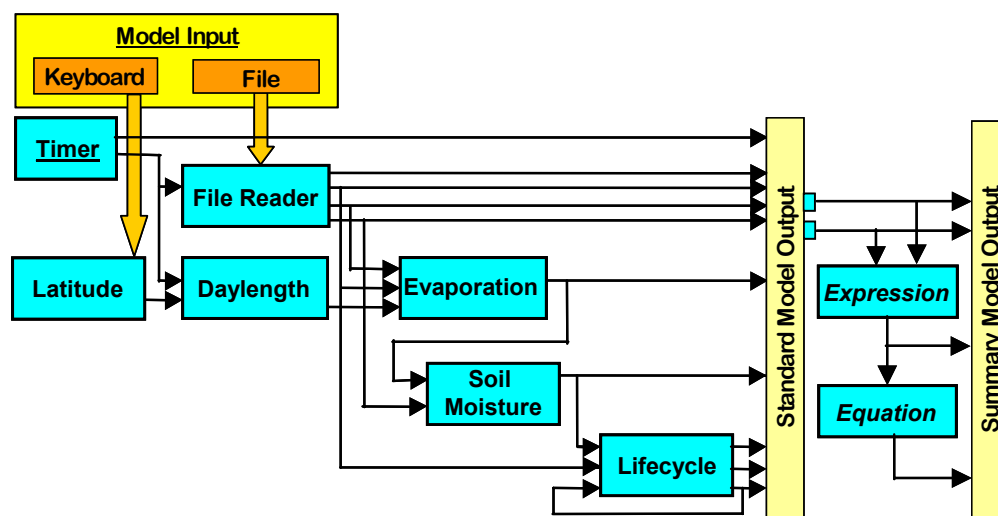
Modules are connected by linking each of their input variables to an appropriate output variable of another (or the same) module. During a simulation, the value of that output variable is then used as the input value. Fig. 2-3 shows an example of a typical model and its linkages between modules. For the moment, disregard the part of the diagram to the right of the long, thin rectangle labelled “*Standard Model Output*”. The model makes use of the **Soil Moisture** module, which has two input variables, Rainfall and Evaporation. The Rainfall input is linked to the rainfall output of a File Reader (**DataFile**) module, while the Evaporation input comes directly from an **Evaporation** module. Input variables must be linked to output variables from other modules that supply the appropriate information, and the correct matching of units must be taken care of by the modeller. For example, not only must the rainfall input of the Soil Moisture module be linked to a rainfall output from another module, these outputs and inputs must be in the same units (e.g., mm or inches). The current version of the *Builder* does only the most basic checks that links are compatible. In models that use sub-population structure, demed variables may not be connected to inputs that require standard variables (Fig. 2-2). Note that in this case, the *Builder* does not allow such a connection. Sometimes an input variable is optional, in which case it does not need to be linked to another variable. Note the feedback loop in the Lifecycle module, where an input variable is linked to an output of the same module.


Fig. 2-2 Module inputs, showing how variables with and without subpopulation structure can be connected as module inputs.



It should be evident by now that when we refer to a module input, we are not referring to information read from a file or supplied via a dialog, but the points of connection to a “previous” module’s output variable. Data from a source external to the model (model input) such as data files or from the keyboard (through dialogs) is accessed via specialized modules (as shown in the Model Input block in Fig. 2-3). This information then becomes available via that module’s output variables (as shown by the arrows from the module to the *Standard Model Output* block).











Fig. 2-3 A DYMEX model showing the flow of information (variables) between modules.



The model designer can configure most modules to adapt them to their required task. For example, Expression modules referred to earlier can be set up to add, multiply or average their input variables. The Lifecycle module in particular is extremely flexible, and can be set up in many ways to suit the particular organism being modelled. Several of the modules are sub-models, whose function is to do a specific job. For example, the 1-layer Soil Moisture module simulates the water balance in the top layer of soil. It is anticipated that one way that DYMEX will be enhanced in the future will be by the addition of new modules. The currently available modules and their application are summarised in Table 2-1. Note that some modules are available in several “varieties” (for example, the “Event” module can have one or more event output variables, with the additional option of a delay between the triggering condition and the action). These have not been shown separately in the Table, but these differences will be discussed in the appropriate section of the Guide. Modules that can be used to manipulate demed variables (i.e., they can take part in the model’s sub-population structure) are indicated with the symbol .

As alluded to above, all module output variables are available as part of the model output for tabulation, graphing, etc. Each of these outputs is an array of values that stores the variable’s values for each time step over the period of a simulation. Collectively, these outputs constitute the *Standard Model Output* (Fig. 2-3). Any of these output variables may be summarised, as described in Section 2.1. These summaries are available as *Summary Variables*, and the set of Summary Variables constitutes the *Summary Model Output* (Fig. 2-3). Modules may be added to a DYMEX model to manipulate these Summary Variables, creating more Summary Variables in the process (as shown by the two modules and their outputs at the right of Fig. 2-3). These **Summary Modules** are updated only once during a simulation run, directly after the simulation has finished running. Only some module types are available for use as Summary Modules.

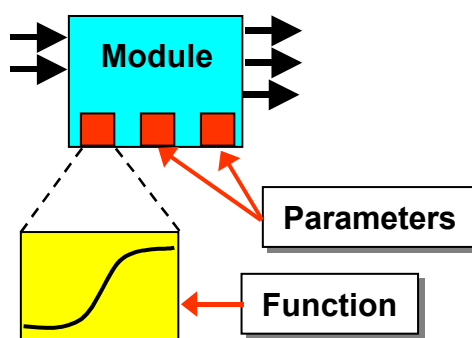
Table 2-1 Modules available in the current version of DYMEX, with a summary of their function and a reference to the sections in the Builder Guide where they are described in detail.

<i>Module Type</i>	<i>Description</i>	<i>Section</i>	
Circadian	Fits a daily cycle to minimum and maximum values	9.1	
Counter	Keeps a counter incremented conditionally	8.3	
DataFile	Reads model variables from a file	7.4	
DegreeDay	Calculates degree-days above a threshold	9.7	
Daylength	Calculates daylength from latitude and day of year	9.3	
Difference	Calculates change in a variable between timesteps	8.6	
Equation	Combines input variables using an equation	8.2	
Evaporation	Calculates Class A Pan Evaporation	9.4	
Event	Simulates date or threshold controlled management	9.2	
Expression	Combines input variables using simple maths	8.1	
Function	Transforms an input variable by use of a function	8.4	
Lifecycle	Models population dynamics of a cohort-based lifecycle	6	
MetBase	Reads meteorological data from a file	7.5	
Metman	Reads long-term average meteorological data	7.6	
QueryFile	Inputs model constants from keyboard or file	7.3	
QueryUser	Inputs model constants from keyboard	7.1	
QueryUser (Discrete)	Inputs model constants from a discrete set of values from keyboard	7.2	
RunMean	Calculates a running mean (or total) of a variables values	8.5	
Soil Moisture	A simple soil moisture balance model	9.6	
Storage	Simulates a storage with inflows and outflows	8.7	
Switch	Allows selection of alternative variables	8.8	
Timer	Provides model timing	5	
Weather	Calculates daylength, evaporation and daily temperature cycle, etc.	9.5	
DemeSplitter	Splits a <i>demed</i> input variable into its components	9.9	
DemeStatistics	Calculates totals and proportions from a <i>demed</i> input variable	9.9	

2.3 Functions, Processes and Parameters

Some modules can be adjusted by setting the values of one or more *parameters*. Parameters are somewhat similar to variables, except that their value is fixed during any simulation run. For example, the **Soil Moisture** module has 3 parameters, the Soil Moisture Capacity, the Evapotranspiration Coefficient and the Drainage Rate (Fig. 2-4 shows this schematically, with the parameters indicated as dark grey squares). Thus parameters can be thought of as devices for adjusting a model to a particular place or situation.

Fig. 2-4 A DYMEX module with 3 parameters, showing how a parameter can be replaced by a functional relationship



Any DYMEX module parameter can be replaced by either a function or process. In the Soil Moisture example, we may have a relationship between the value of the Evapotranspiration Coefficient and some other variable (perhaps crop biomass). DYMEX allows us to substitute that relationship for the parameter (Fig. 2-4). In that situation, we end up with some additional parameters (the parameters of the functional relationship or process specified), as well as a new *de facto* input variable to the module (crop biomass). The term **Factor** is usually used in DYMEX to refer to a component that could be a parameter, function or process.

A DYMEX **Function** consists of a function shape (called a *Function Template*), a driving variable (the independent or x-variable) and a set of parameters (which can be replaced by a function or process, in turn). DYMEX provides a library of over 20 commonly used function templates. If these templates are insufficient for any situation, the modeller can define a new template and add it to the function template library. The pre-defined function templates are described in the Help system.

A **Process** is a set of factors that are combined using a functional relationship termed the *Combination Rule*. On evaluation, each of the component factors is evaluated first, and the results are then combined using the Combination Rule to give a final value.

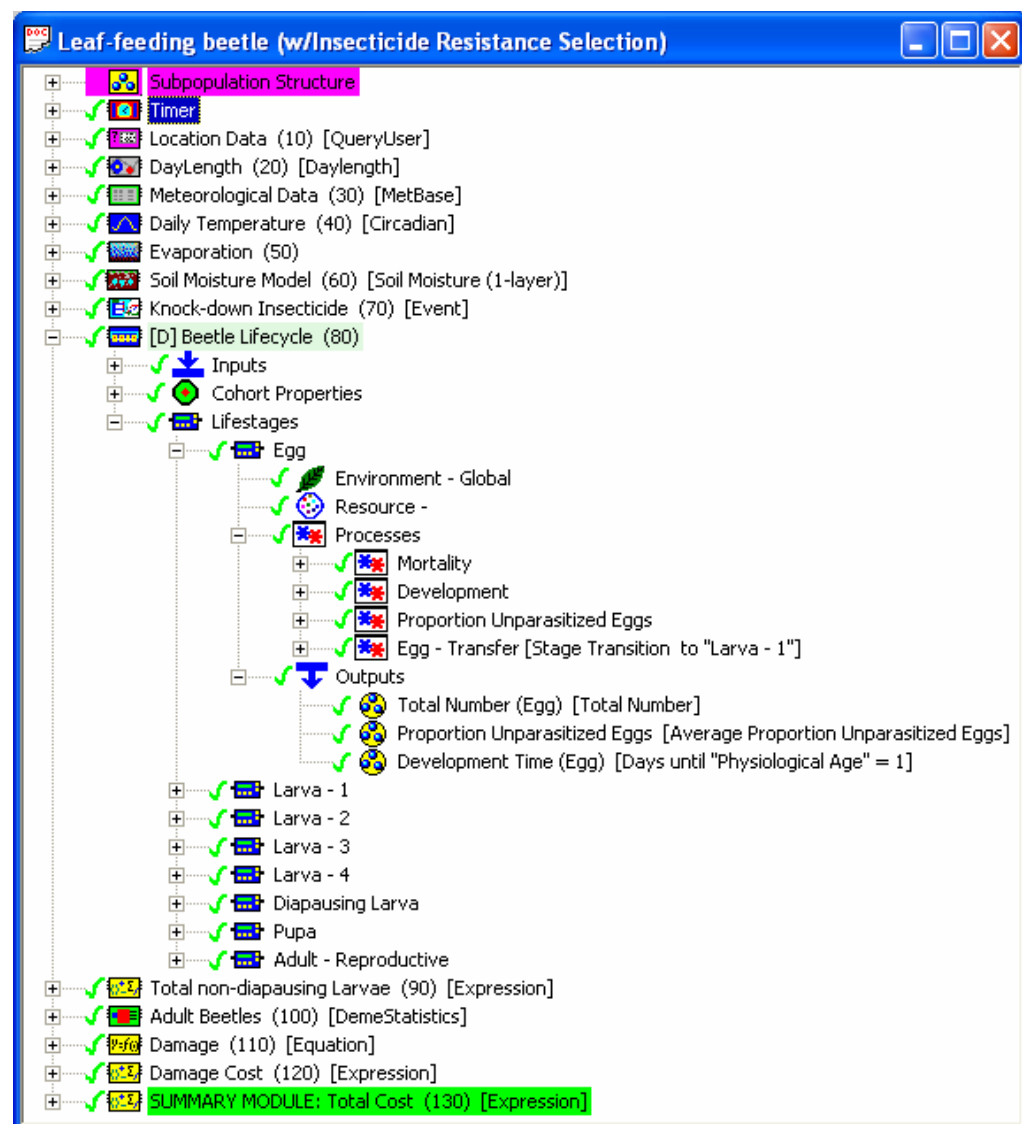
3. Builder Fundamentals

This section describes the appearance of the *Builder* program, and the functions of its various component windows.

3.1 Appearance

Model construction is based around the **Component Window** where modules are added, removed or modified. All the modules used by the model are listed within the Component Window, where they are shown as small rectangular graphics, with a different graphic associated with each module type. This helps the user to visualize the structure of the model, as well as allowing easy access to modules for examination and modification.

Fig. 3-1 The Component window in the Model *Builder*. Note the Lifecycle module, which has been opened to show some module details.



Clicking on the small square (containing a '+') to the left of the module graphic provides more information about that module. The major components of the module are shown as the main branches of a tree diagram originating with the module graphic. Similarly, these components can be opened out further to provide more detailed information.

Many **Builder** operations are available from the menu. Menu commands followed by ellipses (...) will open dialog boxes. Double-clicking on the appropriate module or module component in the Model Components window is an alternative method of access to some of these operations.

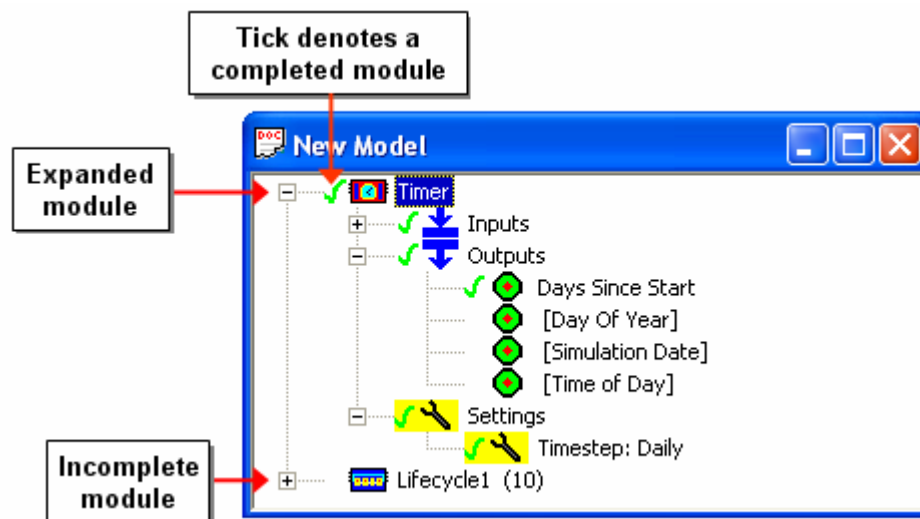
To access a particular module, double click its graphic. To get further information left click on the "+" to the left of the graphic, and then double-click on one of the displayed module components.

3.2 Module Symbols

In the **Component Window**, the display of each module can be expanded so that processes and variables entered into each module may be viewed. All modules can be fully expanded at once by choosing **Expand All Branches**. Selecting the **Collapse All Branches** option from the menu will close up a fully expanded diagram.

When a module representation in the Component Window is expanded, many of the features of the module have symbols next to them (Fig. 3-2). These symbols indicate the type of component.

Fig. 3-2 The Component Window within the Model Builder.

















The components that are represented in the Component Window are tabulated in Table 3-1. At this stage, the use of some of the features may not be clear, but these will be explained in later sections.

The name of a module can be changed by clicking on it once when it is selected in the Component Window, and then typing a new name into the resulting edit window.

Modules can be opened for modification by either double clicking on the module in the **Component Window** or by choosing **Edit Module...** from the **Model** menu. When multiple modules are open, switching between the modules can be done from the **Window** menu where the open modules are listed. Closing a module by left-clicking on the ‘close window’ control button performs the same operation as clicking on the OK button (ie. all settings are saved).

When selected, most modules will open a dialog box (e.g. **Circadian** module). The **Lifecycle** module opens a window depicting the lifestages within the lifecycle. The lifestage currently selected is highlighted with a pink edge. Buttons on the lifestage allow access to the various processes and associated variables.

Table 3-1 Component Window symbols and their definition.

<i>Symbol</i>	<i>Definition</i>	<i>Symbol</i>	<i>Definition</i>
	A module, see <i>Table 2-1</i>		Parameter/constant
	The set of output variables		A resource variable
	The set of input variables		A lifecycle/lifestage
	A variable		An environment
	A demod variable		A process
	A function		An “action” (function or constant)
	Subpopulation structure		Setting/Combination Rule

3.3 Menu Commands

The menu bar changes in appearance with the window that is currently active. Table 3-2 lists the functions of the top level menu items. Note that many actions available from the menu can also be invoked by clicking on the appropriate symbols or buttons in the Component or Lifecycle windows.

Table 3-2 Menu items and their function.

Menu Item	Description of function
File	The <u>File</u> menu enables new models to be created or existing models to be opened, saved and printed (see <i>Building or Modifying a Model</i> , page 17).
Model	The <u>Model</u> menu enables the addition, editing and removal of modules, function templates and environments from a model, the setting of the model description, including its name and author, as well as access to the Simulator to run the completed model.
Tools	The <u>Tools</u> menu enables some operating characteristics of the Builder to be changed. These settings include such features as colour, size, and model opening and printing options.
Window	The <u>Window</u> menu enables the various windows open within DYMEX to be arranged.
Help	The <u>Help</u> menu provides access to the help documentation.
Lifecycle	The <u>Lifecycle</u> menu provides access to various features of the lifecycle, including the name, description, the sort order, and the input variables. The addition and removal of lifestages and the addition of user-defined cohort variables are also performed via this menu item (see <i>The Lifecycle Module</i> , page 31).
Lifestage	The <u>Lifestage</u> menu enables the user to edit features of the currently selected lifestage including its name, the associated processes, the output variables and its associated environment resource variable. (see <i>The Lifestage Window</i> , page 39).

3.4 The Variables window

The **Variables Window** displays the names of all variables that are used in a model. Besides listing the names of the variables, the window can be used to make changes to the properties of any of the variables and to look at and modify the module that update each variable (source module). In addition, the window lists all modules that makes use of the variable.

To open the **Variables Window**, select the **Variables** option from the **Model** menu (note that the **Model** menu will only be present if the **Model Components** window is the active window). A part of a Variable Window is shown in Fig. 3-3.

The Variable Window lists the variables in two sections. The top of the window (mostly light-green in colour) lists the normal variables, while summary variables are shown in a section at the end (coloured pink). When the Variables Window is first opened, the variables are shown in the order that they are declared in the modules (module order). A click on the “Name of Variable” header, will sort them alphabetically and a second click will sort

them in reverse alphabetic order. Note that whichever way they are sorted, the summary variables will remain in one block at the bottom of the window.

Fig. 3-3 The Variables Window within the Model *Builder*, showing both normal, demed and summary variables.

Name of Variable	Mnemonic	Source Module	Used in Module	Used for
Total Diapausing Larva The total number of diapausing larvae in the simulation.		Beetle Lifecycle	-	
Total Larvae		Total non-diapausing Larvae	Beetle Lifecycle	Cohort Process (Larva - 1)
Total Number (Adult)		Beetle Lifecycle	Damage	Module Input
Total Number (Adult) - Composite		Beetle Lifecycle	Adult Beetles	Module Input
Total Number (Egg)		Beetle Lifecycle	-	
Total Number (Larva - 1)		Beetle Lifecycle	Total non-diapausing Larvae	Module Input
Total Number (Larva - 2)		Beetle Lifecycle	Total non-diapausing Larvae	Module Input
Total Number (Larva - 3)		Beetle Lifecycle	Total non-diapausing Larvae	Module Input
Total Number (Larva - 4)		Beetle Lifecycle	Damage	Module Input
Total Number (Pupa)		Beetle Lifecycle	-	
Adult Beetles The average number of adult beetles over the last year of the simulation.	Adults			
Cost of Knockdown Spray (\$) The cost incurred from spraying with the knockdown insecticide in the last year of the simulation (\$).	Knk Spray Cost		Total Cost	Module Input

To the left of the variable name, a symbol indicates whether the variable is *demed* (🌐) or not (🌱). Clicking on the name of a variable opens the **Variable Properties** dialog (Fig. 3-4), from which most properties of the variable (including its name and description) can be changed.

Fig. 3-4 The Variables Properties dialog.

Variable Properties

Name: Total Diapausing Larva

Description: The total number of diapausing larvae in the simulation.

Mnemonic: Lar-D (tot)

Variable type indicator: 🌐

Values:

Minimum: []

Maximum: []

Default: []

☐ Private

Default Precision: 2

OK Cancel Help

Clicking on the name of a module in the “Source Module” or “Used in Module” columns opens the module dialog or window for the corresponding module. More than one module may be shown in the “Used in Module” column if the variable is used in more than one module. The “Used for” column indicates how the variable is used in a module. If it indicates “Module Input”, the variable is directly connected to one of the module inputs. If it specifies “Cohort Process (*stagename*)”, it is used as an independent variable in a lifestage process of the lifestage named “*stagename*”.

4. Building or Modifying a Model

➤ To create a new model

Select **New Model** from the **File** menu.

This will create a new model containing only a **Timer** module and display it in a **Component Window**. A new model is always created with a **Timer** module in it. The Timer module may not be deleted.

➤ To load an existing model

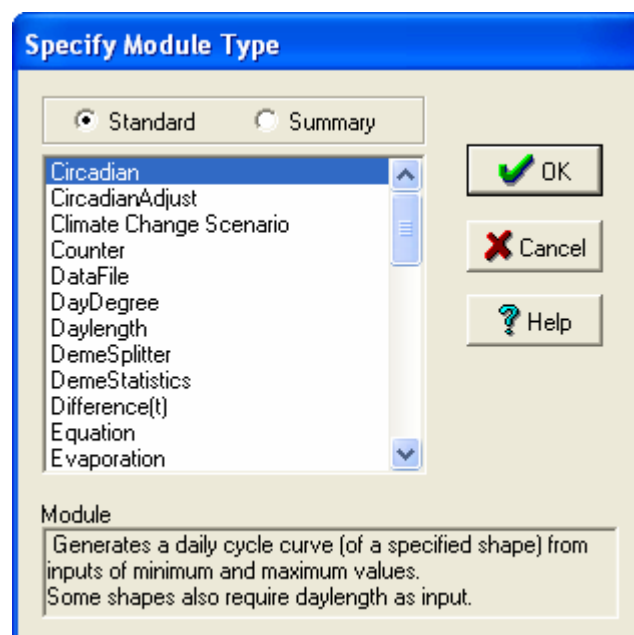
Select **Open Model...** from the **File** menu, and select the required model from the Open File dialog. If the model has been opened previously, it can be re-opened rapidly by clicking on its name at the bottom of the File menu.

The model will be displayed in the **Component Window**, ready for editing.

➤ To add a module to a model

1. From the Component Window select the **Add Module...** option from the **Model** menu.
2. In the module selection dialog (Fig. 4-1), select either **Standard** for a “normal” module, or **Summary** for a *Summary Module*.
3. Select the required module type from the list box, and double-click on it, or click **Ok**, to add a new module of that type.

Fig. 4-1 The Module Type selection dialog, used to add a new module to a model.

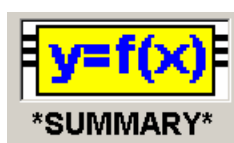


The new module will be added to the end of current list of modules in the model. The placement of the modules in relation to other modules needs to be

carefully considered, and can be changed using the Sort Order facility (Section 4.4). Note that *Summary Modules* are always placed after all standard modules, regardless of the sort order constants that they are given.

All modules except the Lifecycle modules are configured using the **Module Dialog**, examples of which are shown throughout this User's Guide (for example, Fig. 7-1). Note that Summary Modules are indicated by the word "SUMMARY" placed under the module graphic at the top right of the dialog. Lifecycle modules are edited in the Lifecycle window (Section 6).

Fig. 4-2 Part of Module Dialog, showing *Summary Module* indicator.



4.1 General Considerations

What order
do I add the
modules?

There is no required order in which to add modules, but it is important to give careful consideration to the order. If you already have the model planned, then adding the modules that supply data may be your first choice (page 102), or you may prefer to add a lifecycle module first to construct the lifecycle structure (page 31).



Note that while there is no set order on how to add modules when the model is being constructed, simulation proceeds in the order that the modules are listed in the Components Window. Therefore the ordering of modules will need to be carefully considered to obtain the required effect. For example, modules that supply data to another module will usually come before that module (see Module Order, page 21 for more information on module order).

The dynamic aspects of some modules are not fully described in this Guide and further information on their use can be found in the **Model Simulator** User's Guide.

The modules are divided into several groups in the User's Guides: The Timer module (page 29), the Lifecycle module (page 31), the modules used for inputting data (page 102), manipulation of variables (page 111), and other specialised modules (page 123).

Rather than designing and building an elaborate model from the start, it is often better to start with a relatively simple model. This can be tested in the **Simulator**, and experience gained with it can be used to adjust and elaborate it where required.

It is important to be clear about the purpose of the model. One possible and common use is as a simple development predictor. A farmer may want to predict the time when the damaging late larval stage of a particular pest will be

present, using a knowledge of the occurrence of adults from the previous generation. In such a case, no mortality processes need to be modelled at all and in fact it may even be possible to lump several earlier stages together into one stage. At the other end of the scale, if the model is required for optimising the timing and frequency of application of a mix of control methods, a very detailed model may be required, including economic relationships.

Commonly, a model will be one of the tools that an ecologist will use in the study of a particular species or system. The data that can be used to determine the shapes of functions and estimate their parameters will usually be scattered through the literature, or absent. An initial simple model using any available data (with important gaps filled with “guesses”) can be a valuable adjunct to understanding the structure and behaviour of the system under study. It is important that the data sources and quality, together with the assumptions, be rigorously documented (DYMEX provides “Comment” fields for this purpose). The discrepancy between model output and field data can then be used to refine the model further by designing experiments to fill gaps or elucidate puzzling behaviour. As confidence in the model grows, management events (such as pesticide treatment) can be added using Event modules. At any time, the model can serve as a summary of the ecologist’s understanding of the system.



The model parameters are used to “tune” the model, and their values are likely to change often as the model is built and then fitted against available data. Parameter ranges and “default” values are set in the **Builder**, but their actual values are normally stored in a *Parameter File* that is maintained by the **Simulator**. During the model construction phase, it is strongly advisable to disable the *Parameter File* and just use the “default” values set in the **Builder**. This is done by setting the “Use Parameter Default” option in the **Model Options** dialog of the **Simulator** program (where it is accessed from the **Current Model** item in the **Preferences** menu).

☒ Use Parameter Defaults (not parameter file)

4.2 Timestep

A major decision that needs to be made early in the model building cycle is the choice of timestep. This will usually be determined by such factors as the briefest stage in a lifecycle being modelled. For example, whereas a woody weed might be adequately modelled with a 7- or 30-day timestep, an insect whose eggs develop in three days would probably require a one-day timestep. For even shorter timescales, a segmented mode of operation is available. In this mode, the daily timestep is divided into a number of segments that can be used to update variables at more frequent intervals than 1 day. See also *Process Rates and the Model Timestep* (page 50) for a discussion of the effect of timestep on model processes and parameters. See the Timer description on page 29 for instructions on how to change the timestep.

4.3 Sub-population Structure

Normally, DYMEX models a single population (or, if multiple species are included in the model, a single community) of organisms. This is useful for understanding the dynamics of these populations and can give valuable guidance for management and control. In these single point models, dispersal from outside the model domain is dealt with using the immigration process, while emigration can be approximated as a mortality. There are many situations where this approach is unsatisfactory and dispersal needs to be treated in more detail. Version 3 of DYMEX includes a facility to divide the model domain into separate units, each of which can contain sub-populations (also referred to as “demes”) of the organisms being modelled. These sub-populations can be given spatial coordinates and dispersal between them can be modelled explicitly. Parameters can be different for the different sub-populations. For example, sub-populations could represent adjacent fields growing different crops, and different parameters could be used to characterize the suitability of the crops for the pest species being modelled. Another way that sub-populations can be used in DYMEX is to model genetic phenomena such as the development of resistance to a chemical being applied for control. There are currently restrictions to this, in that only a 3-genotype model is possible. The genetic sub-populations can be combined with spatial sub-populations to model, for example, how development of resistance to a genetically modified crop such as BT-cotton might be slowed by planting of normal cotton.



The sub-population structure of the model is modified via the **Subpopulation structure** menu item in the **Model** menu. This opens the **Subpopulation Structure** dialog (Fig. 4-3). The sub-population structure of the model is a fundamental property of the model and should be carefully considered before any changes are made. Once a particular sub-population scheme is adopted, making changes later can involve a considerable amount of work in different parts of the model.

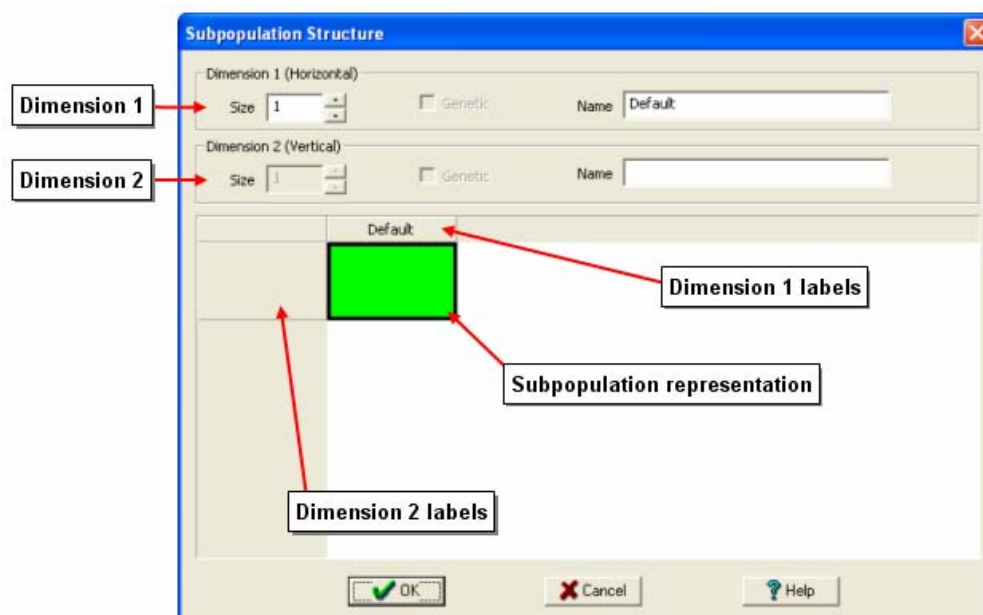
The current sub-population representation is shown as a series of rectangles in the lower part of the dialog. Subpopulations can be laid out in two dimensions (these do not need to represent an actual matrix of cells) as columns and rows. Often, for example, one dimension may represent a number of spatial units, while the other may represent genotypes. Labels for the columns and rows may be specified, and these labels uniquely identify each subpopulation.

➤ **To create three sub-populations that represent genotypes in a population**

1. With the **Model Components** window active, click on the **Model** menu and select **Subpopulation Structure**. This opens the Subpopulation Structure dialog (Fig. 4-3).
2. In the **Size** edit box for Dimension 1, type 3 as the number of sub-populations (or click on the upper small arrow to the left of the edit box until “3” is displayed).

3. The **Genetic** check-box will now be active. Click on it to select genetic sub-populations. The colour of the boxes will change to show that 3 different genotypes are present.
4. In the **Name** box for Dimension 1, type in the desired name for this group of subpopulations (a suitable name in this case may be “*Genotypes*”)
5. Click on the grey header above each subpopulation box and type in a name for the corresponding subpopulation. For example, the three genotypes might be named “*SS*”, “*SR*” and “*RR*”.

Fig. 4-3 The Subpopulation Structure dialog, showing the default, single-population schema.



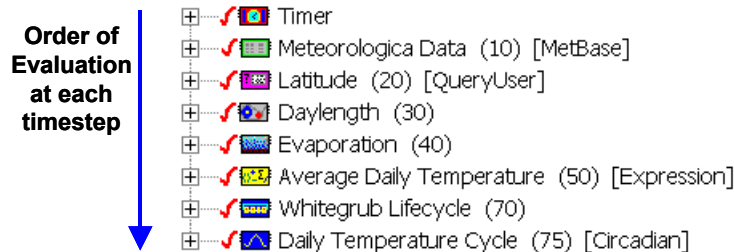
4.4 Module Order



The order of the modules is an important feature of a model. Modules are executed in the order that they appear in the **Component Window** (referred to as the “Sort Order”). Therefore if results produced by one module are used as input to another module; the data producer should generally be placed before the module that uses the data. There will be cases where this is not possible (for example, a density dependent relationship in a lifecycle might use the total number of individuals output from the same lifecycle). In cases such as those, the module using the values will be inputting values from the previous time step. Fig. 4-4 illustrates an inappropriately ordered model. Any modules that use “Daily Temperature Cycle” as input will use its value from the previous time step. Note that a “cycle” consisting of all zero values will be used as input in the first time step.

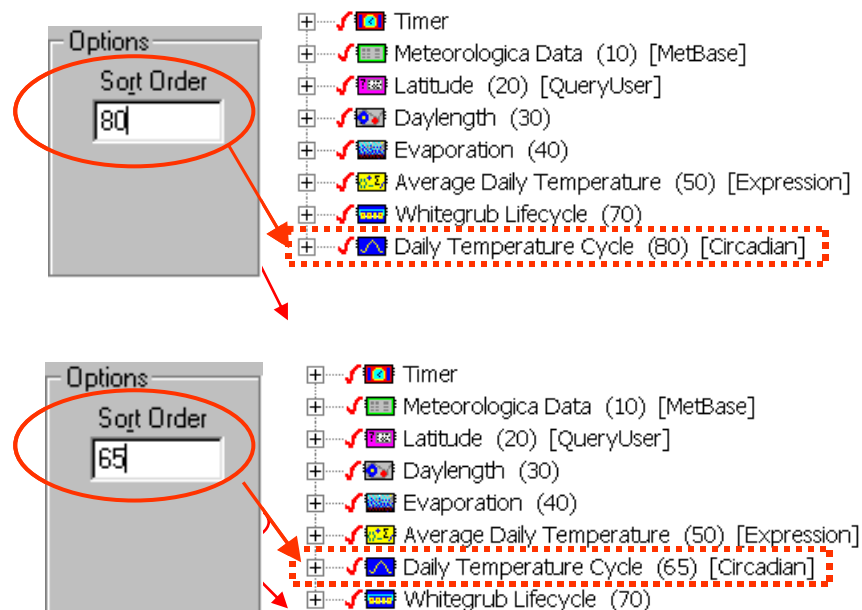
The **Timer** module, which always occurs first, has a sort order of zero. Every new module added to the model will have a sort order 10 greater than the previous module with the highest sort order.

Fig. 4-4 An illustration of the results of an incorrect ordering of modules: in the first timestep the daily temperature cycle used in the lifecycle will be all zeros and in every timestep thereafter will have a daily temperature cycle from the previous timestep.



In the top half of Fig. 4-4 the sort order is incorrect because the daily temperature cycle is used within the lifecycle to determine one or more processes, but is calculated after the lifecycle processes are evaluated. Thus, the previous day's temperatures are used as input to the lifecycle. To correct this situation, the Daily Temperature Cycle module needs to be placed before the Lifecycle module (and after the module that produces its inputs, MetBase). To achieve this, the Daily Temperature Cycle sort order is changed from 80 to 65, as illustrated. Note that any number between 40 and 70 could have been used.

Fig. 4-45 Two different sort orders; The first sort order was incorrect in this case because daily temperature was used in the lifecycle and should come before the lifecycle.



- **To change the sort order of all other modules besides the lifecycle module**

- 1.** Double click on the module to open it.
- 2.** Change the number in the **Sort Order** text box within the **Options** box on the right hand side of the module dialog box to the required value.

The sort order of a Lifecycle module is changed from the **Lifecycle Properties** dialog (see Section 6).

4.5 Naming Variables and Modules

For future reference and good housekeeping, it is important to name the model, provide author and version details, and a description (or overview) of the model. In addition, any assumptions made should be listed, either here, or in relevant comment boxes associated with modules, processes and parameters.

- **To edit the model description**

- 1.** Select **Details...** from the **Model** menu.

The **Model Details** dialog box allows the name of the model, the author, the version number of the model, and comments regarding the model to be set. Comments are not limited to the area shown in the dialog box and a large amount of information can be included in the comment box.

The
importance of
names!

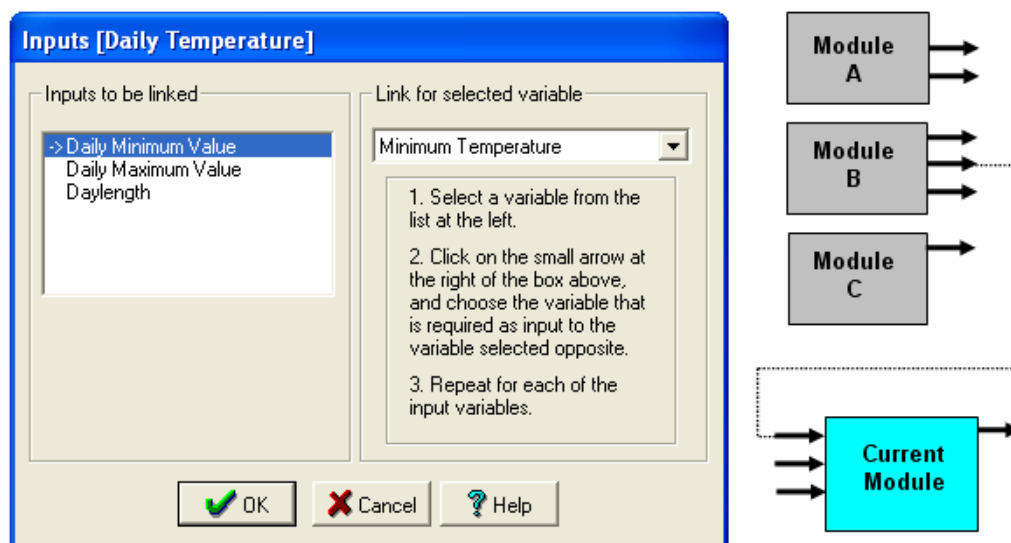


It is important when naming modules, processes, functions and their parameters that certain rules are followed. It is desirable that the names of modules, variables and parameters should be unique and meaningful to avoid later confusion. In most cases, the **Builder** will not allow the use of duplicate names for similar entities – for example, two modules with the same name are not permitted. To avoid problems, it is also wise not to use DYMEX reserved terms such as “Module”, “Function”, “Parameter” or “Variable” as names. The word “run” must not be used as a name. Do not use semicolons (;), quotes (“”) or commas (,) in names.

4.6 Module Inputs

Module inputs are set in the Module Inputs dialog. This dialog is accessed in one of two ways. Either double-click on the required module symbol in the **Model Components** window, and then click on the “**Inputs**” button in the Module window, or open the module using the small “+” symbol to the left of its bitmap, and double-click on the “Inputs” line. A typical Module Inputs dialog is shown in Fig. 4-6.

Fig. 4-6 Module Inputs dialog, showing the linking action performed between two modules schematically on the right



The list box on the left contains all the input variables for that module. Any input that is already linked to a variable is preceded by the symbols “->”, and the currently selected input is highlighted. A small window at the top right indicates the variable that the currently selected input is linked to. To change the linked variable, click on the small button at the top right. This opens up a list of all the variables in the model that are available for linking to the selected input. The required variable is then selected from that list. The diagram at the right of Fig. 4-6 shows the action performed schematically. The module’s input is linked to the output variable of another module, thus linking the two modules (as indicated by the dotted line).

Note that not all model output variables will be listed in the available variables. DYMEX has a simple variable type system, which recognises that some variables are not appropriate for a particular input. This type system is not complete, however, and there will be some variables listed that would not be valid input variables for the selected input. The user must take care to ensure that variables linked to an input are actually appropriate for that input.



In a model that has multiple sub-populations, a module that does not make use of the sub-populations cannot use a demed variable as input. The demed variables will not be listed in the available variables list for these modules. The converse is not true, i.e., modules that use sub-populations are able to use non-demed variables as input.

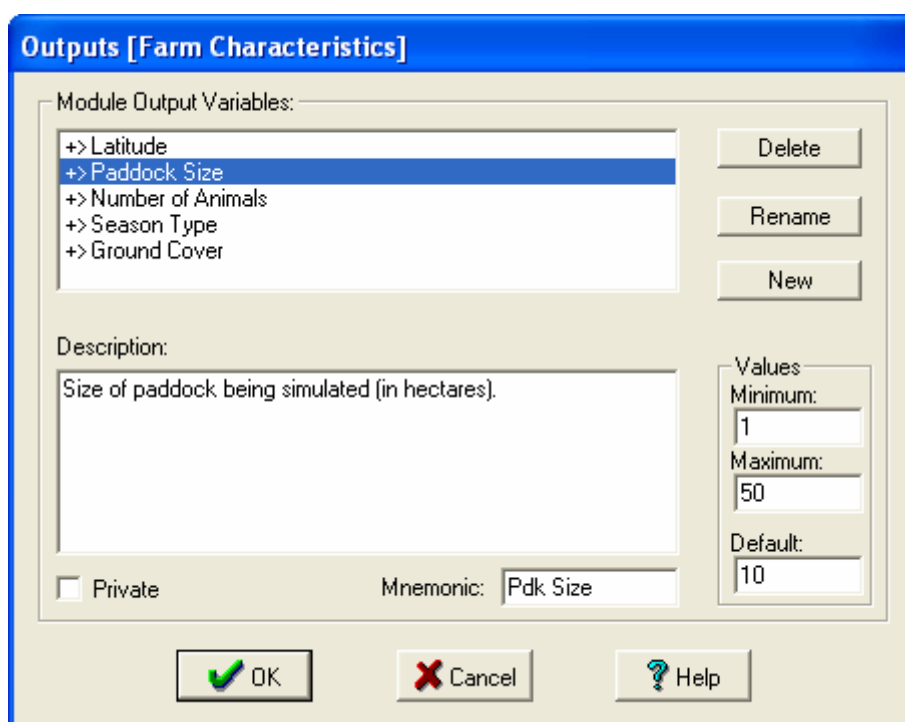
If **automatic linking** is selected in the Options dialog, DYMEX will attempt to link inputs automatically where possible. This is done by matching the names of available variables to the name of the inputs. An informative message that indicates which links have been made is always displayed. It is possible that a variable will be incorrectly linked, or that a suitable linking variable was unable to be determined. Therefore it is important that the message is carefully examined and the correct links made manually if necessary.

Some module inputs have additional features, which will be described when describing the corresponding module.

4.7 Module Outputs

Each module has one or more outputs, which the user may choose to use or ignore. The Module Outputs dialog is used to configure each of these outputs to the current model's requirements (Fig. 4-7) for most modules (an exception is the **Lifecycle** module).

Fig. 4-7 Module Outputs dialog



A list window at the top left shows the outputs available from the module. The symbols “+>” precede each of the outputs that were selected for use in this model. The output that is currently being worked on is highlighted. The button at the top right is used to select a currently unselected output, or to deselect a currently selected output (its label will change from “Select” to “Unselect” accordingly). Double-clicking on the variable name in the list has the same effect as using this button. The “Rename” button can be used to change the name of the variable. The “New” button is only available with some types of modules (for example, **QueryUser** or **DataFile**), and is used to add a new output variable to the list.

The window below the list allows a description for the currently selected variable to be provided. This is useful in documenting the model, and to help the user of the model to understand the precise meaning of each output variable. An abbreviated name (mnemonic) may also be provided for the variable, which will be used in such places as table column headers, where space is limited. The **Simulator** will generate its own mnemonic if one is not

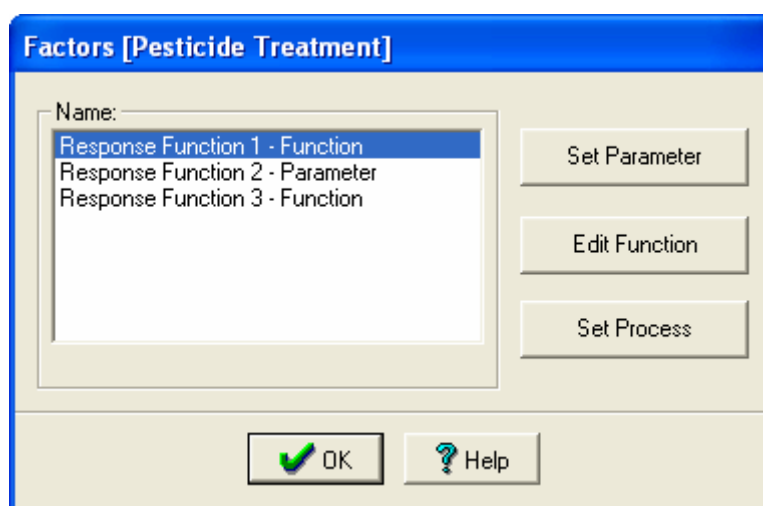
provided, but these tend to be somewhat cryptic. The small box labelled “Private” should be checked if the output variable is not to be made available for inclusion in tables, maps or graphs (i.e., it is for internal use in the model only). This can be useful to reduce the clutter in large models with many variables.

The three “Values” boxes are not available for all modules. Where they are used, they allow limits to be set to the values of the output variables, and a default value to be provided if appropriate.

4.8 Module Factors

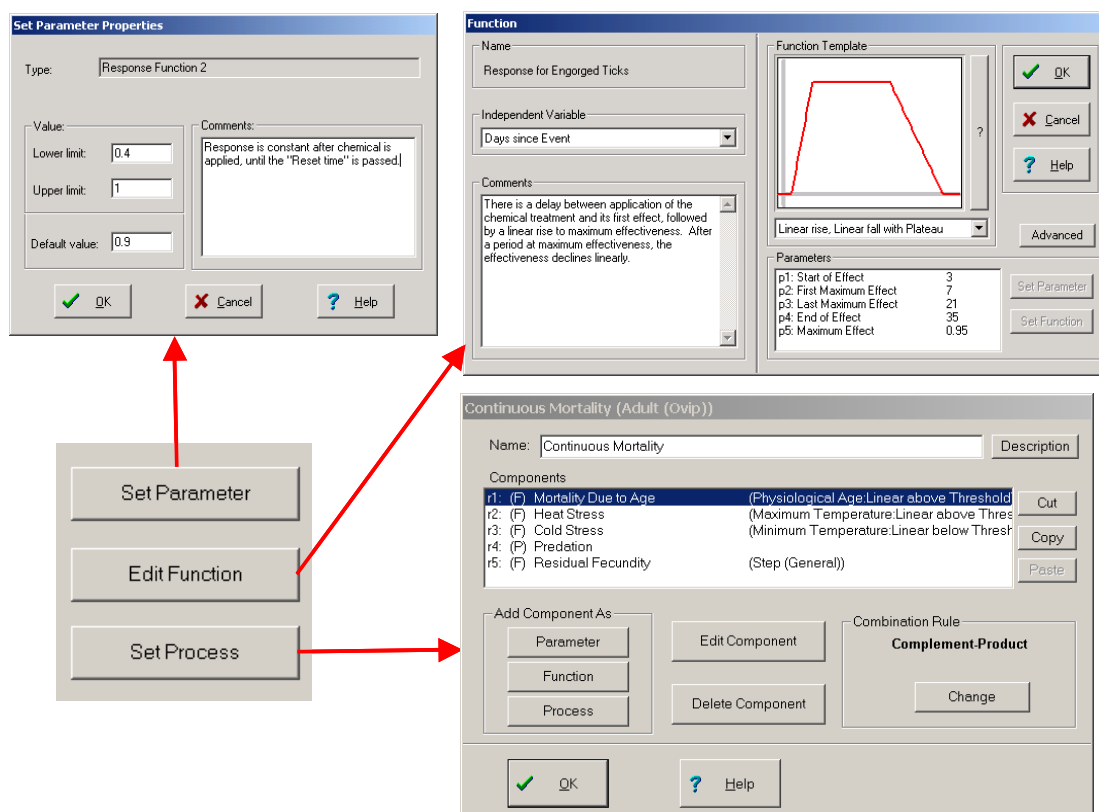
Some modules can be adjusted by setting the values of one or more *parameters*, or replacing these parameters by *functions* or *processes* (see Section 2.3, *Functions, Processes and Parameters*). This is achieved using the Factors dialog (Fig. 4-8). Note that **Lifecycle** processes do not use this dialog, but are set from the Lifecycle diagram.

Fig. 4-8 Module Factors dialog



The Factors dialog lists the module factors (“parameters”) in a window to the left. Any of the factors that have already been set are followed by one of the words “Parameter”, “Function” or “Process” to indicate the type of factor chosen. Three buttons on the right allow the selected factor to be set or edited as one of these factor types. The dialogs corresponding to each of the three buttons are shown in Fig. 4-9. Note that if either a process or function is used, each of the “parameters” within that process or function may in turn be represented by either a function or a parameter.

Fig. 4-9 Module Factors dialog, showing the dialogs corresponding to each type of factor



4.9 Module Settings

The Module Settings button results in a dialog that allows various settings pertaining to that module to be specified. Each module will have a unique dialog, and these are described within that module's documentation.

4.10 Module Description

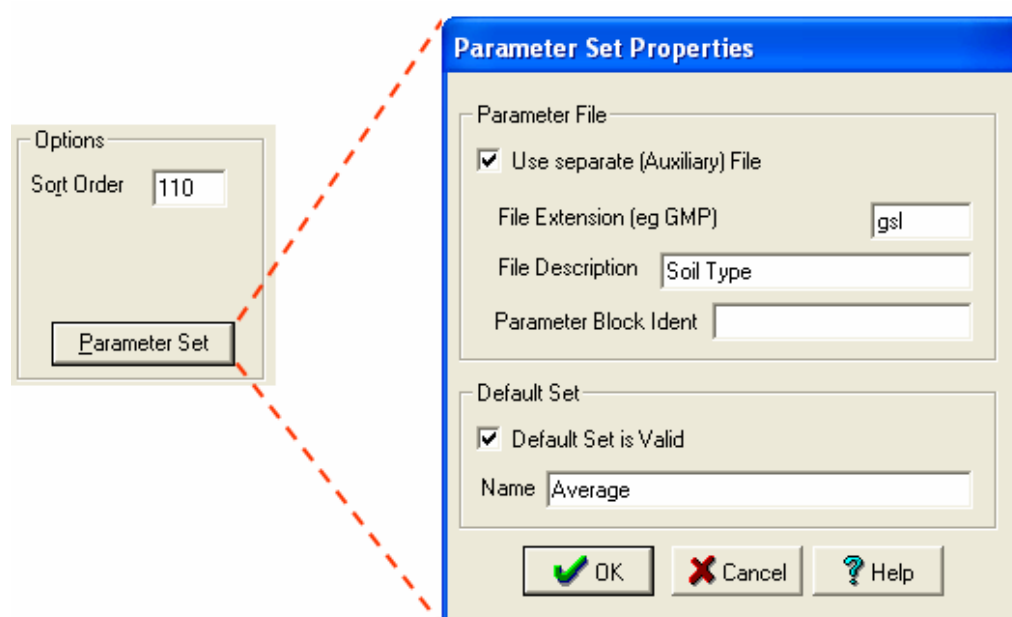
A module description, consisting of a block of text, may be supplied. In the *Simulator*, this description will be automatically labelled with the module's name and appended to the general Model Description. This is a good place to put any information about the module in general (for example, its purpose, a general overview of its operation, etc.). More specific information should be placed into the appropriate comment fields within processes, functions and parameters.

4.11 Parameter Set properties

As described earlier, some modules can be adjusted by setting the values of one or more *parameters*. For these modules, the set of all the parameters of the module is called the *Parameter Set*. The properties of the Parameter Set are

specified using the **Parameter Set Properties** dialog, which is accessed from the **Parameter Set** button in the **Options** panel of the **Module** dialog (Fig. 4-10). Note that for many models, the default Parameter Set properties will be adequate. In that case, the parameters are placed into the model's *Default Parameter File* (which has the same path name as the model, but with extension .gmp). Sometimes, however, it is convenient to place each Parameter Set into a separate, named file (an *Auxiliary Parameter File*) so that it can be manipulated as a unit. In the example shown in Fig. 4-10, a Soil Moisture model has its Parameter Set specified to exist as a separate file. Thus a user of the model could perhaps provide parameter files corresponding to *Sand*, *Loam* and *Clay* soils.

Fig. 4-10 Parameter Set Properties dialog, showing an *Auxiliary Parameter File* specified, with the *Default Set* indicated as valid and named "Average"



➤ **To change the properties of a module's Parameter Set**

- 1.** Click on the module's **Parameter Set** button to open the **Parameter Set Properties** dialog.
- 2.** Check the **Use separate (Auxiliary) File** button if the parameters for this module are to be read from an *Auxiliary Parameter File* and continue to step 3. Otherwise, uncheck that button and click on **Ok** to exit the dialog.
- 3.** Specify the file extension that will be used for the module's Parameter Files by typing it into the **File Extension** edit box.
- 4.** Specify a **File Description** that summarises what a Parameter Set represents. For example, *Soil Type* is used for the above example.

A parameter block in a file is linked to the correct module by a "block identifier" in the Parameter File that is normally the same as the module's name. Sometimes it is useful for two separate modules (with the same type and structure) to be able to share parameter files. In that case, the sharing

modules must define and use a common “block identifier” (so that the parameter files can be recognised by either module).

5. If required, change the module’s “block identifier” from the default by specifying a new identifier in the **Parameter Block Ident** edit box. Otherwise, leave the edit box empty.

The *Default Set* of parameters is the set of parameter values specified in the **Builder**. There may be circumstances where these parameter values, though valid individually, do not form a valid Parameter Set. The lower panel in the **Parameter Set Properties** dialog allows the model constructor to specify how the *Default Set* is to be treated.

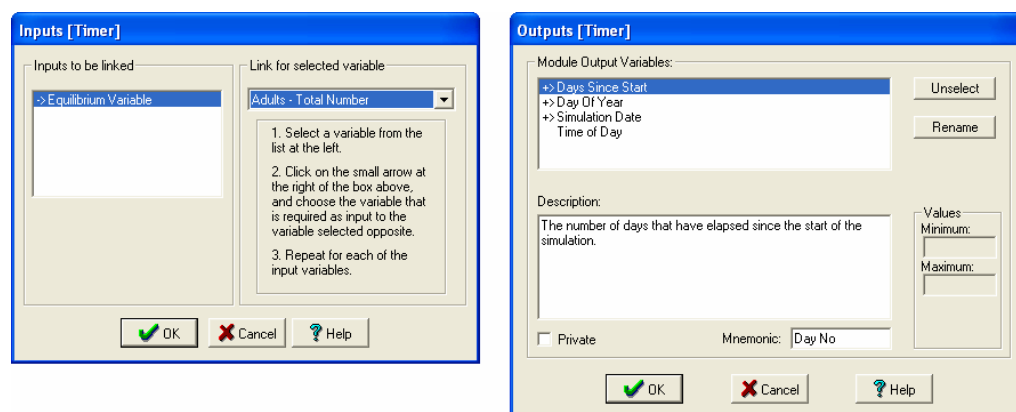
6. If the *Default Set* is a valid set of parameters, check the **Default Set is Valid** button and provide a **Name** for the default set (such as Average Soil in our example).
7. Click on **Ok** to return to the Module dialog.

5. The Timer Module

What is the Timer?

The **Timer** performs the timekeeping functions of the model. When a new model is created in DYMEX, the Timer module is automatically added to the model. The Timer is a mandatory component of the model and cannot be removed.

Fig. 5-1 The Timer module Input and Output dialog boxes.



Input Variable

The Timer module uses a single, optional, input variable: the “**Equilibrium Variable**”. This variable is used during a run of the model within the **Simulator** to determine when the run has reached an equilibrium condition. If the capability to do equilibrium runs is required in the model, the **Equilibrium Variable** input should be linked to the required variable by using the drop down list. In Fig. 5-1, the **Equilibrium Variable** is linked to *Adult Density*.

The equilibrium conditions are set within the *Simulator* and the *Simulator* guide has a full description on how to set the conditions.

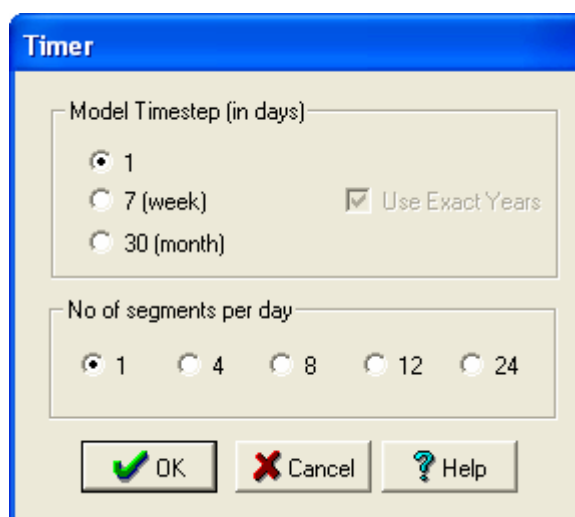
Output Variables

There are four output variables available from the **Timer** module: **Days Since Start**, **Day of Year**, **Simulation Date** and **Time of Day**. **Days Since Start** is the number of days since the start of the simulation. **Day of Year** is the number of days that have passed since the start of the current year (i.e., from, and including, January 1 to the current simulation date) in the simulation, and **Simulation Date** is the date currently being simulated. The **Time of Day** output gives the number of hours that have passed since the start of the current day in the simulation. Normally, this variable would not be selected, as it would always return 0. However, if a segmented timestep mode is used, the variable may be useful.

Which Output Variables should I use?

The choice of output variable is determined by what other modules are required for the model. Note that it is not necessary to make a choice at this point and output variables can be selected or unselected later. In a simple model, it may not be necessary to select any **Timer** output variables. **Simulation Date** is useful as an output if you are using a **Datafile** or **Metbase** module, as it allows synchronization and checking of the data file dates. It is required if you want to report times in model output as dates. **Days Since Start** is the basic time reference used within DYMEX, but may not be required as output very often. However, at least one of the two variables (Days Since Start or Simulation Date) should always be selected for output to act as a time-based x-variable for charts. The **Day of Year** variable is useful if simulations are to be run over multiple years and certain actions (for example, management events) are to be applied at the same time each year.

Fig. 5-2 The Timer module's Setup dialog box.



Setup

Currently there are only three basic timesteps available for a DYMEX model (1, 7 or 30 days, *see* Fig. 5-2). The selected timestep is used throughout the simulation. However, if the 1-day timestep is selected, it is possible to break the daily timestep into a number of segments, which can then be used to update selected modules more frequently than daily. This is termed the 'segmented timestep' mode. The choice of timestep will have no effect on the operation of

most modules. Among the exceptions are the file readers (**DataFile** and **Metbase**) and the **Lifecycle** modules. In the **Metbase** module, for example, if you are using daily rainfall data and the model has a seven-day timestep, DYMEX will calculate the total rainfall for the week and use that figure in the model. In the **Lifecycle** module, process rate parameters will almost certainly need to be different in equivalent models with daily and weekly or monthly timesteps.

If a daily timestep is chosen for the model, this timestep may be divided into a number of segments for updating selected modules more frequently than at the daily interval. This allows effective timesteps as short as 1 hour to be used.

If either a weekly or monthly timestep is chosen, the **Use Exact Years** option becomes available. If selected, the timesteps are updated in such a way that there are exactly 52 weeks in a year for the weekly timestep, and 12 months in a year for the monthly timesteps. This is achieved by adding days to the simulation where required. For example, a 7-day timestep would normally give $7 \times 52 = 364$ days in a 52 week period. To make that period an exact year, DYMEX makes the first week an 8-day week (by incrementing the first 3 Timer output variables by 8 days after the first timestep of each year). If the current year happens to be a leap year, the last week in February is also made an 8-day week. Similarly, five (or 6, in the case of a leap year) 31-day months are used if **Use Exact Years** is selected for a monthly timestep.

See Process Rates and the Model Timestep, page 50 for more information on the effect of changing the timestep on model processes.

6. The Lifecycle Module

What is the Lifecycle module? A number of intrinsic and extrinsic factors affect the population dynamics of any species. Different species have different life history characteristics such as developmental characteristics, how and when mortality factors operate and the timing of reproduction. The **Lifecycle** module is a flexible component that can be adjusted to simulate a wide range of species. In DYMEX, a **Lifecycle** consists of one or more **Lifestages**. Multiple **Lifecycle** modules may be included in a DYMEX model, and these different lifecycles can interact.

Lifecycles do not need to be linear, but may have branches and nested stages. This allows complex phenomena such as a diapause stage in an insect, cyclical dormancy in a seed, or the flowering in a plant to be modelled conveniently.

Note that a **Lifecycle** module does not have to be a complete lifecycle. For example, the module could be used to model a rice crop from planting of seedlings to harvesting.

What is a Lifestage? A **Lifestage** contains all the individuals in a simulation that are at a particular stage of development, whether development is measured in terms of chronological age, physiological age, size or some other measure. DYMEX is

well suited for modelling insects, which do not have continuous development. The cuticle, which makes up the exoskeleton of insects, prevents continuous growth. A moult has to occur at intervals to form a new cuticle of larger surface area, and intervals between moults form a natural division of the lifecycle into stages.

Lifestages in DYMEX can also be used to model organisms with continuous development patterns. Multiple lifestages can be defined in many other organisms, e.g. frogs (eggs, tadpoles, adults), plants (seeds, seedlings, juveniles, adults, flowers, fruits) and so forth.

How many stages are needed?

The identification of meaningful stages depends on the purpose or aims of the modelling exercise and the biology of the organism. It is sometimes possible to group lifestages. If parameters for successive “natural” stages are the same or little detailed data is available on individual stages, then a single Lifestage could be used to model a number of successive actual stages. For example, two or more larval stages (instars) in an insect species could be grouped to form say an ‘immature’ stage. DYMEX makes it relatively easy to insert lifestages later, so in many cases it may be convenient to start with grouped stages early in model construction, and to divide them into separate stages later if it is found to be necessary.

Ordering processes within the lifestage



In the current version of DYMEX, all continuous processes within a lifestage, occur at the same time. If a feature of that cohort (e.g. Physiological Age, Number, etc.) drives a process within a cohort the value will generally be obtained from the *previous* timestep (see however, *Immediate Update Cohort Variables*, Section 6.18). Transfer always occurs once all other processes have occurred within the lifestage. Establishment processes occur before any other processes have occurred within the lifestage, while Exit processes occur during stage transfer. In most cases, it will not matter whether a process occurs as on exit from one lifestage, or on entry to the next lifestage. Where it becomes important is where there are branching lifecycles with two stage transfers entering or leaving a lifestage. In these circumstances, the modeller must decide when the process needs to be updated and place the process accordingly. Misplacing the process may result in the process being triggered too frequently or infrequently, or to the wrong cohorts.

Establishment and Exit processes

➤ **To add a Lifecycle Module**

- 1.** Select **Model** menu, choose **Add Module...** and choose to add a **Lifecycle** module.

This creates a new **Lifecycle** module containing a single **Lifestage**, and displays it in the *Lifecycle Window*. A new menu (“*Lifecycle*”) with choices relating to the lifecycle is now available on the menu bar.

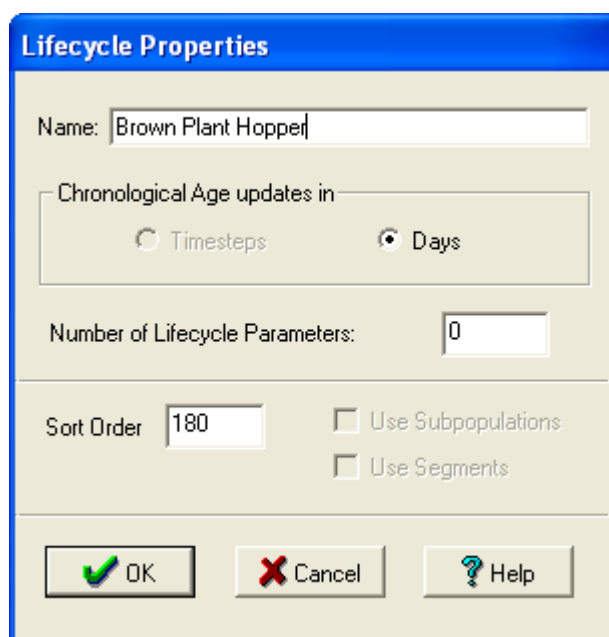
- 2.** Select the **Lifecycle** menu and choose “**Properties...**” to open the Lifecycle Properties dialog (Fig. 6-1).
- 3.** Type a name for the lifecycle into the Name edit box. The name should describe the species being represented (eg, “**Brown Planthopper**”).

4. Choose the method to be used to update the **Chronological Age** Cohort Property.

By default, *Chronological Age* (see Section 6.4) is updated in units of days in models created with Version 2 or above of DYMEEX, while models created with Version 1 update this Cohort Property in units of timesteps (of course, these amount to the same for models with a daily timestep). A model created with Version 1 will still have the *Chronological Age* updated in units of timesteps, but this can be changed in this dialog.

5. Set the Sort Order Constant to a value that places the module into the correct position in the list of modules.
6. If factors will be defined for the lifecycle, specify the number of factors that will be used in the **Number of Lifecycle Factors** box (see Section 6.22).
7. If the model population is divided into sub-populations, and the lifecycle is to use those subpopulations, the **Use Subpopulations** box must be checked.
8. If a segmented timestep is being used, check the **Use Segments** box if the Lifecycle module is to be updated at smaller than daily intervals.
9. Click **Ok** to exit the Lifecycle Properties dialog.
10. Select the “**Description...**” menu item from the **Lifecycle** menu and provide a description of the lifecycle and its properties.

Fig. 6-1 The Lifecycle Properties dialog.



In the following sections, the individual components of lifestages are discussed in detail, with particular reference to Lifestage types, Cohorts (Section 6.4),

Environments (Section 6.6) and Processes (Section 6.7). The various dialogs that are used for specifying processes are then dealt with in Sections 6.10 to 6.11. The uses of each process type are then discussed in detail in Section 6.12 (*The Development Process*), 6.13 (*The Reproduction Process*), 6.14 (*The Mortality Process*), 6.15 (*The Transfer Process*) and 6.18 (*User-defined Cohort Variable Processes*), with examples from vertebrates, plants and insects.

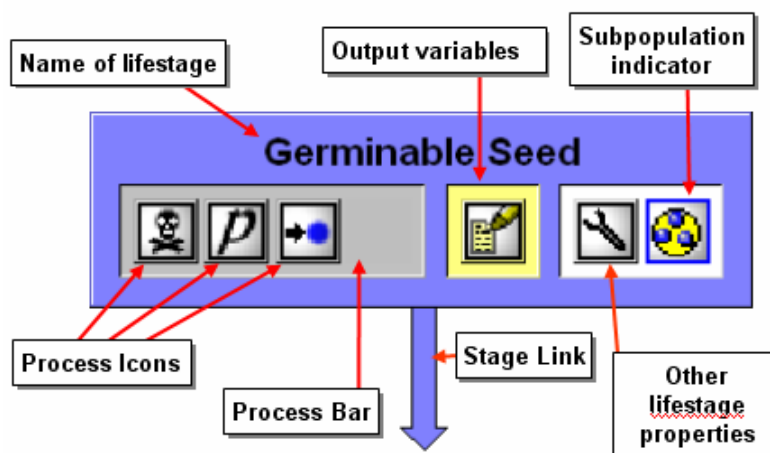
6.1 The Lifestage and its components

The Lifestage is shown in the *Lifecycle Window* as a rectangular box, containing the lifecycle components (Fig. 6-2). If the lifecycle being depicted contains more than one lifestage, a light-red border surrounds the currently selected stage. Selecting menu-items that apply to a lifestage such as “Development...” will refer to the selected stage.

What's in a lifestage?

Name of Lifestage: A label that defines the stage, e.g. “Seed”, “Egg”, “Juvenile”, and is included in the default names that DYMEX gives to output variables (*see Naming Variables and Modules, page 19*). Note that the blue colour of the *Name* panel is replaced by red for stages that contain a reproductive process.

Fig. 6-2 An example lifestage in a lifecycle with labelled process buttons.



Processes: The processes being used in a particular stage are represented by icons on the grey Process Bar. The Stage Transfer process (and Reproductive process in reproductive stages) are always depicted, but are marked with a red cross if they have not yet been specified. If incompletely specified processes are present, the Process Bar will be coloured red and an “incomplete” icon will be displayed. Fig. 6-14 (page 50) shows all the process buttons that may be visible.

Output Variables: These are the variables that will contain the results of the simulation run for that stage. Typical output variables might include *Total*

Number (Population), *Development Time*, *Average Size*, etc. The lifestage output variables are obtained by combining the outputs from individual cohorts in the stage (see Lifestage Output Variables, page 99).

Other lifestage properties: This button is used to set up various lifestage properties. Here a lifestage can be made an **Endostage**, and the resource variable can be set. The resource variable is the denominator used in density calculations for the lifestage, and could include such variables as paddock size or plant biomass.

Subpopulation indicator: This icon, if present, indicates that the lifestage makes use of the models sub-population structure (i.e., the individuals that comprise the life stage may be split among several sub-populations).

Stage Link: The large blue arrow leaving the lifestage is the “Transfer Link” to the next lifestage (i.e., it represents individuals graduating to the next lifestage). In contrast, the “Reproductive Link” (a red line) represents the generation of new individuals via reproduction. Note that functions that affect the properties of the stage links can be accessed by left-clicking on the appropriate link arrow and selecting the function from the popup menu.

Environment: In the present version of DYMEX, the Environment is used as a device for grouping variables that are available in the same situations placed into the same environment. Note that the environment is not accessible from the Lifecycle diagram but must be selected from the menu (see *What is an Environment in DYMEX?* page 46).

A new lifestage may be added below or above an existing lifestage by left-clicking on the part of the lifestage not taken up by icons and selecting “Add Stage After” or “Add Stage Before”, respectively from the resulting popup menu. These Lifestage menu may also be used for this with the additional ability to create new lifecycle branches (see below). If no more lifestages are needed the processes associated with each lifestage can now be edited.

The lifecycle pathway is completed by left-clicking on the lifestage and selecting **Create link** from the resulting popup menu. A sub-menu then opens out and the required reproductive link can be selected from those listed under the “Reproductive Links” menu heading.

6.2 Lifecycle and Lifestage types

Three different lifestage types are the basic components from which different lifecycles in DYMEX can be created. A **Normal** lifestage does not contain a reproductive link out of the stage, and is not contained within another stage. It is shown as a blue box in the lifecycle diagram. A **Reproductive** lifestage has a reproductive link exiting out of it, and is distinguished by its red colour. An **Endostage** is a stage that is contained wholly within another stage and is distinguished by its light-blue colour.

Fig. 6-3 Three different lifecycles in DYMEX. (a) is a simple lifecycle, (b) shows a branched lifecycle, while (c) uses “endostages” to model the reproductive phenology in a plant

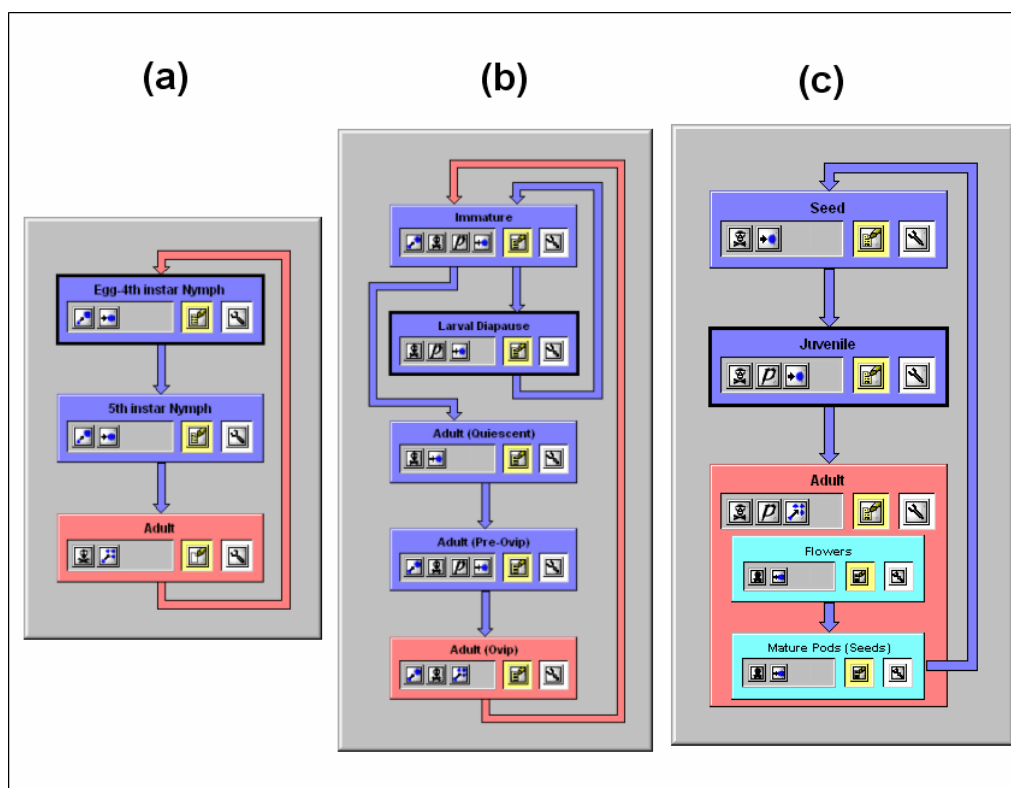


Fig. 6-3 shows three different lifecycles constructed using these components. The first lifecycle (a) is simple, with two Normal stages followed by a Reproductive stage. The second lifecycle (b) uses a lifestage branch to simulate diapause (i.e., individuals from the “Immature” stage may move to either the “Larval Diapause” stage or the “Adult (Quiescent)” stage). The final lifecycle (c) uses two Endostages to simulate the onset of flowering and seed pod development in a plant.

6.2.1 Branching Lifestages

Branching (Fig. 6-3b), which is the use of more than one Stage Transfer exit per stage, allows some very complicated lifecycles to be created in DYMEX. There are some restrictions placed on the use of branches. No more than two standard (non-reproductive) links may exit from any one lifestage. Two branches leaving a stage may not terminate at the same destination stage (this limitation includes reproductive links – i.e., it is not permitted to have both a standard link and a reproductive link connecting two lifestages).

➤ To create a new stage as an alternative exit from a stage

1. Select the lifestage from which the branch will exit by clicking on its blue or red area with the mouse.

2. From the **Lifecycle** menu, choose **Add Stage After....** From the resulting dialog, select the option “**Current Stage → NEW STAGE**” (The alternative option would insert the new stage into the existing stage link.

➤ **To create a new link between two existing stages**

1. Select the lifestage from which the Stage Link will exit by clicking on its blue (or red for reproductive stages) area with the mouse.
2. From the **Lifecycle** menu, choose **Create new Stage Link....** Make sure the correct source lifestage is selected and select the destination lifestage from the list. Click on **Ok** to close the dialog.

The lifecycle window will then show the new Stage Link between the stages. Note that a convoluted lifecycle diagram can often be simplified by rearranging the lifestages in the diagram (which, of course, does not change their logical position within the lifecycle). This is done using the “**Lifecycle|Re-order stages...**” menu item.

6.2.2 Endostages

As indicated earlier, an **Endostage** is a lifestage that is contained wholly within another stage (the “Container” stage), this being shown explicitly in the Lifecycle Diagram. Thus Endostages are mostly suited for simulating the production of flowers or buds on a plant (via the reproductive process), and the development of these structures towards independence (for example, seeds). There are also cases where Endostages may be useful for simulating progeny development in animals where there is *in vivo* competition such as in tiger sharks. Fig. 6-3c shows the way that Endostages are typically used in DYMEX. The critical property of an Endostage is that it is a part of its Container stage in the sense that if (for example) the adult plant is killed, so are its flowers, buds and pods which are insufficiently developed to be capable of independent survival. However, the reverse is not true. By using an Endostage it is possible to allow environmental conditions to act on progeny independently of the parent e.g. frost or drought mortality, whilst also simulating the effects of parent processes (e.g., mortality) on the progeny.



In this version of DYMEX, only reproductive stages can be Container stages and their Endostages must immediately follow via the reproductive link (both logically and as depicted in the Lifecycle window). Note that the use of Endostages can result in a large increase in execution time for a model, due to the fact that during model execution, Endostages tend to generate many more cohorts than other types of stages. The reasons for this are discussed in detail in Section 5 of the Simulator User’s Guide. In all other ways, an Endostage is treated the same as any other lifestage.

➤ **To create an Endostage**

1. Create the required lifestage in the usual way.


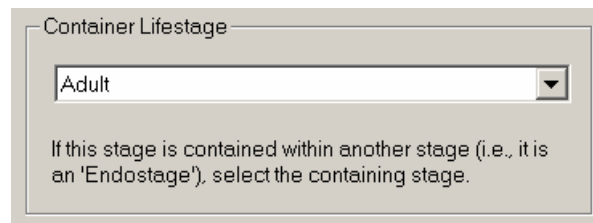
2. Ensure that the selected stage is below the stage that will become its Container stage (the reproductive stage) in the Lifecycle window representation. Use the “**Lifecycle|Re-order stages...**” menu item to reorder the stages as required.
3. Click on the “Other lifestage properties” icon () and from the “Container Lifestage” list near the bottom of the resulting dialog, select the appropriate **Container Lifestage** (Fig. 6-4). Note that only those lifestages that are valid candidates as Container of the Endostage are listed.

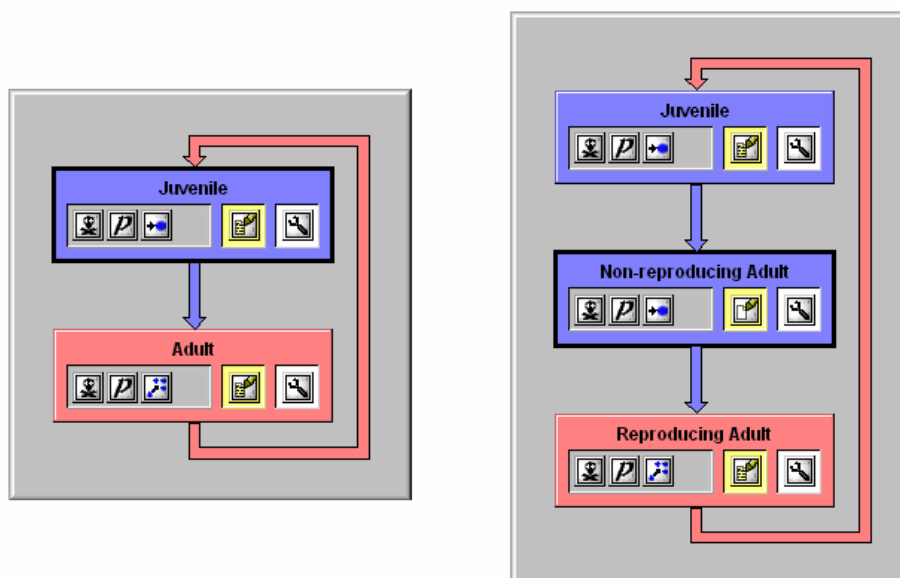
Fig. 6-4 Selecting the Container Lifestage for an Endostage



Lifestages represent various phases within a lifecycle of the modelled organism. A lifestage can be defined as a group of individuals, all of which are affected by the same processes during a particular period of their life. Annual plants are an example of a simple life history, which can often be modelled with as few as two lifestages: *seeds* and *plants*. Similarly, vertebrates may be modelled as *juveniles* and *adults*. If different processes affect adults that are reproducing, another lifestage can be added. *Note: in this conceptual model, once a non-reproducing adult becomes a reproducing adult it cannot revert to being a non-reproducing adult (Fig. 6-5) without the addition of a branching pathway.*

Lifestages do not necessarily have to be used to model obvious discrete ‘stages’ within lifecycles but can also be used to model ‘phases’ within actual stages. The second lifecycle in Fig. 6-5 is an example of where another lifestage is used to model such a ‘phase’ within a lifestage. As another example, a lifestage could be used to model infection where the individuals go to another lifestage when they become infected.

Fig. 6-5 An example of two lifestages modelled within DYMEX: the first being a very simple model while the second is an extension of the first model where the reproducing adults are affected by different processes.



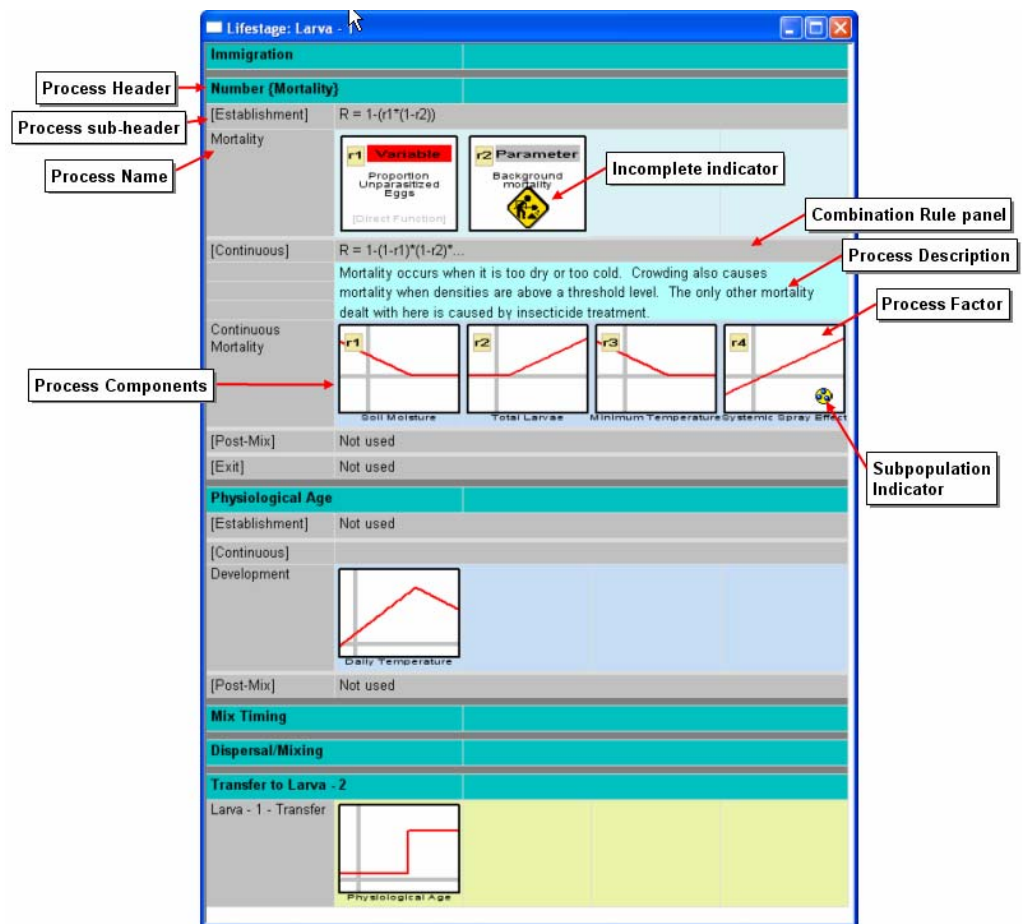
6.3 The Lifestage Window

The Lifestage window displays all of the processes for a single lifestage in a graphical window (Fig. 6-6). Any process can have factors added, changed or deleted, its combination rule adjusted and comments added from this one place. Much of the work of creating and editing a model will be done from within the lifestage window.

Processes are arranged within the lifestage window in a spreadsheet-like grid arrangement, with similar types of processes grouped together under one **Process Header**. For example, several types of mortality processes (i.e., processes affecting the Cohort variable *Number*) are available, and these are all grouped together under the heading “*Number {Mortality}*”.

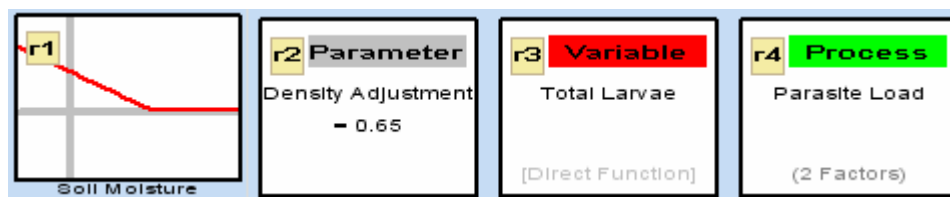
Let us examine a typical Lifestage window (Fig. 6-6). The name of the lifestage is indicated in the window’s caption (in this case, *Larva – 1*). The first thing to note is that a number of broad, green, labelled bars divide the window into a several sections. Each of these bars is a **Process Header**, with its label indicating the process that is displayed under that header. In the example, The *Immigration*, *Mix Timing* and *Dispersal/Mixing* processes are not used at all in this lifestage. A glance at the ***Number {Mortality}*** section shows that both “establishment” and “continuous” mortality process are being used in the lifestage, while the “exit” and “post-mix” mortality processes are not being used. The “establishment” mortality has two components (factors), one of which is a “Direct” function (or variable), the other a parameter. The “continuous” mortality has four factors, all functional in nature.

Fig. 6-6 A Lifestage window.



Each process factor is shown by a panel within the corresponding “Process Components” row. If more than one factor is present, each factor panel contains the factor label (e.g., “r1”, “r2”, etc) at the top left. Four different types of factor panels may be present, as shown in (Fig. 6-7):

Fig. 6-7 A “Process Components” row within the Lifestage window, showing the four different types of factor panel.



1. Function panel

The function panel is shown for a function process component. It illustrates the function shape being used, and shows the driving variable on the x-axis of the graphic. Note that depending on the choice of parameter values, the functional shape may not be an exact representation of the chosen functional shape.

2. *Parameter panel*

The parameter panel is shown for factors that are parameters. It indicates the name of the parameter, as well as its default value (or the centre of the allowed range if no default value is specified).

3. *Variable panel*

The variable panel is shown for factors that use the *Direct* function (i.e., the factor's value is the same as the value of the driving variable). The name of the driving variable is indicated in the panel.

4. *Process panel*

A process panel is depicted if the corresponding factor is itself a process. The name of the process is shown in the panel, and the number of factors that it consists of is indicated at the bottom of the panel.



Incompletely specified factors are shown by an “incomplete” indicator placed on the corresponding panel, as shown in the second establishment mortality factor in Fig. 6-6.



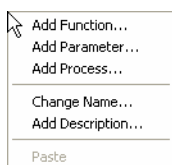
Factors whose values will be different for different sub-populations are indicated using a “sub-population” indicator (see factor “r4” in the continuous mortality process of Fig. 6-6).

Processes that have more than one factor (or single-factor processes that use a user-defined Combination Rule) have their combination rule shown in the right-hand panel of the process heading. The factor labels in the combination rule (e.g., r1, r2, etc) correspond to the factor labels shown in the factor panels.

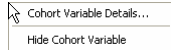
6.3.1 Lifestage Window Operations

Clicking on different parts of the lifestage window will produce corresponding actions. This section describes the actions that the user can perform from the Lifestage window.

1. *Process Header*



When the left side of a “Process Header” bar is clicked, the resulting action depends on whether or not the corresponding process (or processes) is associated with updating the values of Cohort Variables. For processes that are not directly associated with a Cohort Variable, a menu such as the one shown at the left is displayed. It allows a new factor to be added, with a choice of function, parameter or process being available. The new factor will be added after any previously existing factors for this process. The “Change Name” and Add Description” options will only be available if the process contains factors. They allow the name of the process to be changed and a description for the process to be provided. The “Paste” option will be available only if another factor has previously been copied or cut.



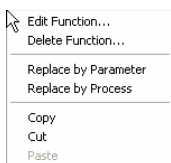
When the Process Header bar of a process group that updates the value of a Cohort Variable (other than “Number”) is clicked, one of two things will happen. (1) If the process group does not yet contain any factors (i.e., none of its component processes are used), the menu shown at the right is displayed. Selecting “Cohort Variable Details” will open the Cohort Variable dialog (see Section 6.18), where details of the corresponding Cohort Variable can be examined or changed. Selecting “Hide Cohort Variable” will remove the whole section dealing with that Cohort variable from the lifestage window for this lifestage. This is useful to avoid clutter in the Lifestage Window in cases where a Cohort Variable does not require updating in a particular stage. Note that this action can only be undone from within the **Lifestage Settings** dialog by un-checking the **No Processes** box in the **Cohort Properties (local settings)** panel. (2) If the process group already contains a process with one or more factors, clicking on its Process Header bar just displays the “Cohort Variable” dialog, from which details of the corresponding Cohort Variable can be examined or changed.

2. Process sub-header or name panel



Clicking on either a process sub-header or process name panel displays a popup menu like that shown on the left. It allows a new factor to be added, with a choice of function, parameter or process being available. The new factor will be added after any previously existing factors for this process. The “Change Name” and “Add Description” options will only be available if the process already contains factors. They allow the name of the process to be changed and a description for the process to be provided. The “Paste” option will be available only if another factor has previously been copied or cut. The “Cohort Grouping” choice is shown only for the Dispersal/Mixing process. It is used to specify how cohorts are combined after dispersal. Use of this option is described in Section 6.17.3 (*Dispersal and Cohort Grouping*).

3. Process Factor panel



Clicking on a Process Factor panel displays a popup menu similar to the one shown on the left. Selecting the first item allows the factor (function, parameter or process) to be changed. The factor can be deleted from the process by selecting the second menu choice. Note that this will move all the higher factors one position down in the factor list (i.e., if “r2” is deleted, “r3” becomes “r2”, etc). If the factor is a function, the third and fourth menu items allow it to be replaced by a parameter or process, respectively. Equivalent options are shown if the factor is a parameter or process. The “Copy” or “Cut” selections can be used to place a copy of this factor into a special buffer, from which it can then be inserted elsewhere in the lifecycle using the “Paste” selection. Note that when “Paste” is used, the factor is placed immediately to the right of the factor from whose popup menu the “Paste” command was selected.

4. Combination Rule panel

A left-click on the Combination Rule panel will display the **Combination Rule** dialog (see *The Combination Rule*, page 57). Once a combination rule is set, it will subsequently be displayed in the panel. Incorrect combination rules are indicated by the phrase “[Invalid]” in the Combination Rule panel.

5. Process Description panel

The Process Description panel is only displayed if a description has been provided for the process and shows the process description. The description can be edited by clicking on the panel.

6.4 What is a Cohort in DYMEX?

DYMEX does not model the fate of individual organisms. Individuals are grouped into assemblages termed *Cohorts*, where each cohort consists of a number of individuals that belong to the same lifestage, occupy the same spatial unit, and share the same properties, like the time (day/week) they entered a stage. Cohorts are the basic units that are modelled in a DYMEX lifecycle. An example of a cohort would be all the juveniles born on a particular day during the simulation. All the individuals within a cohort experience the same conditions during the course of a simulation.

Table 6-1 Cohort Properties and their associated processes

<i>Cohort Property</i>	<i>Associated Processes</i>
Number	Mortality Stage Transfer Dispersal
Chronological Age	<i>Intrinsic</i>
Physiological Age	Development, Aging, Maturation
Residual Fecundity	Fecundity Progeny Production
Size	Growth

Cohorts have a number of properties (*Cohort Properties*) that are shared by all the individuals in the cohort. A list of these properties is provided in Table

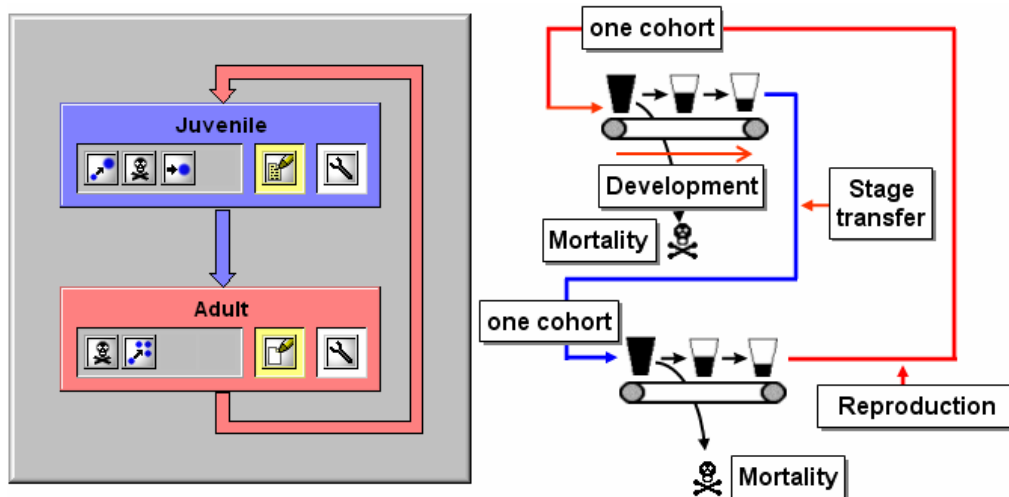
6-1. For example, **Number** contains the current number of individuals in the cohort, while **Physiological Age** contains their current state of development. There is no user-adjustable process associated with the **Chronological Age** property, which always reflects either the number of *days* or *timesteps* since the cohort was created, depending on the settings in the Lifecycle Properties dialog. Note that models created in Version 1 of DYMEX update the **Chronological Age** in *timesteps*, while the default for models created in Version 2 is *days*. The **Residual Fecundity** Cohort Property is only present in reproductive stages. Note that **Size** is not a pre-defined Cohort Property, but must be defined by the modeller if it is required. Up to 32 of these user-defined Cohort Properties may be defined for each lifecycle. Section 6.18 describes how to create or modify a user-defined Cohort Property.

During a simulation, each lifestage may contain many cohorts. The total number of individuals in a lifestage is the total of the **Number** Cohort Property for all the cohorts in that stage. Figure 6-8 simply illustrates how individuals in a cohort are ‘transported’ through a lifestage until they either die or are transferred to another lifestage where they form a new cohort. The full glass represents the full complement of individuals that the cohort is created with. These are depleted by mortality as development moves the cohort along the “conveyor belt”. When development is complete, and if any individuals remain in the cohort, they are used to form a new cohort within the next stage.



It is important to note that any individual can die at any time along the ‘conveyor belt’ of cohort development.

Figure 6-8 A simplified diagram of the function of cohorts in DYMEX.



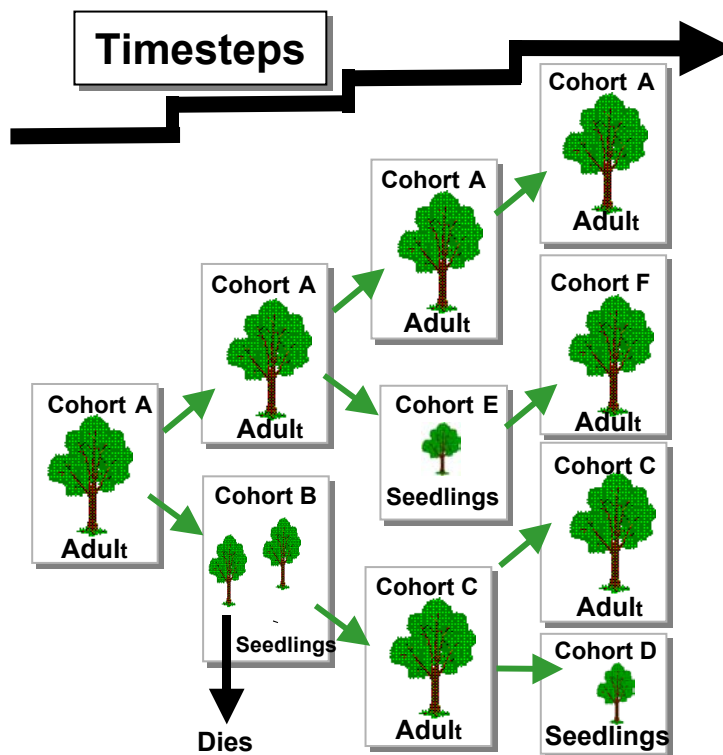
How do cohorts work in practice?

Say we have a model containing a multi-stage insect lifecycle, the first two stages of which are “Juvenile” and “Adult”. We might initialise a simulation by introducing 100 juveniles into the simulation on day 1. These 100 juveniles would form a cohort. During the course of the simulation, the **Development** process chosen by the user acts on this cohort, increasing its **Physiological Age**. At any time step, each member of the cohort has the same **Physiological Age**. If a **Mortality** process has been chosen to apply to the “Juvenile” stage, this mortality process acts on the cohort to reduce its numbers from the initial

100. If mortality is such that the number of individuals in the cohort is reduced to 0 before they are ready to move to the next stage, the cohort is automatically removed from the simulation. If the mortality is lower, the cohort may persist long enough for its members to become eligible to move on to the next lifestage (as determined by the appropriate **Stage Transfer Process**). In this case, a proportion of individuals from this cohort (determined by the **Stage Transfer Process**) graduate to the next stage (“Adult”), where they contribute to create a new cohort. This new cohort will consist of all the new adults for that time step, recruited from a number of different juvenile cohorts, with early transfer from later juvenile cohorts added to late transfer from earlier juvenile cohorts. Similarly, when the reproduction process creates new juvenile individuals from a number of different adult cohorts during a particular time step, these new individuals are placed into just a single juvenile cohort (This behaviour can be changed so that cohorts are transferred between lifestages – see *Cohort and Cohort Variable Transfer*, page 81).

Below is another example of cohort formation in a population of trees (Fig. 6-9). There are two lifestages in this example, adults and seedlings. In the first time step, the one adult tree produces two seedlings in the second time step, which form cohort B. In the second time step one of the seedlings dies and the other one receives enough nutrients to develop to an adult (Cohort C). In addition, in the second time step the adult tree (Cohort A) produces two seedlings (Cohort E). In the third time step, the Cohort E seedlings mature to adults (Cohort F), while the Cohort C adults produce another cohort of seedlings (D).

Fig. 6-9 An example of cohort formation in a population of trees.



The cohort concept is central to the operation of the DYMEX Lifecycle module. In practice we have found that many problems in interpreting DYMEX results are due to a lack of understanding of how cohorts work.

6.5 Cohorts and sub-populations

In a lifecycle model that uses sub-populations, separate cohorts are present for each sub-population. New cohorts created from individuals transferring to the next stage include only those individuals from the same sub-population. Individuals can only be transferred between sub-populations by the *Dispersal (Mixing)* processes and the *Genetic Mixing* process.

6.6 What is an Environment in DYMEX?

In DYMEX, an **Environment** groups variables into sets. This can simplify the construction of a complex model by restricting the available variables when selecting driving variables for processes. Below in Fig. 6-10 is an example of four environments (Global, Canopy, Fruit & Soil). Each lifestage resides within a particular environment. By default a lifestage is assigned to the **Global** environment, which contains all variables.



In simple models, you can probably ignore the environments feature (i.e. use the default **Global** environments for all lifestages)

The example below is an example of the environments that a fruitfly will experience during its lifespan: the larvae live within the fruit, the pupae within the soil and the adults within the canopy.

Fig. 6-10 Example environments on a tree.

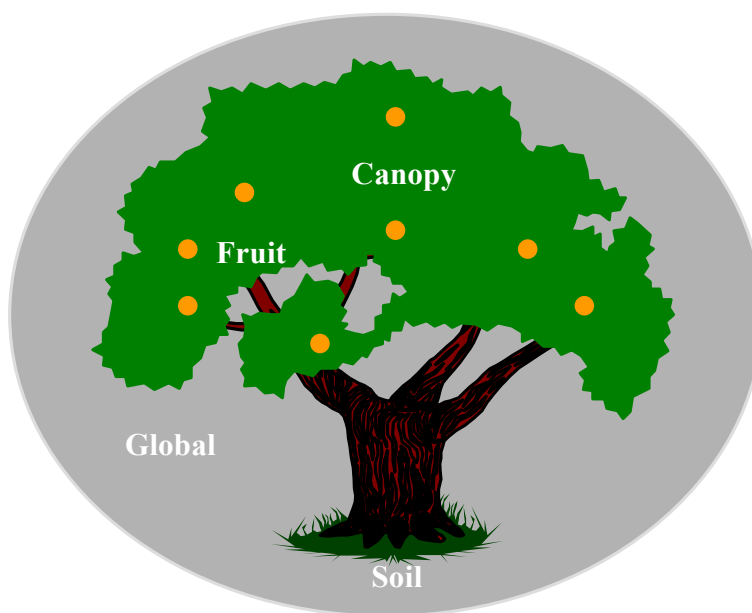


Table 6-2 Example variables associated with the four environments listed in Fig. 6-10.

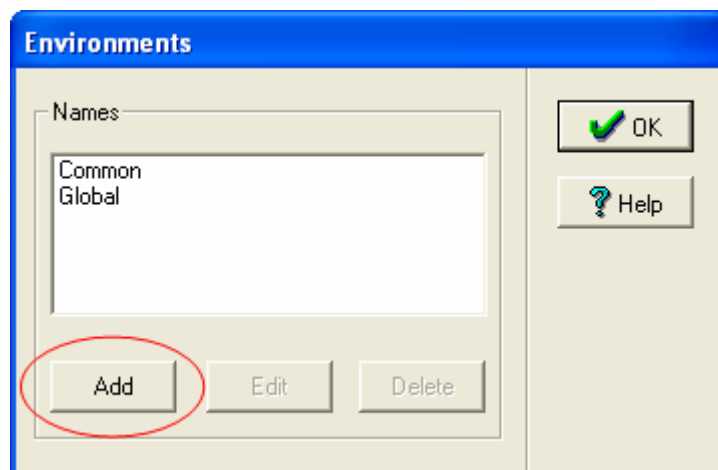
<i>GLOBAL</i>	<i>SOIL</i>	<i>FRUIT</i>	<i>CANOPY</i>
Air Temp.		✓	✓
Rainfall			
Soil Moisture	✓		
Humidity			✓
Evaporation	✓		✓
Soil Temp.	✓		
Fruit Type		✓	
Ripeness		✓	

In Table 6-2 is an example of how a series of model variables could be grouped into four environments. Air Temperature and Evaporation is available from two environments while the other variables are only available from a single environment.

➤ **To add a new environment**

- 1.** Go to the Component Window **Model** menu.
- 2.** Selecting the **Environments...** menu item will make the Environments dialog box appear.
- 3.** To create, edit or delete environments left click on the appropriate buttons within the **Names** box.

Fig. 6-11 Environments dialog box.



4. To create an environment left click on the **Add** button and the environment dialog box will appear.
5. Give a name to the environment within the small box just below the top edge of the **Environment** box (Fig. 6-12).
6. Variables can be added to the environment by either double clicking the variable selected or left-clicking the **Add to Environment** button.

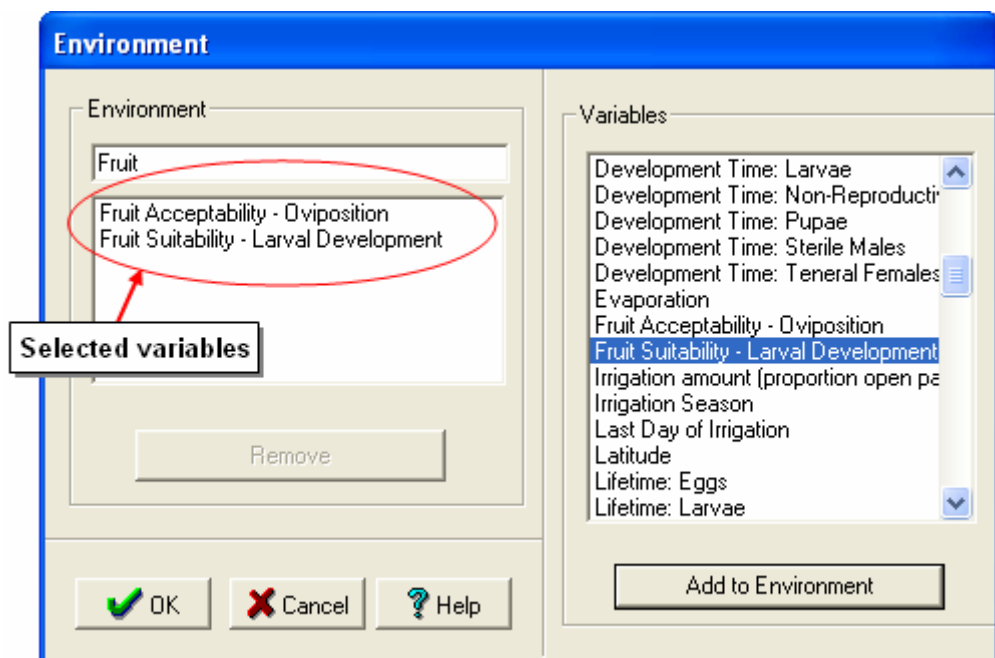


If a lifestage is assigned to an environment other than Global, only those Variables within that environment will be available to the lifestage.

➤ **To remove a selected variable from an environment**

1. Select variable within the variable list box on the left and left click the **Remove** button.

Fig. 6-12 Environment dialog box with open Add dialog box.



It must be emphasised that multiple environments are not necessary in simple models. If your model is becoming so large that the number of variables is becoming confusing, then creating multiple environments can be useful. In a future version of DYMEX, environments will be used more extensively in conjunction with spatial modelling.

Continuing... Once the number of lifestages needed has been decided, the lifestages have been added and their environments specified, the various processes of the lifestages can be edited.

6.7 What is a Lifestage Process?

A lifestage process in DYMEX is the mechanism that changes the value of a *Cohort Property*. For example, the process of **Development** increases the value of the **Physiological Age** at each timestep, while a **Mortality** process removes individuals from the cohort (i.e. it decreases the value of the cohort property called **Number**). Some cohort properties may have more than one process acting on them. A typical example is the **Number** of individuals in the cohort. **Mortality**, **Dispersal** and **Stage Transfer** processes all affect this property.

The description of processes is fundamental to the construction of a DYMEX model (or any other type of model, for that matter). Once the basic lifecycle structure has been determined, we have generated a static picture of our system. By the addition of the processes, which allow individuals to grow, die and progress through the lifestages in the lifecycle, we add the dynamic aspects of the model.

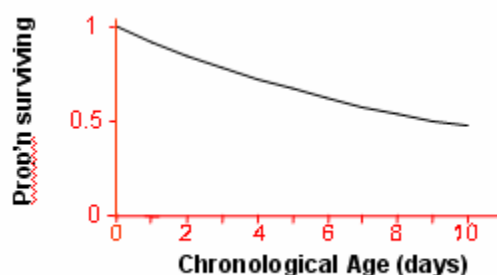


Since processes are so fundamental to the operation of a DYMEX model, a considerable amount of flexibility has been built into them. It is important to understand the structure of the process in order to build a useful model.



A Mortality process might have a constant value of 0.1, thus reducing the number of individuals in the cohort by 10% in each timestep. Over a number of timesteps, the proportion of individuals surviving from the original number that formed the cohort would be as in Fig. 6-13.

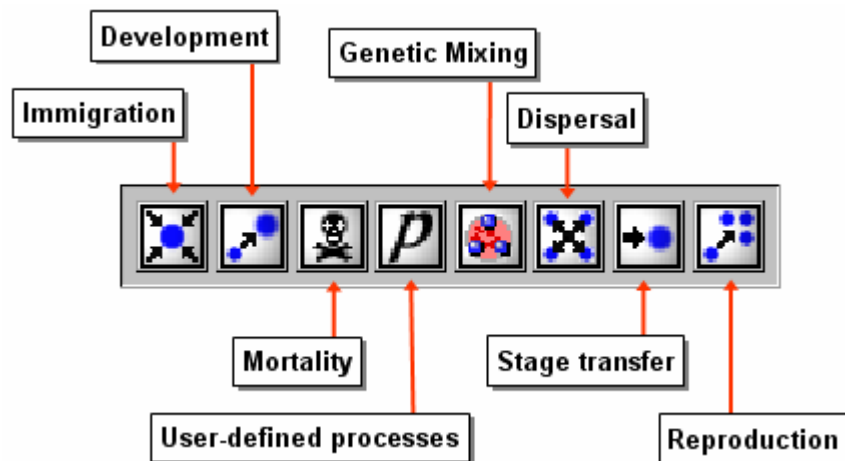
Fig. 6-13 An example of a mortality process.



The precise way in which the process value changes the corresponding Cohort Property can be described when the process is specified.

Each process consists of one or more Factors (termed **Process Components**), which are evaluated separately, and then combined using the **Combination Rule** to arrive at a final value. Process Factors can be either static **Model Parameters** (which do not change their value during a simulation run), or dynamic factors, either **Functions** or other **Processes** (see Functions, Processes and Parameters, page 11).

Fig. 6-14 Process buttons in the Lifecycle module.



What are the various Processes in DYMEX?

The processes used in a particular lifestage in DYMEX can be seen within the grey region of the Lifestage boxes and may include the pre-defined processes: **immigration**, **development**, **mortality**, **dispersal**, **genetic mixing**, **reproduction**, and **stage transfer** and the user-defined cohort variable processes. The **Reproduction** and **Genetic Mixing** processes are only used in reproductive stages. *Note that most of these processes buttons only appear if the corresponding processes are actually used in the lifestage.*

➤ To edit a Lifestage process

- 1.** Left click the process buttons panel in the lifestage graphic (Fig. 6-2).
- 2.** Find the process in the Lifestage Window and add or edit factors to the process as required.



Note that when a process button appears with a red cross on it, that process is required for the lifestage but has not as yet been sufficiently specified.

The components of all processes are set in the same way. Parameters, functions or processes can be used as components of a process and if more than one factor is chosen then a combination method must be selected (*see The Combination Rule, page 57*).



Note that the output (value) of a process is a *rate* in units of /timestep (per day, week or month, for daily, weekly or monthly models, respectively).

6.7.1 Process Rates and the Model Timestep



Equivalent models that differ only in timestep would need their process rates to be different to give the same results. The relationship between the equivalent process rates would vary depending how the processes are applied to change

the associated Cohort Property. Where process rates are applied *directly* (for example, development), the weekly process rate should be 7 times the daily rate. However, where process rates are applied *proportionately* (eg, mortality and stage transfer), the relationship is not so simple. In the example in Fig. 6-13, if the timestep is 1 day, we would end up with a total mortality of .5217 after 7 days.

$$\begin{aligned} \text{WeeklyMortality} &= 1 - (1 - 0.1) \times (1 - 0.1) \times (1 - 0.1) \times (1 - 0.1) \times (1 - 0.1) \times (1 - 0.1) \times (1 - 0.1) \\ &= 0.5217 \end{aligned}$$

Thus, for a weekly timestep model, a value of 0.5217 would need to be used as the mortality rate to get equivalent results for that process. In practice, there will not be such simple relationships between equivalent rates, as other components need to be taken into account. For example, if the mortality is driven by Soil Moisture, that variable will also be a weekly value rather than 7 daily values, and the exact value of the weekly rate function will need to be adjusted to reflect this. Generally, weekly input variables will not show the extremes in values that occur with daily values (they will be averaged out), and process parameters will need adjustment when moving from a weekly to daily model or vice versa, to cater for this. The “take-home” message is that the model timestep is a fundamental model property, and changing model timestep in an existing model requires considerable (non-trivial) adjustments to model parameters to give similar results. In fact, in some cases, similar results may not be achievable.

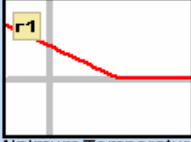

6.8 Modifying Lifestage Processes

Lifestage processes are modified through the **Lifestage Window** (see Section 6.3). As an example, Fig. 6-15 shows the part of the **Lifestage Window** dealing with the **Mortality** processes. Three **Mortality** processes are available in a normal DYME model. These are Establishment, Continuous or Exit mortality (see Section 6.14 for details) and are separated within the process group by a thick light-grey line. If the model uses spatial sub-populations, an additional mortality process (**Post-Mix** mortality) is also available.

Either constants (Parameters), Functions or other Processes can be added to a process. To add a new factor, click on the process name panel (the grey area to the left of the factor graphics) and select “Add Parameter” (Section 6.10.1), “Add Function” (Section 6.10) or “Add Process” (Section 6.9), as required. Existing factors can be edited or deleted by clicking on the corresponding graphic and selecting the “Edit” or “Delete” menu option, respectively.

The factors row shows all the factors that are currently specified for this process. If there are multiple factors, each is labelled sequentially “r1”, “r2”, etc. A **Combination Rule** is set or changed by clicking the right-side panel of the Process Header (or sub-header if related processes are grouped together under the Process Header, as in the mortality processes shown in Fig. 6-15). The Combination Rule will be displayed in that panel if more than one process factor is present (or at any time a user-defined Combination Rule is used).

Fig. 6-15 Part of a Lifestage Window, showing the Mortality processes.

Number {Mortality}	
[Establishment]	
Nymphal one-shot mortality	<p>A proportion of nymphs die from unknown causes.</p> <div> <div>Parameter</div> <div>Unspecified Mortality</div> <div>= 0.15</div> </div>
[Continuous]	<p>$R = 1 - (1-r_1) * (1-r_2) * \dots$</p> <p>Nymphal mortality due to extremes of either cold or heat are specified in this process.</p>
Nymphal Mortality	<div> <div> <div>r1</div>  <div>Minimum Temperature</div> </div> <div> <div>r2</div>  <div>Maximum Temperature</div> </div> <div> <div>r3</div> <div>Variable</div> <div>Chemical A Effect</div> <div>[Direct Function]</div> </div> <div> <div>r4</div> <div>Variable</div> <div>Chemical B Effect</div> <div>[Direct Function]</div> </div> </div>
[Exit]	Not used

The process name and description can be changed by clicking on the Process Name panel and selecting “Change name” or “Add Description” from the resulting popup menu. Once a description is provided, it is displayed in a separate panel below the Combination Rule. The description can then be changed by clicking on that panel and editing it in the dialog that appears.

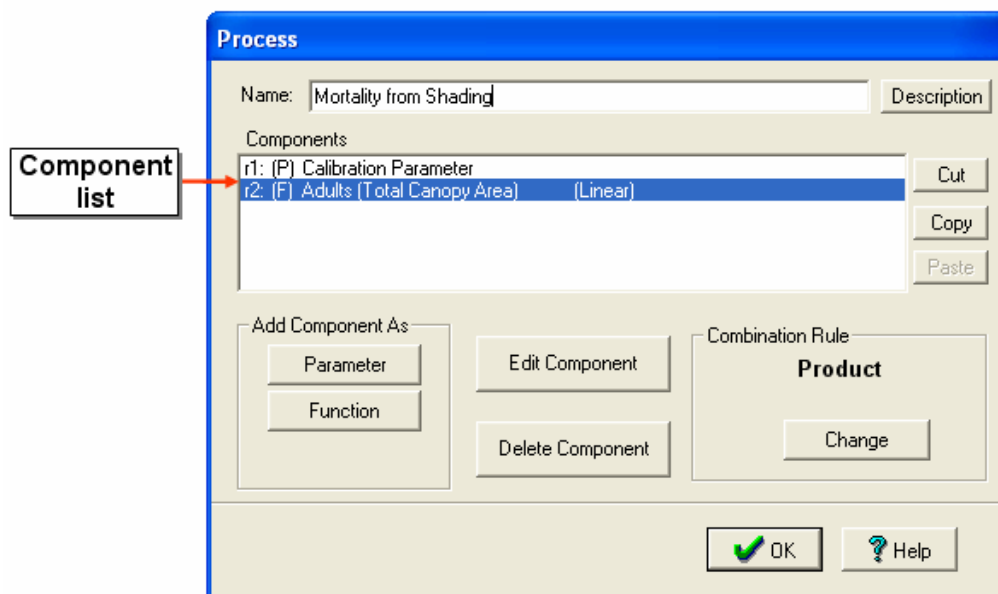
6.9 Process Component Dialog Boxes

The Process Component Dialog (Fig. 6-16) is used to specify the properties of a process factor. This dialog is used for this purpose for any process factor, not just those used within lifecycle modules. The name of the Process can be specified or changed in the edit box at the top of the dialog. If required, a comment that documents the Process may be supplied by clicking on the **Description** button.

Either constants (Parameters), Functions or other Processes can be added to a process. By left-clicking on the **Parameter**, **Function** or **Process** buttons within the **Add Component As** region, another dialog box will appear where the detailed description of the **Parameter**, **Function** or **Process** can be supplied. Note that if a component Process is added or edited, its dialog will not have another Process button – i.e., processes may only be nested to a depth of one level.

The Components box lists the factors that are currently specified for this process. Each factor is labelled “r1:”, “r2:”, etc., in sequence. The label is followed by either (P), (F) or (X), indicating that the factor is a Parameter, Function or Process, respectively, and then its name. If the factor is a Function, its Driving Variable and the Function Shape are given between parentheses at the right.

Fig. 6-16 Process Component Box for an example mortality process with component factors shown in the component list.



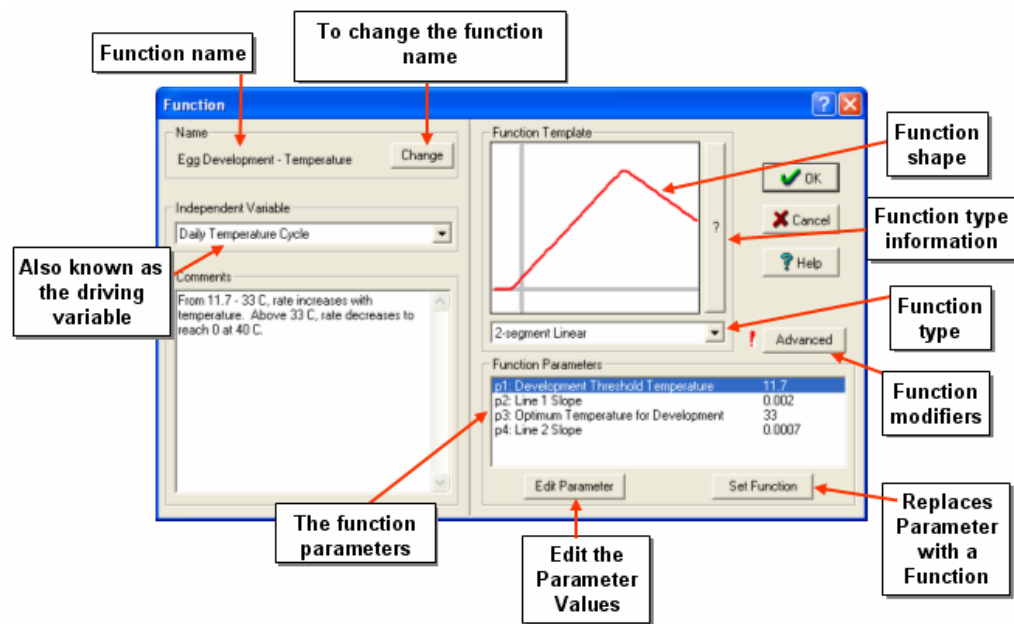
Components can be edited or removed by selecting the **Edit Component** or **Delete Component** buttons respectively, or using the **Cut**, **Copy** and **Paste** buttons. If more than one component is shown within the **Process Components** box, the Combination Rule is displayed in the “Combination Rule” section of the dialog. The **Change** button will also be enabled to allow the rule to be changed if required. See section 6.11, *The Combination Rule*, page 57.

6.10 Function Dialog Boxes

Selecting a **Function** component opens the **Function** dialog box allowing the function type, the driving variable and the parameters of the function to be chosen.

In the example below (Fig. 6-17) a development component (Egg Development - Temperature) is dependent on the variable *Daily Temperature Cycle* and is modelled using a **2-segment Linear** function. To complete the specification of the component, select the function template (ie. **2-segment Linear**) from the function template drop-down list, below the white box illustrating the currently selected template shape (empty when no template is selected). Select the **Independent Variable** (ie. *Daily Temperature Cycle*) from the drop-down list. The **Parameters** box will list the parameters when you choose any template that has parameters. Selecting one of the parameters (by clicking on it), and then clicking on the **Edit Parameter** button opens the **Set Parameter Properties** dialog box, where the parameter details can be specified. Alternatively, the parameter may be replaced by a function. This is achieved by selecting the required parameter and clicking the **Set Function** button (this opens another Function dialog).

Fig. 6-17 Function dialog box with example function.



If required, the selected Function Template may be modified (for example, limited to positive values), by clicking on the **Advanced** button (see Section 6.10.2).



Functions and their parameters should be given descriptive names to help in documenting the model. To change the name of a function, select the **Change** button in the **Name** panel at the top left of the dialog box.



Not all variables displayed in the list of driving variables are suitable for use as a driving variable in a function (e.g. summary variables). See section 2.1, *Variables* for details.

6.10.1 Parameter Properties Dialog Boxes

Set Parameter Properties dialog boxes are used in many places in DYMEX. Two forms of the dialog are used. The first is used for parameters whose values are the same across all subpopulations. The other (**Sub-population Parameter** dialog) is used where the module that owns the parameters takes part in the model's sub-population structure and therefore the parameters may have different values for different sub-populations. Both types of dialogs have a section at the top that shows the type (the "generic" name) and name of the parameter. In the example shown in Fig. 6-18 we are looking at the first parameter of the "2-segment Linear" function (**Line 1 X-intercept**). By default, this is the name given to the parameter, though it is recommended that the name is changed to something more meaningful using the **User Name** edit box.

Fig. 6-18 Parameter Properties dialog box with example values.

Default values and upper & lower limits



All **Set Parameter Properties** dialog boxes contain three boxes labelled: **Lower Limit**, **Upper Limit** and **Default value**. The first two of these allow the user to restrict the values that the selected parameter can take within the **Model Simulator**. In Fig. 6-18 *Threshold Egg Development* is set to a default of 11.7 with a permitted range of 8 to 15. This means that the user can change the parameter to any value between 8 and 15, inclusive, in the **Simulator** (Fig. 6-19). If no upper and lower limits are entered, then there are no restrictions on the parameter value.

Fig. 6-19 Parameter list from the Model Simulator.

If no default is entered and upper and lower limits are set then the initial value of the parameter will be set to the mean of the upper and lower limit. If either an upper or lower limit (or both) is omitted then the default value must be set. Parameters can be restricted to a single value by setting the upper limit, lower limit and default to the same value.

If any of the values are negative, the **Lower Limit** will be the most negative of the values, even if this is of greater magnitude than the **Default value** or **Upper Limit**. This is logically correct, although some people find it counter-intuitive.



Note that it is not necessary to set upper and lower limits.

An example of the **Sub-population Parameter** dialog is shown in Fig. 6-20. It is very similar to the normal parameter dialog discussed above, with the exception of an additional list of parameter values below the name of the parameter. When this dialog is entered for the first time for a particular parameter, only a single item, with the *Sub-population Id* of “(Default)”, is shown in the list box. The values shown for that item will be used for the parameter for all sub-populations. To set different values for different subpopulations, click on the **Add Subpopulation Items** button. The different subpopulations will now be displayed in the first column of the list (the “(Default)” item will also still be present). The minimum, maximum and default values for the subpopulations can then be set by selecting the appropriate row and typing in the desired values in the **Value** boxes. A comment can be provided for each subpopulation parameter.

Fig. 6-20 The Sub-population Parameter dialog box.

Sub-population Id	Minimum	Maximum	Default	Description
(Default)	-	-	-	
SS	0	1	1	The relative effect of the insecticide...
SR	0	1	0.5	The relative effect of the insecticide...
RR	0	1	0.5	The relative effect of the insecticide...

6.10.2 Advanced Function Properties (Function modifiers) Dialog Box

Sometimes one of the user-defined functions almost does what is required. For example, we may have a process that we want to behave like a **Linear above Threshold**, but with the restriction that it must never exceed a value of 1. In this case, the **Advanced Properties** setting may be used to modify the selected function. In this example, a High Limit of 1.0 would be set for the function to achieve the desired result.

The settings in the **Advanced Properties** dialog box modify the result of the selected function, $f(x)$, to yield the process component rate output, y . The final value can be modified by using any of **Y-offset** (y_{off}) and **Scale Factor** (s), or

limited by **High Limit** (y_{max}) and **Low Limit** (y_{min}) (see Table 6-3). The evaluation proceeds as follows:

$$y = \max(y_{min}, \min(y_{max}, y_{off} + sf(x)))$$

Table 6-3 The advanced properties of functions and their definitions.

<i>Modifier</i>	<i>Definition</i>
Y-Offset	Adds this value to the value output from the function (if left blank it is equal to zero).
Scale Factor	The value that is output from the function is multiplied by this value (if left blank is equal to one).
High Limit	This is the maximum value the value output from the function can attain (blank means no upper limit enforced).
Low Limit	This is the minimum value the value output from the function can attain (blank means no lower limit enforced).



Note that some advanced parameters are invalid with some functions. In those cases, the corresponding edit field will be disabled. For example, the **Low Limit**, **Y-Offset** and **Scale Factor** are not available with any of the “above Threshold” functions.

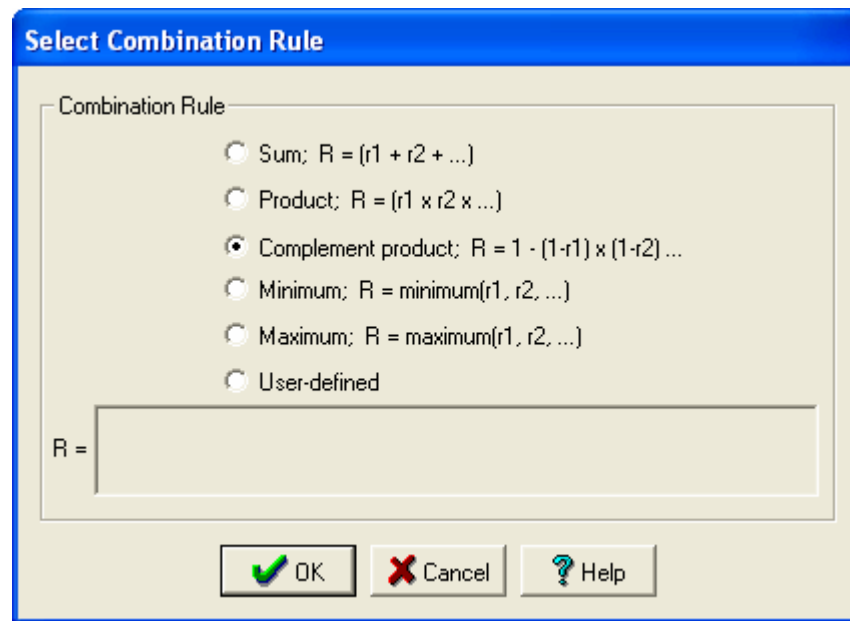
6.11 The Combination Rule

If there are multiple components in a process, the combination rule must be set. The results of evaluating each process factor (f_1, f_2, \dots, f_n) are combined as defined by the combination rule to give a single process rate, r .

The Combination Rules provided for in DYMEX are as follows:

Sum	$r = f_1 + f_2 + \dots + f_n$
Product	$r = f_1 \times f_2 \times \dots \times f_n$
Complement product	$r = 1 - (1 - f_1) \times (1 - f_2) \times \dots \times (1 - f_n)$
Minimum	$r = \text{minimum of } (f_1, f_2, \dots, f_n)$
Maximum	$r = \text{maximum of } (f_1, f_2, \dots, f_n)$
User-defined	$r = f(f_1, f_2, \dots, f_n)$

Fig. 6-21 The combination rule dialog box.



Choice of the appropriate combination rule will depend on what is required for the particular process. For example, let us assume we have an insect (such as a white grub) where adults emerge from the ground after a minimum hardening period, and when a sufficient amount of rain has fallen. We could model this by creating two adult stages (“Adult in Ground” and “Emerged Adult”), with individuals in the former stage moving to the latter via a transfer process containing two factors. The first factor (the “hardening period”) could be driven by *Physiological Age* (using a “Step” function), while the latter could be driven by *Rainfall* (possibly using a “Linear above Threshold” function). Since we need both conditions to be satisfied, the **Product** combination rule would have to be used.



The **Complement product** combination rule is appropriate for mortality processes, since, when combining mortalities from different sources, we are actually doing *survival* (1-mortality) calculations. An example will illustrate this best. Assume we have a mortality process with 3 independent factors, which during a particular timestep evaluate to 0.5, 0.2 and 0. That means that, taken on their own, the first factor would cause a mortality of 0.5, the second of 0.2, while the third would cause no mortality at all.

Combining these using the appropriate Complement product rule would yield a total mortality of 0.6, as below:

$$1 - (1 - 0.5) \times (1 - 0.2) \times (1 - 0) = 0.6$$

If we had used the **Product** combination rule, the total mortality from these factors would be 0 – obviously wrong:

$$0.5 \times 0.2 \times 0 = 0$$

An example of the use of the **Sum** combination rule would be in a user-defined Cohort Property such as Stress. The associated process could sum the contributions to Stress from high and low temperatures.

Combination rules allow for complex logical control of when life processes such as stage transfer occur. The process factors used for logical control are most commonly step or step (general) functions with a step height of 1 to signify the “on” condition and a step height of 0 to signify the off state.

The **Product** combination rule is the same as a logical “and” i.e. both factor A and factor B must be true or “on” in order for the process to evaluate to true or “on”. If either or both of the factors are not “on” then the process is turned off. For example, seed germination requires that both temperatures and soil moisture be within suitable bounds. If either factor is not suitable then germination does not proceed.

The **Maximum** combination rule is similar to a logical “or” i.e. either factor A or factor B or both A and B can be true or “on” and the process evaluates to true or “on”. The **Minimum** combination rule is similar to a logical “nor” i.e. if either factor A or factor B or both A and B are false or “off” then the process evaluates to false or “off”.

A factor can be logically inverted with a user-defined combination rule using the term:

$$1-[x1]$$

where [x1] can have a value in the range 0-1. The same effect can be achieved by using a *Linear below Threshold* function with a threshold of 1 and a slope of -1.

6.12 The Development Process

In DYMEX, development is the process that affects the built-in Cohort Property *Physiological Age*.

Physiological
age



Physiological Age is thus a measure of the state of development (or maturity) of a cohort, with units that are generally stated as a proportion (or percentage) of completed development. For example, seeds of plants such as annual ryegrass undergo a period of dry after-ripening to ensure they don’t germinate during the unfavourable summer season. The Physiological Age of the seeds might be scaled to be 0 when they are first shed, and 1 when the seeds are ready to germinate. Since development in plants is usually dependent on temperature, the rate of accumulation of Physiological Age is generally not constant.

Update
method



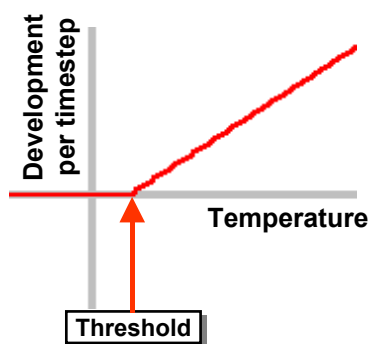
The **Physiological Age** of each cohort is updated, using the development process, once during each timestep. Development is additive (direct). When a process value is obtained from the set of process components, that value is

added to the current value of **Physiological Age**. See Development Update Method page 61.

Note that the accumulated development is not automatically linked to the length of time that an individual spends in a particular stage. The latter is determined in the **Stage Transfer** process. Usually, and especially with insect models, the Stage Transfer process will be driven via *Physiological Age*, which achieves the coupling of development and time spent in the lifestage.

Example development process component Below is an example where a linear above threshold relationship (Fig. 6-22) between rate of development and temperature, has been set for juveniles in this example population (Fig. 6-23).

Fig. 6-22 A DYMEX “Linear Above Threshold” relationship between development and temperature.

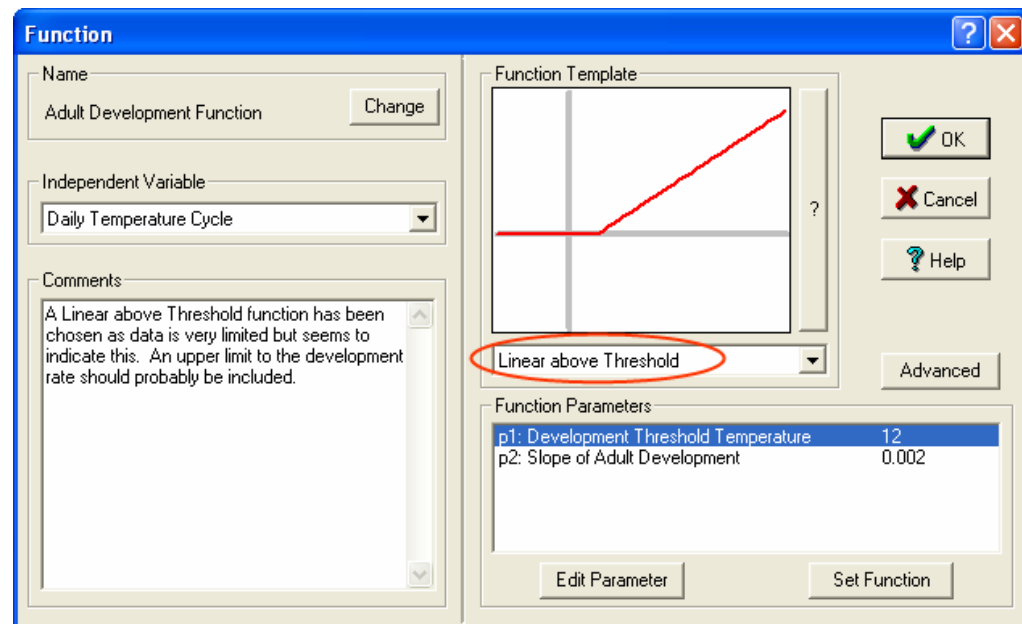


If there is a Linear above Threshold relationship between development and temperature (Fig. 6-22) and the temperature that is used is to drive development is a daily temperature cycle, the changes in *Physiological Age* are equivalent to an accumulation of **degree days**. In fact, if the slope of the Linear above Threshold function is set to 1, then the *Physiological Age* at any time will be the same as the accumulated degree-days above the threshold temperature. In order to implement a degree-day model fully, the stage transition threshold (in the **Stage Transfer** process) needs to be set to the number of degree-days required for full development of that stage. In addition, the lifestage output variable “Development Time” should have its *Physiological Age* set to the required number of degree-days for reporting purposes.

➤ **To complete an example development component**

- 1.** Left click on the **development** button in the lifestage box to open the development dialog box.
- 2.** To add a “Linear above Threshold” relationship left click on the **Function** button within the **Add Component As** box.
- 3.** From the drop down box on the right-hand side of the function dialog box choose the “Linear above Threshold” function.

Fig. 6-23 Development process function dialog box with example *Linear above Threshold* function.



Temperature data must be available for calculation of temperature-dependent development. This is usually obtained using a data file module. Mean temperature can be obtained from maximum and minimum temperature by means of an **Expression** module (page 111) or, as in the case illustrated; a daily cycle is derived by the **Circadian** module (page 123).

Data for the process

4. Select the temperature input variable from the drop down list of possible independent variables (Fig. 6-23).
5. Set the parameters to suitable values using the **Parameters** dialog box (Fig. 6-18).
6. Rename the function giving it an informative name (e.g. “*Development Threshold Temperature*”).
7. Left click on **OK** and when you return to the lifecycle window there should be a green tick on the development process button for the appropriate lifestage.

The estimated timing of completion of development of a particular lifestage tells us when they are likely to appear in field samples and when to apply control measures. However, it does not tell us the abundance of the species, which depends on the reproductive and mortality factors of the population.

Combination rule If more than one process is used to drive development, a combination rule has to be used. The **Product** combination rule is often the most appropriate for development (see *The Combination Rule*, page 57).

6.12.1 Development Update Method



The cohort’s **Physiological Age** is incremented by the value obtained from the combination of the process components in the current timestep. For example,

assume that a development process consists of 3 factors, which evaluate to f_1 , f_2 and f_3 , respectively, and the **Product** combination rule has been selected for combining these factors. If A_t and A_{t-1} are the values of *Physiological Age* in the current and previous timesteps, respectively, then:

$$A_t = A_{t-1} + (f_1 \times f_2 \times f_3)$$

6.12.2 Insect Development

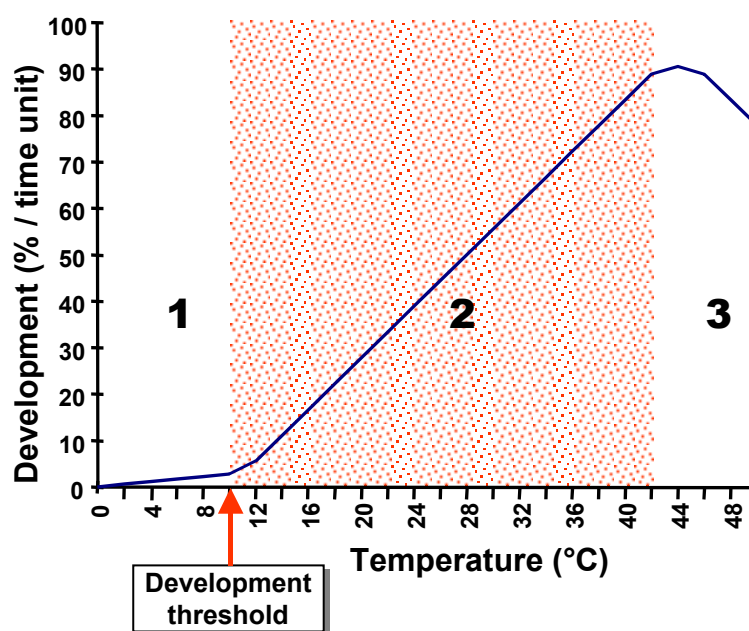


Due to the nature of the cuticle of insects, which makes up the exoskeleton of insects, development is not continuous. As the epicuticle cannot be stretched, insects must moult regularly. The stages between moults are called instars. In some insects, growth is indeterminate (Apterygote) and moulting continues as long as the individual is alive. In the remaining insects (Exopterygote and Endopterygote) growth is determinate and the insects moult to a final instar when growth ceases. The last instar is reproductively active and is called the adult stage.

Factors that influence development

The time spent in each instar, the intermoult period or stage duration, depends on a number of factors, including temperature and diet. In most insects temperature accounts for the bulk (over 90%) of the variation in development time.

Fig. 6-24 An example of a typical insect development function relative to temperature.



Functions to describe development

Many functions have been proposed to describe the development-temperature relationship (e.g. Allsopp *et al.* 1991). In general, the relationship between development and temperature is non-linear. However over a wide range of temperatures the rate of development is effectively linear (Fig. 6-24 stippled region 2). Using a linear relationship, we can identify a temperature threshold

for development below which there is negligible development (Fig. 6-24 *region 1*). Above this development threshold (often termed the base temperature for development), development proceeds at a constant rate. This type of relationship between temperature and development within region 2 of the temperature range is common amongst insect species.

If the temperature experienced by the species being modelled extends beyond region 2 of the temperature range, other functions can be used from the list of function templates available from within DYMEX e.g. Logistic, Pradhan, Stinner, 3-segment linear, etc. to describe development. Alternatively, an appropriate custom-made function template can be created using the user-defined template facility (Section 10.1).

The development functions, together with a daily fluctuating temperature (which can be obtained from the **Circadian** module), can be used to calculate the physiological age accrued during each timestep of the model.

6.12.3 Plant Growth and Development



Plant development is very different from that of most animals. With animals, the fundamental body plan of the adult is laid down in the embryo so all the organs and tissues are present, at least in a rudimentary form. Once an animal has reached maturity, development usually stops. With plants, size, measured as biomass, basal area or leaf area index, is usually more important than physiological development *per se*. Abiotic environmental conditions, herbivory and competition from other plants play an important role in many cases in restricting the maximum size that a plant can attain.

The benchmark relative growth rate of a plant (e.g., $\text{g g}^{-1} \text{ day}^{-1}$) is taken as the rate attained by an isolated plant (i.e., free from competition), under ideal growing conditions. As a plant increases in size, self-shading of leaves within a developing plant canopy, and commensurate increases in the volume to area ratio of the plant reduces the efficiency with which light and other resources are captured relative to the total plant biomass. The change in the relative growth rate of a plant throughout its life has been termed *ontogenetic drift*.

For sessile organisms like most plants, physical space is a resource, as well as an index of available resources “captured” by an individual. If the architecture of the plant does not change appreciably throughout its life, then the area occupied exclusively by it (its *ecological field*) can probably be estimated by an allometric function of its biomass.

$$E_F = Am^b$$

where A is the allometric constant, and b indicates the rate of change in the relationship between mass and area, and theoretically has a value close to $2/3$.

Factors
affecting plant
development

Plant growth is controlled by a complex mixture of factors and is poorly understood in most species. Primarily there are five factors that affect plant development: moisture, temperature, radiation nutrients and daylength. All

may need to be included in a plant population model.

- Nutrients** Plant development is halted if the necessary nutrients are not present, are present at insufficient concentrations, or in combinations that are too unbalanced to absorb. Development can also be inhibited if one or more of these nutrients are in too high a concentration. In cultivated soils the ions of nitrogen, potassium and phosphorous are most commonly present in less than optimal levels, and one or more can be used as a driving variable for plant development functions.
- Moisture** Moisture (soil and atmospheric moisture) has a very important role in plant development. Plants need water as a solvent to allow nutrients to enter the plant and move through its tissues. Water is necessary to maintain turgidity in plants. A special module is available in DYMEX to allow the calculation of soil moisture from rainfall and evaporation (*see The Soil Moisture (1-layer) Module, page 139*).
- Temperature** Temperature is an important factor in plant development. Because of the variation in the relationship between rate of growth and temperature, the relationship can be modelled in a manner similar to that used for insects above.
- Radiation** Radiation in the photosynthetically active range (PAR) is necessary for plant growth. Competition for PAR is an important component of plant competition, with profound effects on plant community composition and vegetation structure. However, for most plant population models, the correlation between temperature and radiation is probably sufficient to use a temperature growth function as a proxy for radiation, and competition for space as a proxy for competition for PAR.
- Daylength** Plants often use daylength and daylength change as means of cueing different development processes in order to avoid unfavourable climatic seasons. For instance, many temperate annual plants use daylength and daylength change to switch from active vegetative growth of rosettes to bolting, when they start absorbing the rosette leaves and translocating the biomass into an upright stem and associated reproductive structures. The daylength cues help these plants avoid seasonal summer drought.

The Development process (and its associated “Physiological Age” cohort property) may be useful for modelling development of plants with an annual life history, but can often be ignored when modelling plant growth in biennials or perennials. Instead, appropriate Cohort Properties such as “Basal Area”, “Biomass” or “Leaf Area Index” can be defined and growth modelled in terms of these. However, the development process could be applied for determining the maturation of reproductive stages such as flowers or ovules, or after-ripening of seeds in the seedbank.

6.13 The Reproduction Process



The reproduction process creates new individuals for the simulation. Reproduction is indicated on the Lifecycle diagram as a broad red line,

originating from the stage that produces the new individuals (the “Reproductive” stage) and terminating at the stage that represents the type of individuals produced (the “Progeny” stage). This directed line is termed a “reproductive link”. In order to make a stage reproductive in DYMEX, it is necessary to create a reproductive link.

A new reproductive link is formed by right-clicking on the stage that will form the Reproductive Stage. This produces a popup menu from which the “Create link” option is selected, after which a link can be chosen from those listed below the “Reproductive Links” label. Note that progeny stages must precede the reproductive stage in the lifestage diagram. If necessary, lifestages can be re-ordered to achieve this. When the new reproductive link is created, empty fecundity and progeny production processes are also created and can be accessed from the Lifestage Window (Fig. 6-25).

Fig. 6-25 Empty Fecundity and Progeny Production processes are created when a reproductive link is established.

Fecundity	
[Establishment]	Not used
[Recharge]	Not used
Progeny Production	

How progeny production is calculated

What is fecundity in DYMEX?

Each cohort in a reproductive stage starts with a potential number of progeny. This potential value can vary depending on the history of the organism. For example, a cohort of flies may have fed better as larvae, and thus become larger adults, with a potential for producing more offspring. The **Establishment Fecundity** process in a DYMEX reproductive stage sets the *maximum* number of progeny capable of being produced per *organism* while it is in that stage. It could also be called *potential fecundity*. When the cohort is created, the Fecundity process initializes a Cohort Variable, the **Residual Fecundity**. The value of this Cohort Variable at any time indicates the remaining reproductive potential of that cohort. If we have more than one reproductive stage in a lifecycle, each will have its own Fecundity process. For iteroparous organisms (for example, a polycarpic perennial plant that flowers each year), it may be useful to be able to “recharge” the **Residual Fecundity** at intervals. This can be done using the “Recharge” Fecundity process component (Fig. 6-25).

At each timestep, a value for Progeny Production (i.e., *the actual number of progeny per organism*) is calculated. If the resulting number is less than or equal to the **Residual Fecundity**, it is used as the current timestep’s progeny production rate, and the **Residual Fecundity** is reduced by the same value. Otherwise, the value of **Residual Fecundity** is used as the progeny production rate and then reduced to zero.

For example, assume that the value of **Residual Fecundity** in a cohort before the calculation of the current timestep’s reproductive rate is F_{t-1} . Also assume that the Progeny Production process consists of 3 factors, which evaluate to f_1 , f_2 and f_3 , respectively, and the **Product** combination rule has been selected for

combining these factors. If R_t is the current rate of progeny production, and F_t is the new value of **Residual Fecundity**, then:

$$R_t = \min(f_1 \times f_2 \times f_3, F_{t-1})$$

$$F_t = F_{t-1} - R_t$$

The number of progeny actually produced will be determined by such factors as the exact form that Progeny Production takes and the longevity of individuals in the stage. The Fecundity processes can be defined by one or more factors, for example accumulated stress, size of organism etc. or it could be a constant. Fecundity components (factors) can be added by clicking on the panels labelled “[Establishment]” or “[Recharge]” and selecting the required “Add” option. Note that normally the Fecundity will be set at the start (establishment) or set using the “recharge” method– it will rarely be appropriate to use both processes within the same lifecycle.

Sex ratio



DYMEX currently does not explicitly account for different sex ratios. There are two means of dealing with this issue. If the reproductive life stage is defined to include both males and females then it will be necessary to set the fecundity value to the potential progeny per female multiplied by the female fraction of the adult population. If half the population consists of males, the actual value of Fecundity should be set to one half of the potential progeny per female. This sex ratio parameter could be made a lifecycle parameter. Alternatively, males and females could be modelled separately for part of the lifecycle using a branching life history scheme. In this case the fecundity value should be set to the potential progeny per female. An explicit sex ratio will be included in a later version of DYMEX.

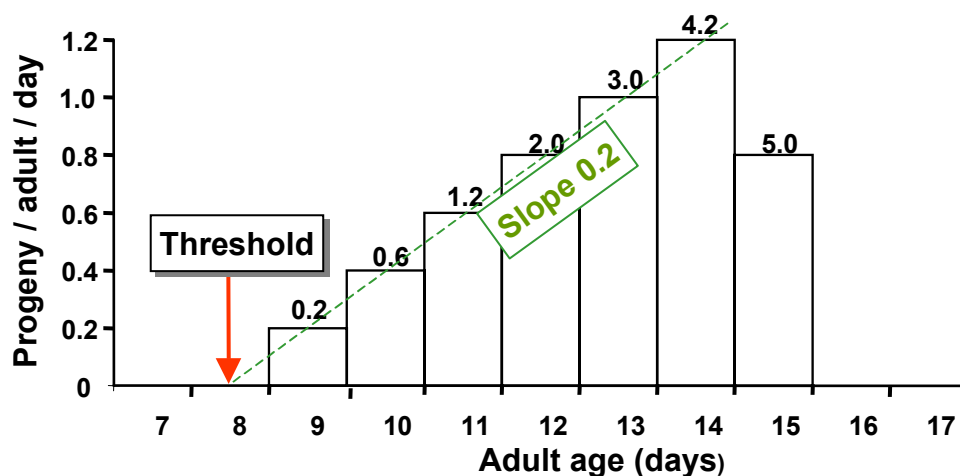


If the potential fecundity is **unknown, very high** or you do not intend to use either the **Fecundity** or **Residual Fecundity** variables in the progeny production processes, then Fecundity can be set to an extremely high number (e.g. 1×10^{10}) so that there is effectively no limit on the total production of progeny.

What is progeny production in DYMEX?

Progeny Production defines the timing of births. No more offspring can be produced than defined by **Fecundity**. For example, in Fig. 6-26 there is a relationship between **Progeny Production** and chronological age of the individual. The **Fecundity** (not recharged) equals 5 and there is a “Linear above Threshold” relationship between chronological age and progeny production with a threshold (in this case the age after which they start reproducing) of 8 days. The slope is 0.2, i.e. after 8 days the number of progeny being produced increase linearly until 15 days when the potential fecundity (5) is reached (Fig. 6-26). Note that the numbers above the bars indicate the cumulative progeny produced by the corresponding age.

Fig. 6-26 An example of progeny production as a function of age. Cumulative numbers of progeny are shown at each age step.



➤ To complete example Fecundity and Progeny Production processes:

1. Create the reproductive link, as described in Section 6.13 above and open the Lifestage Window of the reproductive stage. The **Fecundity** process (and its two components) will be visible near the bottom of the window (Fig. 6-25).
2. If Potential Fecundity is to be set once per cohort, left click the **[Establishment]** panel. Add parameter, function or process components as appropriate. If the Fecundity is to be reset throughout the life of the cohort, click on the **[Recharge]** panel and set the appropriate process factors.
3. Left click the **Progeny Production** panel and choose whether you have a changing progeny production rate (i.e. a function dependent on some driving factor such as time) or a constant progeny production per timestep.

If necessary, set the **Parameters** for any function components. Rename the parameters to meaningful names, e.g. Slope: “Rate of seed production”.

If steps 2 to 3 above have not been fully completed, there will be a red cross across the reproduction process button when you return to the lifecycle window.

6.13.1 Timing of Reproduction



In DYMEX, progeny production always occurs at the end of a timestep. Therefore, the cohort of offspring does not appear until the next timestep. For example, if, in your progeny production process you have offspring being produced on day 10 of your simulation then the cohort of progeny will not appear until day 11. As Progeny Production is a special case of a Stage Transfer process, the same timing as for Stage Transfer applies (Section 6.15.2).

6.13.2 Genetic Mixing

If genetic subpopulations have been defined for the lifecycle, a **Genetic Mixing** process is automatically added to every reproductive lifestage. In the current version of DYMEX, exactly three subpopulations (genotypes) must be defined for this to happen. This corresponds to a single, autosomal locus with two alleles with Mendelian inheritance, with each sub-population representing one genotype (AA, Aa and aa). The genetic mixing process simulates the mechanism of genetic exchange during reproduction and assumes random mating. The genetic mixing process is not user-adjustable.

6.13.3 Insect Reproduction



Most insects are oviparous; that is, the adults lay eggs, although other modes of reproduction, such as larvipary, do occur.

Factors that
determine
egg
production

While age can be important in determining the egg production rate, with higher egg production occurring as the individual gets older, temperature and resource availability are also important driving variables in insect progeny production. The latter is obviously an important factor in phytophagous insects, where often no eggs will be laid if either the host plant is unavailable or is at the wrong stage of development. Insects often lay eggs in batches, with considerable periods of time between each batch. For some situations, this may be modelled best by using several successive reproductive stages, with the Fecundity in each stage referring to the batch size.

Below is an example of a simple model of Fecundity for a fruit fly (Fig. 6-27). In this example potential fecundity is constant, and has been set to half the normal female potential fecundity to account for a sex ratio in the population of 1:1.

Fig. 6-27 An example of constant egg production.

Set Parameter Properties

Type:

User Name:

Value:

Lower limit:

Upper limit:

Default value:

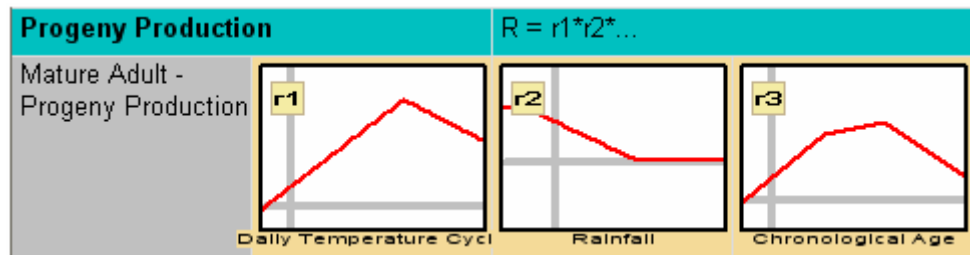
Comments:

Progeny
production

For the same fruit fly example, Progeny Production is dependent on two factors, temperature, rainfall and age. High rainfall reduces egg production

and higher temperatures (up to a threshold) increase egg production. To obtain these relationships, a linear below threshold function driven by rainfall (i.e. egg production increases below a certain daily rainfall e.g. 100 mm) and a 2-segment linear function driven by temperature (i.e. egg production increases above a certain temperature e.g. 20 degrees) are used. A product combination rule is used to combine the components (Fig. 6-28).

Fig. 6-28 An example component list for progeny production.



6.13.4 Plant Reproduction



Plants reproduce in a wonderful variety of ways, including the sexual or asexual production of seeds and spores, the production of clonal embryoids or plantlets (e.g., *Bryophyllum* spp.), modular vegetative reproduction as in grass tillers or rhizomes, and leaf or stem fragments (e.g., *Salix* spp.). This variety of mechanisms is compounded by the diversity of life histories employed by plants (i.e., ephemeral, annual, biennial, facultative biennial, perennial) and the pattern of climatic seasons to which they are adapted. Such marvellous variety makes generalising about modelling plant populations difficult.

The most important population modelling distinction between plants and animals is that plants are mostly *indeterminate* organisms, whilst insects are mostly *determinate*. This means that a plants' fecundity will almost never be satisfactorily simulated by a fixed fecundity value. Most often fecundity will be a function of plant size and/or some form of integration of the growing conditions prior to reproduction. In annual plants this will usually be in the form of the size of the plant around the time of reproduction e.g., at anthesis. In perennial plants fecundity may be a function of the plant size combined with some measure of the net annual growth increment. Depending upon the floral phenological characteristics of the plant, the relevant growth period that influences the annual fecundity may either be instantaneous such as in *Acacia nilotica* where flowers are produced on "new wood", or it may be a function of the net amount of growth attained during the previous growing season. The production of fruits on "old wood" may occur in some tropical trees for instance when the resources for reproduction are stored up during a favourable season, but not applied to reproduction until some time following, when the dispersal and survival of the progeny is most assured.

Temperate
Annual Plants

Many temperate annual plants are adapted to germinating following the first Autumn (Fall) rains, growing rapidly during Autumn, Winter, and early-to-mid Spring, and then entering a reproductive phase in order to produce seeds prior to the Summer drought. In this case, the transition to the reproductive phase is

likely to be a function of daylength, daylength change or temperature. It is often useful to distinguish this phase change with the use of a separate life stage. Fecundity is often a function of the size of the plant at the time of this phase change, and so upon entry to the reproductive stage, the fecundity may be set as a function of size measures such as basal area or aboveground biomass.

Because seeds mature at different rates, progeny production in annual plants tends to spread across time as a function of day degrees following the onset of the reproductive phase. A direct update **Cohort Variable** can be created to accumulate the day degrees above the base temperature. Some form of linear or monotonic increasing function of the day degree cohort variable (e.g., exponential or quadratic) can be combined with a direct function of **Residual Fecundity** using the **Product** combination rule. This will produce a sigmoidal pattern of cumulative seed production between the minimum and maximum day degree thresholds.

Facultative
biennial
plants

Facultative biennial plants such as some thistles are typically found in stressful environments where the minimum size for reproduction may not be reached by all plants in the population due to variation in germination timing (*somatic heterochrony*) or variation in the intensity of plant competition. In this case, cohorts of plants which are too small to successfully reproduce remain in the rosette lifestage, attempting to survive the unfavourable season and resume growth in the following favourable season. When the plants would normally change to the reproductive phase, all second year plants attempt to reproduce, irrespective of their size. We would expect that the plants that over summer would experience negative growth rates due to moisture stress.

To simulate progeny production in facultative biennials it would be necessary combine functions of a plant size measure and Chronological Age with the phenology function as a switch for stage transfer to the reproductive stage, and then proceed as for the annual plant example above. To account for the plants which are too old to persist as rosettes, and too small to produce viable seed, a linear above threshold function of plant size can be used to drive the fecundity process.

6.13.5 Vertebrate Reproduction

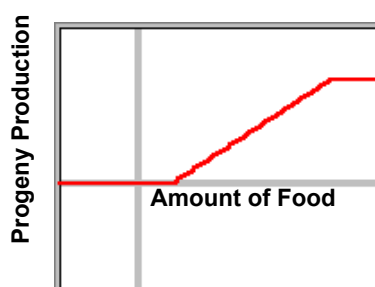


Reproduction is one of the major factors controlling vertebrate population dynamics. Vertebrates in general are not as responsive to the environment as invertebrates or plants.

The maximum number of offspring produced is usually fixed in vertebrates, as potential fecundity tends not to vary with environmental factors as it may in invertebrates.

An important controlling variable for vertebrate progeny production is food availability. The relationship between vertebrate progeny production and food availability is often linear.

Fig. 6-29 An example linear relationship for progeny production for vertebrates.



6.14 The Mortality Process



Mortality in an ecological sense is not a single process, but represents the outcomes of all the factors that may cause the death of an organism. This makes it one of the hardest processes to model, and ecologists have tended to concentrate on subsets of the factors causing mortality such as disease, predation, harvesting, etc. (because many factors often act in combination). Mortality tends to be highly variable in time and space and difficult to predict.

There are three separate mortality processes within each lifestage in DYMEX, termed **Establishment**, **Continuous** and **Exit** mortality. If a dispersal process is used in a particular lifestage, a fourth mortality process (named **Post-mix** mortality) becomes available.

Continuous mortality is applied to cohorts of organisms during every timestep while they are in that lifestage. The **Establishment** mortality is applied just once, immediately after individuals have entered the lifestage (ie, just after cohort formation). **Exit** mortality also is applied just once, immediately before individuals leave the lifestage. **Post-mix** mortality is applied only to individuals that have dispersed from one spatial unit to another, just after (and during the same time step) that the dispersal has taken place. It is of course possible to have all sources of mortality in your model simultaneously.

In any mortality process, the process rate is the proportion of the cohort population (NOT the actual number of individuals) that dies in a particular timestep.



As with all other processes, multiple components can be used to drive mortality. When combining mortality factors, the **Complement Product** combination rule should always be used at the highest level (see *The Combination Rule*, p. 57), though mortality sub-process factors can be combined using other combination rules. For example, the susceptibility of a plant to a herbicide may be related to its age. This may be modelled using a herbicide mortality factor that is a process. This process could consist of a function relating the mortality rate to plant chronological age, and a direct function of a herbicide application event (1=herbicide applied, see *The Event Module*, page 127 for more details). The herbicide mortality sub-process would use a Product combination rule, whilst the combination rule for the different mortality processes (e.g. drought, tillage, competition and herbicide) would be the Compliment-product rule.

Fig. 6-30 The mortality process portion of the Lifestage Window. The Post-mix component will only be present if a dispersal process is used in the lifestage.

Number {Mortality}	
[Establishment]	Not used
[Continuous]	Not used
[Post-Mix]	Not used
[Exit]	Not used

How do I describe a mortality process?

➤ **To create a mortality component**

1. Open the Lifestage Window of the lifestage to which the mortality component is to be added.
2. Select the type of mortality you are adding to the lifestage by left-clicking on the appropriate panel in the column on the far left (Establishment, Continuous, Post-Mix or Exit).
3. Select the Add Parameter, Add Function or Add Process item, as required, from within the popup menu.

If more than one mortality component of a particular type is added, the combination rule must be set by left-clicking on the **Combination Rule** panel (the panel to the right of the panel that shows the mortality type).

Converting lifetable data for use in DYMEX

DYMEX like most models requires the mortality values in terms of a single timestep. If you have the proportion surviving a single lifestage, you have to convert this finite survival rate to an instantaneous rate of mortality.



Here is a quick example of how to calculate an instantaneous rate of mortality, given a total mortality for the stage and assuming the mortality acts continuously and uniformly throughout the duration of the stage. If we have 40% of the lifestage surviving a lifestage that lasts 25 days (i.e., 60% mortality), we first calculate the natural logarithm of the survival.

$$\ln(0.4) = -0.916$$

For a model with a daily timestep, we divide that result by 25, which then gives us the logarithm of the daily survival rate. (For weekly timestep models, the result must be multiplied by 7, giving the logarithm of weekly survival rate.)

$$\frac{-0.916}{25} = -0.03664$$

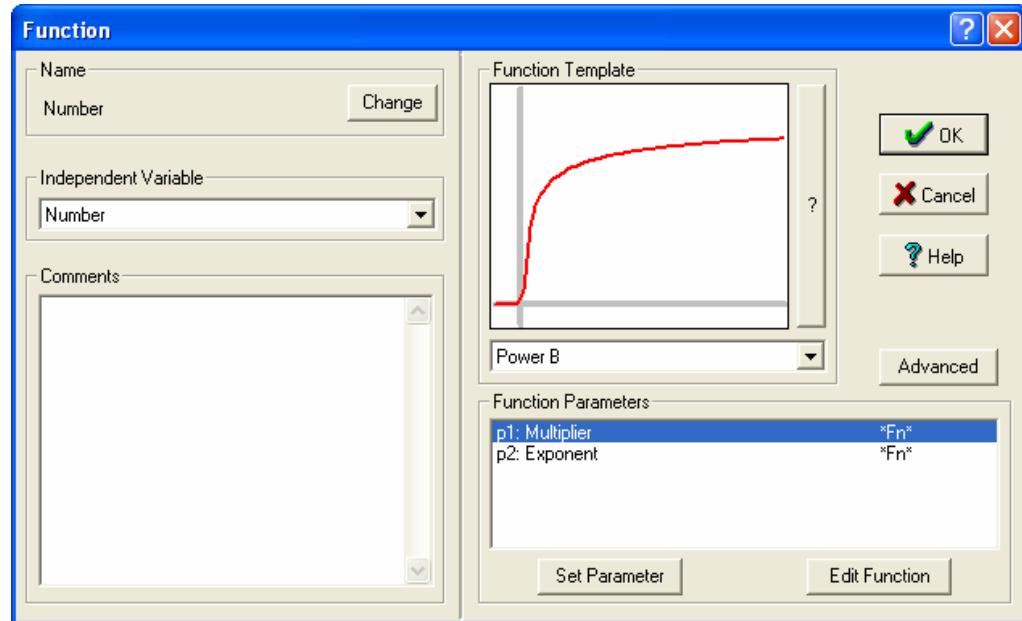
This then has to be converted back to the appropriate mortality rate as follows.

$$1 - e^{-0.03664} = 0.036$$

The final value is used as a constant mortality rate in a daily timestep model.

An example of establishment mortality Establishment mortality is used below to remove larvae at the beginning of the lifestage to simulate a failure to select or attach to a host due to host resistance. This is common in ticks where a large proportion of immature stages die when attaching themselves to their host.

Fig. 6-31 An example of establishment mortality.



In Fig. 6-31 the function used to determine establishment mortality is a modified “Power” function, while the driving variable is the current number of individuals in each cohort. The function produces a mortality rate that increases rapidly with population size, thus producing a strongly density-dependent effect that will have a tendency to limit the size of the population. Note that in the above example, each of the two “parameters” is in turn a function.

Establishment mortality acts upon the cohort upon entry into the lifestage. Using the approach above, it is possible to have a complete failure of a cohort to establish (mortality = 1) if the total number of larvae (current cohort plus previously established cohorts) equals or exceeds the carrying capacity of their host resource.

Continuous mortality effects are more common in population models. Often mortality is observed in a stage, but it is not known what causes the mortality, and it may be necessary to add a constant mortality to the lifestage. Below in Fig. 6-32, 2% of the lifestage population dies per timestep.

Fig. 6-32 An example of a constant continuous mortality.

The dialog box is titled "Set Parameter Properties". It contains the following fields and controls:

- Type:** A dropdown menu set to "Parameter".
- User Name:** A text field containing "Constant Pupal Mortality".
- Value:** A group box containing three sub-fields:
 - Lower limit:** A text field with "0".
 - Upper limit:** A text field with "0.05".
 - Default value:** A text field with "0.02".
- Comments:** A large text area containing the text: "About 2% of pupae are lost through various (mostly unknown) factors each day."
- Buttons:** At the bottom are three buttons: "OK" (with a green checkmark icon), "Cancel" (with a red X icon), and "Help" (with a question mark icon).

Event driven mortality Events are discussed later in the User Guide (*see The Event Module, page 127 for more details*). Spraying herbicide would be such an event. In the example below (Fig. 6-34) the effect of the spray is calculated within the **Event** module as an exponentially decaying effect from the spray date, and output from that module as the variable *Herbicide Spray Effect* (Fig. 6-33). That variable is then used to drive continuous mortality of a weed using the **Direct** function (ie, the variable *Herbicide Spray Effect* is used directly as the mortality value).

Fig. 6-33 An Event module function produces a variable (Herbicide Spray Effect) that is used as driving variable for the function in Fig. 6-34

The dialog box is titled "Function". It contains the following sections and controls:

- Name:** A text field with "Response Function" and a "Change" button.
- Independent Variable:** A dropdown menu set to "Days since Event".
- Comments:** A text area containing the text: "The herbicide starts to decay from its maximum efficacy after about 2 days."
- Function Template:** A graph showing a red line that starts at a high value, remains constant for a short period, and then decays exponentially towards zero. A question mark icon is next to the graph.
- Exponential Decay:** A dropdown menu set to "Exponential Decay".
- Function Parameters:** A table with three rows:

Parameter	Value
p1: Days at Maximum Effect	2
p2: Decay constant	0.2
p3: Maximum Effect	0.95
- Buttons:** On the right side are "OK", "Cancel", and "Help" buttons. At the bottom are "Edit Parameter" and "Set Function" buttons.

Fig. 6-34 The mortality caused by the “event” illustrated in Figure Fig. 6-33.

6.14.1 Mortality Update Method



The mortality process affects the Cohort Property **Number**, which is updated using a *proportional* update method. The new value of **Number** is obtained by removing the proportion of individuals obtained from evaluating the mortality process components. Note that DYMEX automatically limits the mortality rate to values between 0 and 1, inclusive.

For example, assume that a mortality process consists of 3 factors, which evaluate to f_1 , f_2 and f_3 , respectively, and the **Complement Product** combination rule is used to combine these factors. If N_t and N_{t-1} are the values of *Number* in the current and previous timesteps, respectively, then:

$$N_t = N_{t-1} - N_{t-1} \times (1 - (1 - f_1) \times (1 - f_2) \times (1 - f_3))$$

Note that the complement product rule should always be used when combining mortality processes.

6.14.2 Mortality Timing



On the first day that a cohort appears within a lifestage, whether it has arrived from a transfer from the previous lifestage, from immigration or from reproduction, both establishment and continuous mortality will occur. Establishment mortality happens first, when the cohort is created in the lifestage. However, new individuals that arrive in a subpopulation via dispersal do NOT invoke the establishment mortality process.

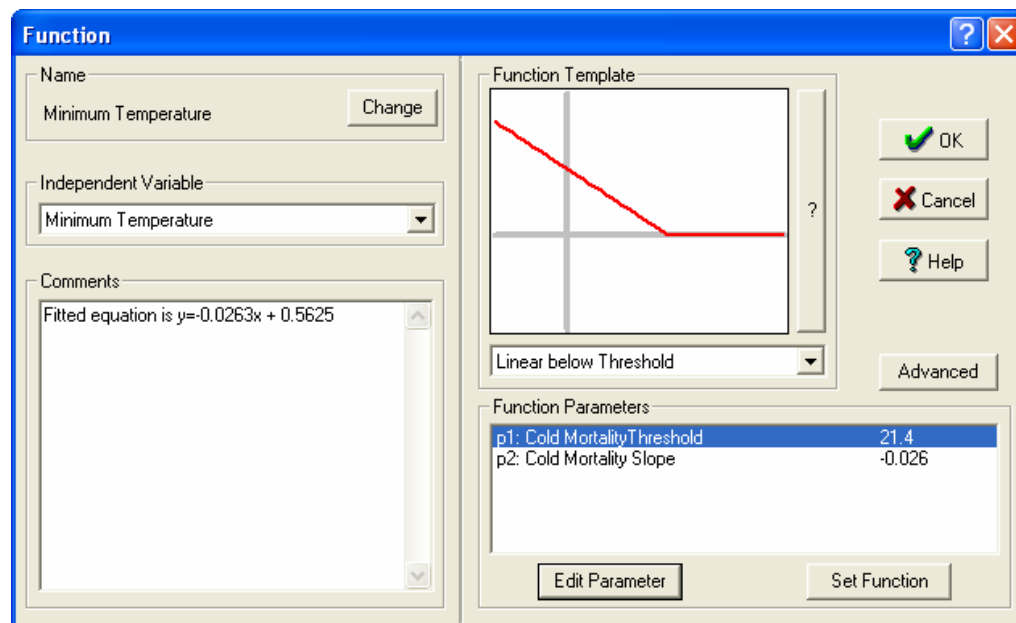
Individuals can leave a cohort via stage transfer (i.e., they move on to another lifestage) or dispersal (they move to another sub-population). In the former case, any exit mortality defined for the originating stage is applied as they transfer to the next stage. In the later case, any post-mix mortality is applied during the dispersal phase.

6.14.3 Insect Mortality



Insect survival is often driven by environmental factors. For example, extremes of temperature can cause increases in deaths. Below is a common example of mortality driven by low temperatures.

Fig. 6-35 An example mortality process driven by low temperature.



In the example above (Fig. 6-35), there is a linear increase in the proportion dying below a particular minimum temperature.

As well as temperature, other factors such as rainfall and predators are very important to the survival of insects.

6.14.4 Plant mortality



The causes and patterns of plant mortality tend to vary throughout its life. Generally, smaller plants are more susceptible than larger plants to inclement climatic effects because they have fewer reserves to draw upon e.g., shallower root systems. Terrestrial plants tend to be susceptible to extremes of soil moisture and temperature.

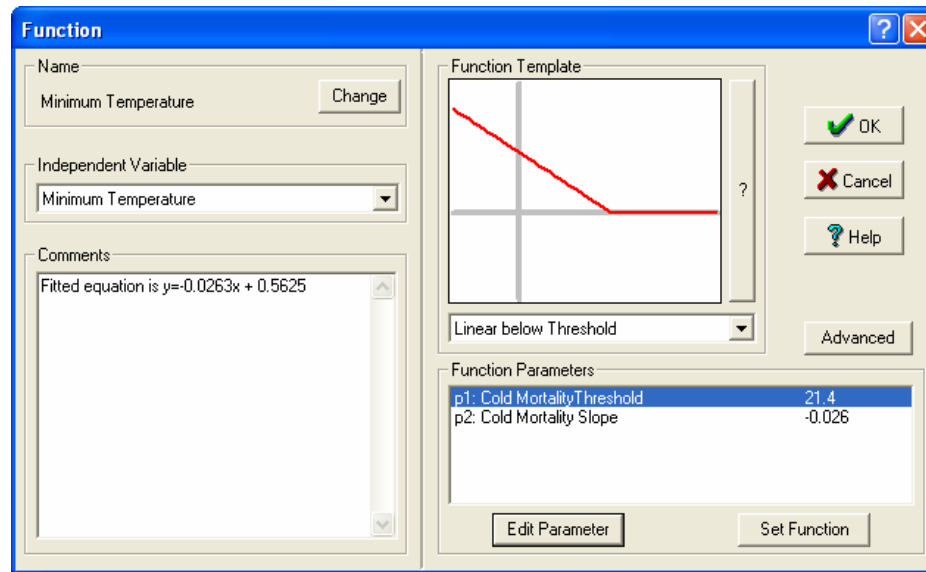
Drought

Drought occurs when soil moisture levels approach or drop below the permanent wilting point for an excessive period. It is important to consider the depth of the soil profile being accessed by plants as they grow. This determines the size of the soil moisture store that they are accessing. Seedlings typically only access a very shallow soil moisture store which makes them particularly vulnerable to warm, dry periods immediately following their germination such as occurs when there is a *false (Autumn) break*. It is possible to include several soil moisture modules in a DYMEEX model, each with a different sized bucket. Different plant life stages can access different soil

moisture stores for growth and mortality processes, depending upon the effective depth of its roots.

Frost	Frosts frequently kill seedlings and immature seeds (endostages) in plants such as wheat. This process could be modelled as a linear-below threshold function of air temperature as was done for insects in Fig. 6-35.
Herbivory	Insect herbivory generally only results in mortality of seeds, seedlings or immature plants. It is unusual for insect attack to result in mortality of mature plants, particularly shrubs and trees. Rare exceptions include the case of a stem-boring weevil, <i>Neodiplogrammus quadrivittatus</i> that destroys vascular tissue in adult plants (Hoffmann and Moran 1992).
Self-thinning	<p>Most plants are sessile organisms. One implication of this is that as they grow, they tend to require more space to persist. The density of seedlings that germinate and establish following a disturbance is frequently much greater than the density of mature plants that can be supported. As the seedlings grow, the total area they need to survive (the sum of their ecological fields) exceeds the unit area available to them. At this size-density limit, further growth of individuals comes at the expense of the mortality of some (usually the smallest) individuals in the population. This mortality process is termed self-thinning to distinguish it from the intentional silvicultural practice of periodically removing trees to avoid excessive plant competition, and maintain maximum growth rates of remaining plants.</p> <p>The so-called $-3/2$ power rule governing self-thinning is thought to be one of the few patterns in ecology that is so reliable that it has been suggested that it is a law. The rule takes its name from the slope of the perimeter boundary line of a graph of $\ln(\text{plant density})$ and $\ln(\text{total plant biomass})$ of populations undergoing self-thinning.</p>
Competition	Competition in plants is usually asymmetric, larger plants tend to outcompete smaller plants. In resource abundant environments, more vigorously growing plants will tend to outgrow less vigorous plants, further reducing the growth rate of the less vigorous plants, and sometimes eventually leading to the death of the less vigorous plant.
Using events to drive mortality	Herbicide and tillage operations are frequently applied in order to kill pest plants. These factors can be set up as event modules and applied to the relevant life stages as direct functions as in Fig. 6-33Fig. 6-34

Fig. 6-36 An example mortality process where the survival of the deer is dependent on pasture biomass: below a threshold pasture biomass there is a non-linear increase in deaths.



6.14.5 Vertebrate Mortality



Vertebrate survival is often dependent on food, for example reindeer introduced to St. Matthew island in the Bering Sea increased to 6000 then in a massive die off that number reduced to 42 as resources became limited (see Fig. 6-36 for an example of a mortality process in a vertebrate lifestage).

Other important factors in vertebrate survival are weather (affects food supplies), disease (could be modelled with an *Event module*), predation (could be modelled with two lifecycle modules: one prey, one predator) and limited places to live and breed.

6.15 The Transfer Process



The transfer process is responsible for moving individuals from a cohort into the next lifestage. It is a continuous process, in that it is applied to each cohort at each timestep (after all the other processes have been applied). The value obtained by evaluating the process is the proportion of individuals in the cohort that are graduating to the next lifestage during that timestep. The value is forced to be within the range of 0 and 1, inclusive, even if the process evaluates to a number outside that range. In all respects, the Transfer Process is a typical DYMEX process, and can be a combination of as many factors as required.



The most suitable combination rule for the transfer process will generally be the **Product**.

➤ **To complete an example transfer component**

1. Open the Lifestage Window of the lifestage to which the transfer component is to be added

The transfer process appears as the last process in the Lifestage Window. If the lifestage has multiple exits, there will be a corresponding number of transfer processes, each labelled with the destination stage.

2. Choose the required transfer process and left-click on the panel to the far left.
3. Select either the Add Parameter, Add Function or Add Process item, as required, from within the resulting popup menu.

If more than one transfer component is added, make sure that the **Combination Rule** is appropriately set.

6.15.1 Transfer Update Method

The process components calculate the proportion of individuals in the cohort that is being transferred to the next lifestage during a timestep. For example, assume that a transfer process consists of 3 factors, which evaluate to f_1 , f_2 and f_3 , respectively, and the **Product** combination rule has been selected to combine these factors. If N is the number of individuals currently in the cohort, the number graduating to the next stage (T), is given by:

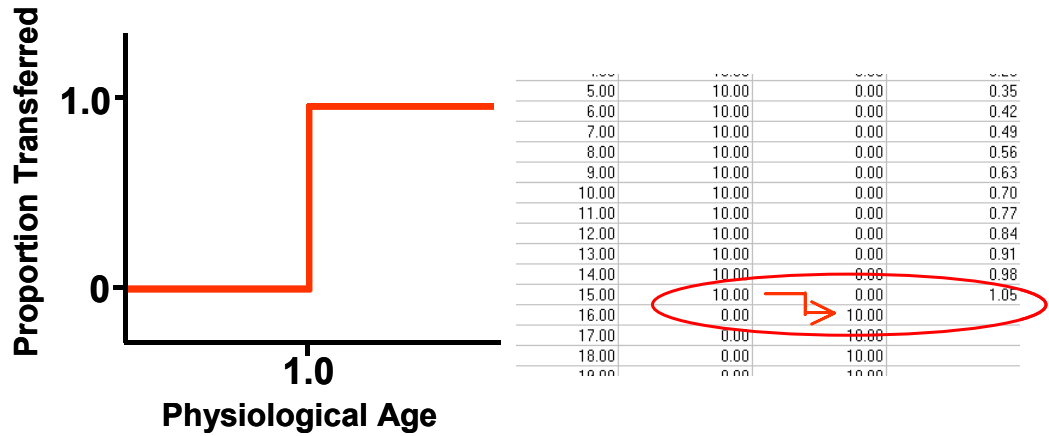
$$T = N \times (f_1 \times f_2 \times f_3)$$

6.15.2 Transfer Timing

Transfer occurs at the end of the timestep, after all processes (including dispersal) have been applied. Each cohort that has a Transfer Process rate greater than 0 moves the graduating individuals to a **Graduate Cohort** (or two Graduate Cohorts in the case where there are two stage exits). Any **Exit Processes** are then applied to each graduate cohort. Note that at this stage the individuals still belong to the source stage as far as lifestage reporting is concerned. At the beginning of the next timestep, all the Graduate Cohorts destined for a particular destination lifestage are combined to form a new cohort of the next stage, to which any **Establishment Processes** are then applied. Some Cohort Variables will have their values transferred to the new cohort (see Section 6.15.4), while others will be reset to the starting value.

For example, consider a case where Physiological Age is driving the transfer and a step function is used with a threshold of 1 and a step of 1.0. If the physiological age of the cohort at the beginning of a particular day was 0.98 and rose to a physiological age of 1.5 by the end of the day, the entire cohort would be transferred at the end of that day (Fig. 6-37).

Fig. 6-37 A transfer function and its effect.



6.15.3 Branching Transfer

If a lifestage has two Stage Links exiting from it (i.e., the stage is the start of a lifecycle branch), each exit must have a Transfer Process specified for it. How many individuals actually transfer (graduate) to each of the next stages on a particular time step will depend on the value of each Transfer Process and are calculated as follows. Assume t_1 and t_2 are the proportion transferring via branch 1 and branch 2, respectively, from a particular cohort as calculated directly from the respective Transfer Processes. Then the proportion of individuals *actually graduating from the cohort* on that timestep (p_t) is

$$p_t = 1 - (1 - t_1) \times (1 - t_2)$$

Then the **actual** proportion of individuals transferring to the next stages via branch 1 (p_1) and branch 2 (p_2) are

$$p_1 = p_t \times \frac{t_1}{(t_1 + t_2)}$$

$$p_2 = p_t \times \frac{t_2}{(t_1 + t_2)}$$

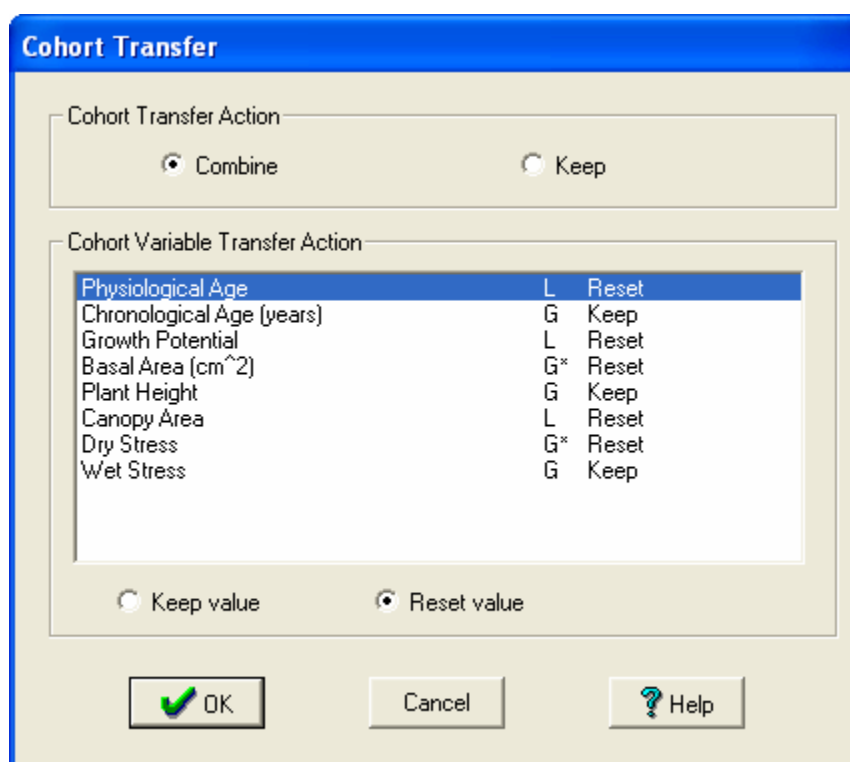
6.15.4 Cohort and Cohort Variable Transfer

The **Cohort Transfer dialog** (Fig. 6-38) can be used to specify how cohorts and Cohort Properties are handled during transfer from one lifestage to the next. The dialog is accessed by left-clicking on the appropriate stage link arrow in the lifecycle diagram and then selecting **Cohort Transfer** from the resulting popup menu.

The default behaviour of DYMEX is to create a single new cohort in a lifestage each timestep from individuals transferring to that stage from another (no matter how many cohorts they originated from). This default can be overridden for any pair of stages in the **Cohort Transfer** dialog, so that a separate cohort is created from those individuals transferring from each source cohort destination stage, without being combined into one. To do this, make sure that the **Keep** option is selected in the **Cohort Transfer Action** panel. This option should be used with caution, as it is likely to create a large number of cohorts and severely slow the running of the model.

As individuals leave one lifestage and transfer to the next stage (as members of cohorts), some of their properties (Cohort Variable values) move with them. Any Cohort Properties whose values are not reset to their starting value during the transfer (which is the default for *Global* properties, see Section 6.4) are dealt with in the following manner: The values of the Cohort Properties in each of the parent cohorts are combined using a weighted average (the number of individuals in each cohort is used as the weight) to give a value for the new cohort.

Fig. 6-38 A Cohort Variable Transfer dialog, showing two Cohort Variables with overridden default transfer handling.



The **Cohort Transfer Dialog** can be used to specify which Cohort Variables are transferred (i.e., the dialog can be used to override the default handling of *Global* and *Local*). The dialog lists all the user-defined Cohort Variables (Fig. 6-38), along with their default handling (**L** for *local*, **G** for *global*), and how they will be handled in this stage transfer. An asterisk following the default-handling symbol indicates that the default handling has been overridden. To change the way that a Cohort Variable is handled during the transfer, highlight the required variable by clicking on it, and then select either the **Keep value** or **Reset value** buttons below the list.

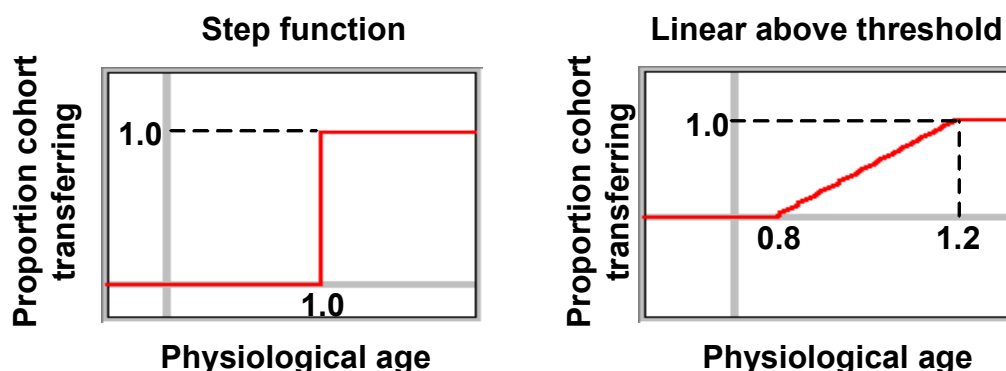
6.15.5 Insect Stage Transfer



With insects, the transfer process can be dependent on a number of factors including temperature, moisture, food availability, etc. The most common situation is one where *Physiological Age* (driven by the development process) causes stage transfer when it reaches a particular value. Sometimes this needs to be combined with a second factor. For example, eggs may hatch when they are fully developed, as long as the relative humidity is high enough.

Two common functions used to link *Physiological Age* to the transfer of individuals to the next lifestage are illustrated below.

Fig. 6-39 Two example functional relationships between physiological age and the proportion transferring.



The first function in Fig. 6-39 is a little unrealistic. The entire cohort is transferring once they have reached full development. The second function has some individuals within the cohort transferring to the next lifestage a little early while some transfer later. *Note that with function 2 (the Linear above Threshold) the function's maximum Y-value is 1.0 (see Advanced Function Properties (Function modifiers) Dialog Box, page 56).*

6.15.6 Plant Stage Transfer



In plants, the stage transfer processes depend largely upon how the lifestages are defined. Generally, germination will depend upon soil moisture and

temperature, though after-ripening or complex dormancy cycling mechanisms may also be important factors. Dormancy cycling can be simulated using a branching lifecycle and using a stage transfer function similar to Fig. 6-39. The dormancy induction or breakage processes can be used to drive the *Physiological Age* process as in Fig. 6-39, or a user-defined cohort variable (section 6.18) could be used.

The seedling lifestage is typically short-lived, but highly vulnerable to mortality processes. Because of its short-duration and the fact that by definition the seedling is largely self-reliant for resources other than moisture and temperature, it is often useful to use a simple function based upon chronological age to transfer seedlings into the next (juvenile plant) stage. Transferring between the juvenile and adult (reproductive) lifestages is frequently a complex combination of daylength, daylength change (positive or negative), and minimum plant size.

6.16 The Immigration Process



The **Immigration Process** is a way of adding individuals into the simulation domain (the other way is via the Lifestage initialisation mechanism in the Simulator). It is a continuous process, in that it is applied to each cohort at each timestep. The value obtained by evaluating the process is the number of individuals of that lifestage that are injected into the simulation during the timestep (i.e., the immigration rate/timestep).

To create or edit the immigration process, select the required lifestage by clicking on it in the Lifecycle Window, and then open its Lifestage Window. The Immigration Process is the first process shown at the top of this window. Left-click on the panel labelled “Immigration” and then select the required process component type from the popup menu.

6.17 Dispersal and related Processes



The **Dispersal Process** moves individuals between different subpopulations. It is a normal, continuous process with the process rate being applied at every time step of the simulation. Because more than one subpopulation is involved in calculating the process rate, it is formulated a little differently from the standard processes that are calculated from properties of just a single subpopulation. Dispersal processes will be present in a particular *Lifecycle* module only if the model defines more than one non-genetic subpopulation and that *Lifecycle* module takes part in the subpopulation structure.

Fig. 6-40 Four subpopulations, showing the possible dispersal paths between them.

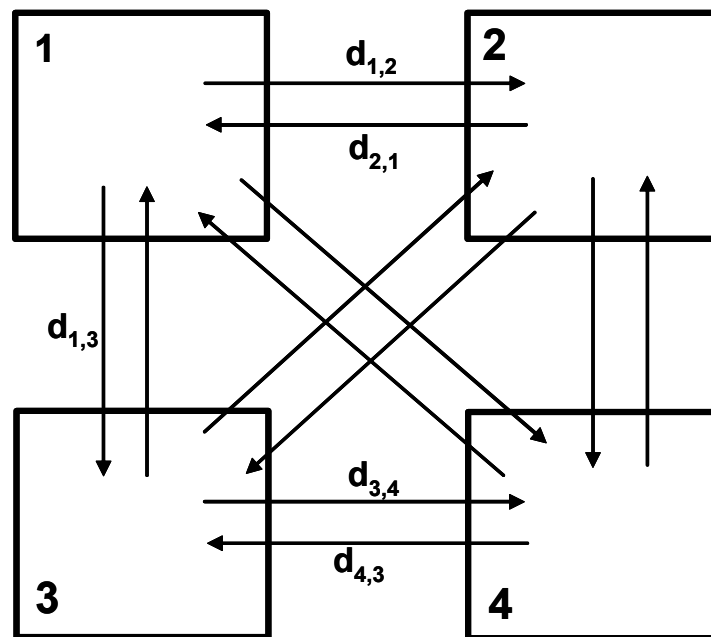


Fig. 6-40 shows a hypothetical model with four subpopulations (cells). The arrows drawn between the subpopulations show all the possible ways that individuals can be transferred between subpopulations via the dispersal process. The dispersal paths are labelled $d_{h,a}$, where h is the label of the subpopulation that the individuals originate from (the ‘*Home*’ subpopulation), and a is the label of the location that they disperse to (the ‘*Away*’ subpopulation). In the case of this four-subpopulation model, there are 12 dispersal paths. In the general case of N subpopulations, there will be $N \times (N-1)$ dispersal paths. During each time step of the model, each dispersal path will have a corresponding **dispersal rate**, which specifies the proportion of individuals in the *Home* cell that move to the *Away* cell. This dispersal rate is determined by the combined action of two processes, the **Dispersal-timing Process** and the **Dispersal Process**. The way that these two processes act together is similar to the way that **Fecundity** and **Progeny Production** combine to produce new individuals. Fecundity provides a pool of “potential individuals” that are available for moving to the progeny stage by the Progeny Production process. In the same way, the Dispersal-timing process specifies how many individuals are ready to disperse, but the Dispersal process is what does the dispersing.

An example may help to make this clear. The buffalo fly is a pest of cattle that spends most of its adult life on or around its host. However, the newly-emerged fly can move over quite long distances in search of a host animal on its first day after emergence. To model this, the Dispersal-timing process rate should be large (perhaps 1) on day 1, becoming close to 0 thereafter, i.e., all the flies are available for dispersal on day 1 after emergence, but do not disperse thereafter. The Dispersal process itself might then be a function of

wind vector and speed, to model the observation that flies disperse more readily down-wind than upwind. Note that the Dispersal-timing process is specified in the same way as any other DYMEX process, depending only on variables within the *home* subpopulation.

Note that DYMEX accounts for all individuals in the simulation. In a model that uses spatial patches with exchange of individuals between the patches, some individuals will be lost from the model domain when they end up in a region of space not included in one of the model patches (emigration). Similarly, individuals will arrive in patches being modelled from outside (immigration). There is currently no special way to account for these losses and gains in DYMEX. One possible technique for simulating this might be to create an extra cell that represents the non-model domain, and thus include it explicitly in the dispersal process. Another may be to simply model emigration as a mortality process, and use the included immigration process to account for individuals arriving from outside the model domain.

6.17.1 The Dispersal-timing Process

The dispersal-timing process indicates when dispersal can take place, and can also be used to set a maximum limit on the proportion of the cohort that is dispersing. For example, in a particular weed, the maturing seeds may remain in a capsule on the plant until the capsule dries out sufficiently to split open. After that, seeds are able to disperse (perhaps when wind strength exceeds a certain threshold). The timing for this splitting of the capsule is best modelled using the dispersal-timing process. In other situations, dispersal may always be possible in a particular stage. For example, bush flies are able to disperse at any time during their adult stage. For this situation, the dispersal-timing process can be set to a constant value.

➤ To create a Dispersal-timing process component

- 1.** Open the Lifestage window for the appropriate stage.
- 2.** In the Lifestage window, find the process (green horizontal bar) labelled **Mix Timing** and left-click on it. This will display a small menu.
- 3.** From the menu, select “**Add Function**”, “**Add Parameter**” or “**Add Process**”, according whether a function, parameter or process is to be added as a factor to the **Mix Timing** process.
- 4.** A dialog for setting the details of the new component will appear. Fill in the details, then click “**Ok**”, and a graphic representing the new component will appear below the **Mix Timing** bar in the Lifestage window.

6.17.2 The Dispersal/Mixing Process



The Dispersal (or Mixing) process moves individuals between subpopulations. The specification of the Dispersal process is more complex than that of other

processes, as it can depend on values of variables for both the originating (*Home*) and destination (*Away*) subpopulations. In DYMEX, all the dispersal rates for a single Dispersal process are specified using a just one equation. The equation is formulated using the same syntax as is used for other user-defined process rate equations, with the following exception. The process components, which for most processes are indicated as $[r1]$, $[r2]$, etc in the process equation, are indicated using $[h1]$, $[h2]$, etc and $[a1]$, $[a2]$, etc, where components prefixed with ‘*h*’ refer to *Home* cell values and those prefixed with ‘*a*’ refer to *Away* cell values. For example, if factor 1 specifies the x-coordinate of a cell, then $[h1]$ is the x-coordinate of the home cell and $[a1]$ is the x-coordinate of the away cell.

Assume we have a particular Dispersal process with 3 components (factors). The first two factors are the longitude (factor 1) and latitude (factor 2) of the centre point of the cells in the simulation. Factor 3 is a simple parameter that is used to scale the dispersal rate. Then a dispersal rate process equation that causes dispersal rates to be inversely proportional to the distance between cells is formulated as follows:

$$[h3] / \text{sqrt}((([h1]-[a1])^2)+(([h2]-[a2])^2))$$

This calculation is used to obtain each dispersal rate, with the appropriate values being substituted for $[a1]$, $[a2]$, $[h1]$, $[h2]$, and $[h3]$. If the model population was divided into 4 subpopulations as in Fig. 6-40, 12 rates would be calculated using this formula at each time step.

Consider the cell numbered “1” in Fig. 6-40. There are 3 dispersal paths leading out of this cell, each with its own dispersal rate. How many individuals actually disperse to each of the other cells on a particular time step will depend on the value of each Dispersal Process and are calculated as follows. Assume $d_{1,2}$, $d_{1,3}$ and $d_{1,4}$ are the proportion dispersing to cells 2, 3 and 4, respectively, from a particular cohort as calculated directly from the Dispersal Process. Then the proportion of individuals *actually dispersing from the cohort* on that timestep ($D_{1,t}$) is

$$D_{1,t} = 1 - (1 - d_{1,2}) \times (1 - d_{1,3}) \times (1 - d_{1,4})$$

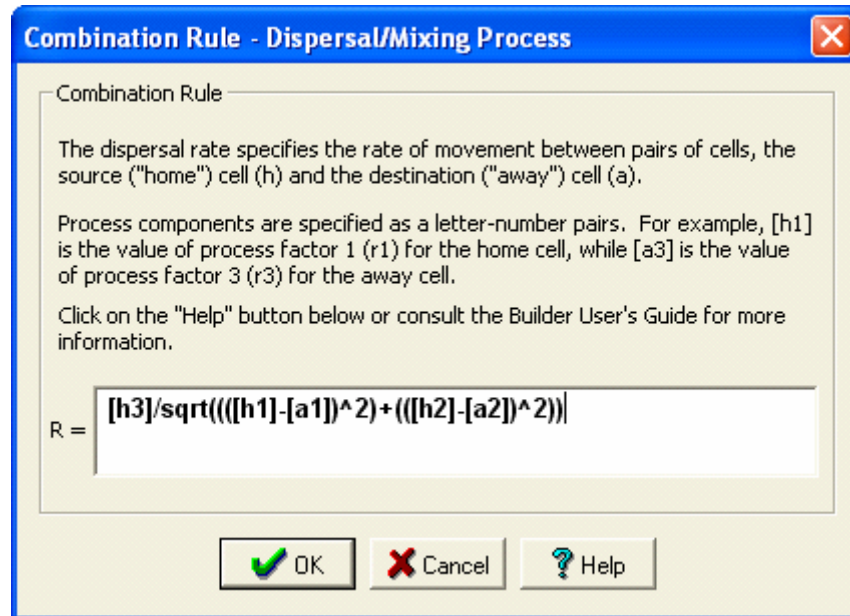
Then the **actual** proportion of individuals dispersing from cell 1 to cell i ($D_{1,i}$) is

$$D_{1,i} = D_{1,t} \times \frac{d_{1,i}}{(d_{1,2} + d_{1,3} + d_{1,4})}$$

The individuals that arrive at the *Away* subpopulation in any one timestep are placed into new cohorts. By default, a new cohort is created for the set of individuals from each dispersal path. It should be immediately apparent that this has the potential to create a large number of cohorts very quickly, causing the simulation to proceed very slowly or even to run out of available memory. Several strategies are available to reduce the number of cohorts that are

generated and these are discussed later in this Section. The method used will, of course, depend on the system being modelled since the purpose of the model is to approximate the system as accurately as required.

Fig. 6-41 The Combination Rule dialog used with the Dispersal/Mixing process.



➤ **To add and configure Dispersal process components**

- 1.** Open the Lifestage window for the appropriate stage.
- 2.** In the Lifestage window, find the process (green horizontal bar) labelled **Dispersal/Mixing** and left-click on its left side. This will display a small menu.
- 3.** From the menu, select **"Add Function"**, **"Add Parameter"** or **"Add Process"**, according whether a function, parameter or process is to be added as a factor to the **Dispersal/Mixing** process.
- 4.** A dialog for setting the details of the new component will appear. Fill in the details, then click **"Ok"**, and a graphic representing the new component will appear below the **Dispersal/Mixing** bar in the Lifestage window.
- 5.** Repeat steps 2 – 4 for each component to be added to the process.
- 6.** Click on right-hand region of the green horizontal bar labelled **Dispersal/Mixing** to open the Combination Rule window.
- 7.** Set the required Combination Rule (see Section 6.11)

The dispersal process will now appear in the Lifestage window as shown in Fig. 6-42.

As was mentioned earlier, dispersal can potentially generate a large number of cohorts, causing simulations to run very slowly or even depleting the available memory. There are several ways to reduce the number of cohorts generated.

Any of the following will restrict the number of cohorts generated, but the method chosen should be a realistic representation of the system being modelled:

- Judicious use of timed dispersal (for example, only when the Chronological Age == 1, Physiological Age. > 0.5, or similar. This method is appropriate when dispersal takes place during a particular phase of an organism's life (for example, buffalo flies tend to do almost all of their dispersal during the first day of their adult life).
- Dispersal in some species may be density dependent (for example, only when *Number* or *Total Number* exceeds some value).
- A limit can be set to the number of dispersal events that individual organisms can undergo. For example, a proportion of organisms may be able to disperse at any time, but once they have dispersed, they stay put for the rest their time in the lifestage. This limitation is set in the *Simulator*.
- Small cohorts can be removed by setting the Cohort Removal conditions appropriately in the *Simulator* (for example, remove cohorts where Cohort Survival < 0.1).
- Cohorts containing a very small numbers of individuals could be removed by setting appropriate dispersal mortality.
- Use the “cohort grouping” mechanism to combine similar new cohorts into a single cohort (see Section 6.17.3 below).

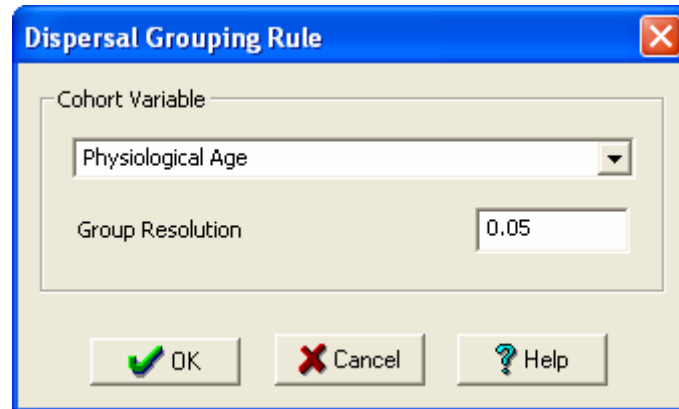
Fig. 6-42 The Dispersal/Mixing process, as shown in the Lifestage window.

Dispersal/Mixing		$R = h3/\text{sqrt}(((h1-a1)^2)+((h2-a2)^2))$	
Dispersal	<div> <div>r1 Variable</div> <div>xpos</div> <div>[Direct Function]</div> </div>	<div> <div>r2 Variable</div> <div>ypos</div> <div>[Direct Function]</div> </div>	<div> <div>r3 Parameter</div> <div>Scaling Factor</div> <div>= 0.1</div> </div>

6.17.3 Dispersal and Cohort Grouping

Cohort Grouping reduces the number of new cohorts created during dispersal by combining “similar” cohorts into a single cohort. The feature is accessed from the Lifecycle Window of a lifestage that uses the Dispersal/Mixing process. Click on the left side (grey) panel of the row that lists the dispersal process components, and from the resulting popup menu, select the **Cohort Grouping** option. This opens the **Dispersal Grouping Rule** dialog (Fig. 6-43).

Fig. 6-43 The Cohort Grouping Rule dialog, showing *Physiological Age* being used to combine dispersal cohorts.



The box at the top shows the Cohort Variable that is to be used to determine the combination of cohorts. Click on the small box with arrow on the right to see the available Cohort Variables, and select from that drop-down list. The Group Resolution specifies a range for that Cohort Variable that will be used to do the grouping. In the example of Fig. 6-43, the Group Resolution is set to 0.05. Cohorts in the “Away” (see Section 6.17) cell after a dispersal event with the same *Chronological Age* and *Physiological Age* ranging from 0 to 0.05 (exclusive) will be combined into a single cohort, with other cohort properties weighted appropriately. Another set of cohorts will be formed by the combination of cohorts with the same *Chronological Age* and *Physiological Age* ranging from 0.005 to 0.1, and so on. This is illustrated by the following table:

Potential Cohort After Dispersal	Chronological Age	Physiological Age	“Away” Cohort created
1	1	0.02	1
2	1	0.04	1
3	1	0.07	2
4	2	0.08	3
5	2	0.09	3
6	2	0.12	4
7	3	0.12	5
8	3	0.13	5
9	3	0.14	5
10	3	0.16	6
11	4	0.18	7
12	4	0.19	7

The initial potential 12 cohorts that would be created without the use of “Dispersal Grouping” are combined into 7 actual cohorts. In practice, much larger reductions in cohort number are possible. There will be some loss of precision as cohorts are combined, but a judicious choice of Group Resolution should reduce this problem to acceptable limits in most cases.

6.18 User-defined Cohort Variable Processes

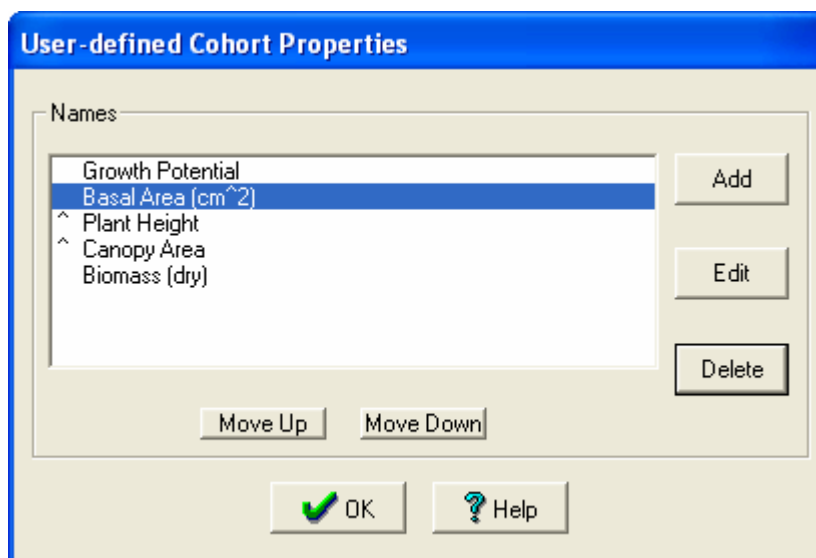


User-defined Cohort Variables are lifecycle variables that are defined by the user. In modelling a plant, for example, its size may be an important factor that determines whether it will flower and how it reacts to herbicide. A Cohort Variable, Size, could be created for the lifecycle to keep track of the size of plants in each cohort. Once the Cohort Variable is created, the associated process is also automatically made available. Up to 32 Cohort Variables may be defined for each lifecycle. Note that Cohort Variables are defined for a whole lifecycle (though often an individual Cohort Variable may be used in only one stage). The **Cohort Properties (local settings)** options available in the **Other Lifestage Properties** dialog (Fig. 6-46) may be used to limit the availability or visibility of a Cohort Variable in one or more lifestages.

➤ To create a user-defined cohort variable

1. Open the lifecycle module by double clicking on the lifecycle name in the **Component Window**.
2. Once you have the lifecycle open go to the **Lifecycle** menu and open the **User-defined Cohort Variable...** menu item. This will open the **User Defined Cohort Properties** dialog box (Fig. 6-44) where user defined cohort variables can be added, deleted or edited.

Fig. 6-44 User-defined Cohort Properties dialog box



3. Left click on the **Add** button. The **Cohort Variables** dialog box contains the features of the user defined cohort variable (Fig. 6-45).
4. Edit the features to suit the use of your cohort variable and close the dialog box.
5. Rearrange the order of the Cohort Variables using the **Move Up** and **Move Down** buttons, if required. Note that the order is only important for *Immediate Update* Cohort Variables.

User defined variable settings Each cohort variable has to have various properties set before it can be used in the model. These properties include **Scope**, **Update Method**, **Permitted Change**, **Range** and **Allowable Operations (Output)**.

Fig. 6-45 An example user-defined Cohort Variable settings.

The screenshot shows the 'Cohort Variable' dialog box for a variable named 'Basal Area (cm^2)'. The dialog is divided into several sections with various settings and checkboxes. Red arrows point from text boxes to specific settings in the dialog:

- 'Basal Area' is incremented directly.** points to the **Update Method** section where **Direct** is selected.
- 'Basal Area' accumulated during the juvenile lifestage is kept until the adult lifestage.** points to the **Scope** section where **Global** is selected.
- 'Basal Area' can only increase.** points to the **Direction of change** section where **Increase only** is selected.
- Cohorts are created with a small 'Basal Area'.** points to the **Range** section where the **Initial Value** is set to 0.01.
- Only interested in total and average 'Basal Area'.** points to the **Output Operations** section where **Total** and **Average** are checked, and **Accumulate** is unchecked.

At the bottom of the dialog are buttons for **OK**, **Cancel**, and **Help**.

Scope, which is either local or global, defines whether the user-defined variable transfers its value between lifestages. A local variable will not transfer its value while a global variable will. *Physiological Age* is an example of a pre-defined variable that is local, the *Physiological Age* is started at 0 in every cohort. *Size* would be an example of a global user defined variable, as the *Size* would be passed on to the next lifestage (ie, a large than average juvenile plant would transfer to the adult stage as a larger than average new adult).

If a variable is specified as Global and the lifecycle is complete, then at least one lifestage must be specified, at the end of which the Cohort Variable is *reset back to its initial value*. If this is not done, the value of that variable is carried through between generations. The reset specification is done via a **Cohort Variable Transfer** dialog, which is accessed from the

Stage Transfer and Reproduction Process dialogs. For example, we may want to accumulate and transfer *Stress* between a number of juvenile lifestages until it is used in the adult lifestage for some purpose. It should then be reset in the Adult stage using the **C.V. Transfer** button in the Adult stage's **Transfer Process** to access the Cohort Variable Transfer dialog (see Section 6.15.4). All accumulated Stress is then lost at the completion of the Adult stage, and the Cohort Variable reset to its starting value. In plants, we will usually be interested in creating a variety of size measures (e.g., aboveground biomass, basal area, leaf area index), and treating them in the same manner as for insect stress in the above example. That is, setting the Global scope, and resetting the variable after the Adult stage. When modelling vertebrates, age may be an important factor to track as it affects annual fecundity and senescent mortality. It would also be specified in the same manner as the previous example.

Setting
starting
values


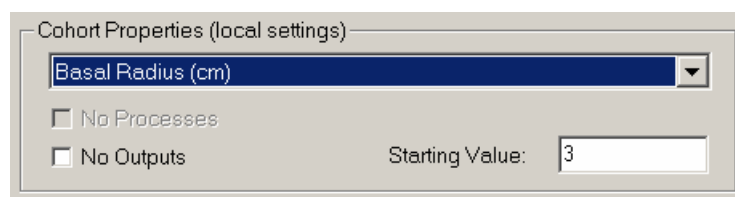
It will often be necessary to specify the starting values of global Cohort Properties in lifestages that will be used to initialise the model. For example, in a plant model a “Plant Height” global Cohort Variable may be used to keep track of plant size. By default, the variable is initialized with its “Initial Value”, as specified in the Cohort variable dialog (Fig. 6-45). However, when the simulation is initialised with adult plants, it is important to make sure that this variable is initialized to the height of an adult plant. This can be achieved by using the “Cohort Property Starting Value” panel (obtained via the “Other lifestage properties” icon () in the Lifestage diagram). Select the required Cohort Property from the drop-down list (Fig. 6-46), and type the desired starting value into the edit box. If the edit box is empty, the Cohort Property “Initial Value” will be used to initialise new cohorts for that stage.

Fig. 6-46 Setting a lifestage-specific Cohort Property Starting Value.



Update Method, in conjunction with the **Inverted** setting, determines how its associated process changes the value of the Cohort Variable. The possible settings are as follows:

- **Direct** update is an incremental update method where the value of the associated process (P) in the any timestep is added to (or subtracted from if the **Inverted** box is checked) the value of the Cohort Variable from the last timestep (V_{t-1}) as in the equation below. Examples where the **direct** update method would be used are *Stress*, where *Stress* is accumulated throughout the lifestage and age (years). *Physiological Age* also uses the direct update method.

$$V_t = V_{t-1} \pm P$$

- **Proportional** update considers the value obtained from the process components in the current timestep (P) to be a proportion. This proportion is multiplied by the cohort variable value in the last timestep (V_{t-1}), with V_{t-1} incremented (or decremented if the **Inverted** box is checked) by the result to obtain the new value of the cohort variable (V_t), as in the equation below. Mortality is an example of an **Inverse proportional** update, where the value obtained from the process components are multiplied by the value of Number in the previous timestep and that value is removed from Number. For example, given a population in the previous timestep of 100, if there is a mortality of 0.8 then there will be only 20 left at the end of the current timestep. Plant growth is frequently specified in terms of relative growth rate (RGR) which uses a proportional update method where $P=RGR-1$.

$$V_t = V_{t-1} \pm (V_{t-1} \times P)$$

- **Current Value** update uses the value obtained from the process (P) in the current timestep as the new value of the Cohort Variable (V). **Current value** cohort variables are clearly always local Cohort Variables.

$$V_t = P_t$$

- **Current Average** updates the Cohort Variable (V_t) with the running average of the process values over the life of the cohort. Again, **Current average** cohort variables are obviously always local variables. Thus, if P_t is the process value for the current timestep, and t is the age of the cohort (in timesteps), then:

$$V_t = \frac{P_t + V_{t-1}}{t}, \quad V_0 = 0$$

The **Use Latest Inputs** option is used to change the order of Cohort Variable updating. Normally, when a Cohort Variable is updated, the functions in the associated process use the previous timestep's value for the value of each "Driving Variable", even if the Driving Variable is another Cohort Variable. In that case, it does not matter in what order they are updated. However, with this option checked, the "most recent" value of the Driving Variables is used. Hence, if there is more than one Cohort Variable with this option checked, it is important to consider their order. Updating of Cohort Variables proceeds in the following sequence: (1) All Cohort Variables with the **Use Latest Inputs** property not set are updated, (2) Cohort Variables with the **Use Latest Inputs** property set are updated in the order that they are listed.

Permitted Change has three options, **increase only**, **increase or decrease**, or **decrease only**, indicating the allowed directions of change for the Cohort

Variable. When the model is run in the *Simulator*, an error will be generated when the associated process produces a change in the Cohort Variable not permitted by this setting. For example, we know that age can only increase, while stress may be able to increase or decrease.

Range restricts the values that the cohort variable can attain. The **Minimum** and **Maximum** set lower and upper bounds, respectively, to the value of the Cohort Variable. The **Initial Value** is the value that the cohort variable is set to when a cohort is created and the Cohort Variable value is not transferred from the preceding stage. This is the case for all cohorts created as a result of lifestage initialization, for all local Cohort Variables, and for global Cohort Variables created in the stage following a Cohort Variable reset. For example, Stress would be have an initial value of zero.

Output Operations. Three operations can be performed on Cohort Properties in a lifestage to generate lifestage output, as indicated in the equations below. **Total** adds the value of the Cohort Variable over all the cohorts in the lifestage. **Average** finds the average value of the Cohort Variable weighted by number of individuals in the cohort. **Accumulate** reports the value of the Cohort Variable at the time when the initial cohort population has halved. Only those output operations that make “sense” for a particular Cohort Variable should be selected. For example, for a Cohort Variable created to keep track of the average height of a cohort of trees the **Total** would not be a useful output. The **Accumulate** output can often be useful in fitting parameter values. Note that by selecting an operation, no lifestage output variables are actually created – it only enables the creation of the corresponding outputs. To actually create the corresponding output variable for a particular lifestage, see Lifestage Output Variables, on page 97. The selected output operations are available in all lifestages by default. If they are not required for a particular stage, they may be removed for that stage by ticking the “No Outputs” checkbox in the Lifestage Properties dialog (Fig. 6-46). However, they may be The following equations detail how each of the outputs are calculated, with V_i being the value of the Cohort Variable in the i -th cohort, and n_i the number of individuals in the i -th Cohort.

$$Total = \sum V_i$$

$$Average = \sum (V_i n_i) / \sum n_i$$

$$Accumulate = V_i, \text{ for } i \text{ when } \frac{n_i}{n_0} \leq 0.5$$

Modifying the
user-defined
cohort
variable
process

User-defined Cohort Variable processes are created and modified in a similar way as the pre-defined processes, by opening the Lifestage Window and adding, deleting or modifying process components listed under the Cohort Variable name. For example, a user-defined Cohort Variable named *Egg Stress*, with 5 continuous-acting components, might show as shown in Fig.

6-47. The panels on the right can then be used to set Establishment, Continuous or Exit processes affecting the selected Cohort Variable. Existing process components are edited or deleted by clicking on them, as described in Section 6.3.1. Note that if no processes are specified for a particular Cohort Variable in a lifestage, then its initial value will stay unchanged throughout the cohort's lifetime.

Fig. 6-47 The process associated with a user-defined Cohort Variable (*Egg Stress*), as shown in the Lifestage window.

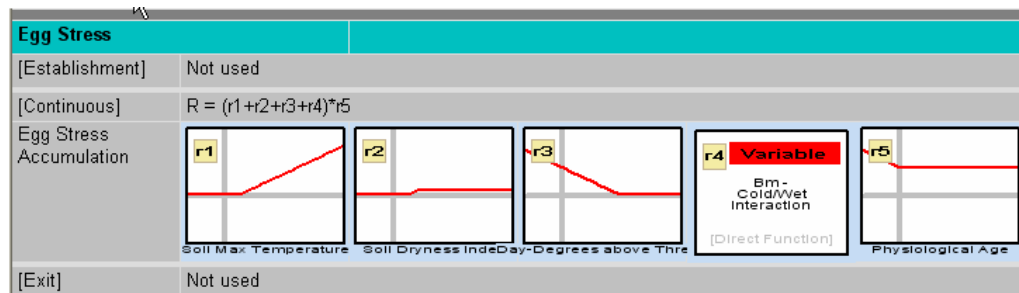
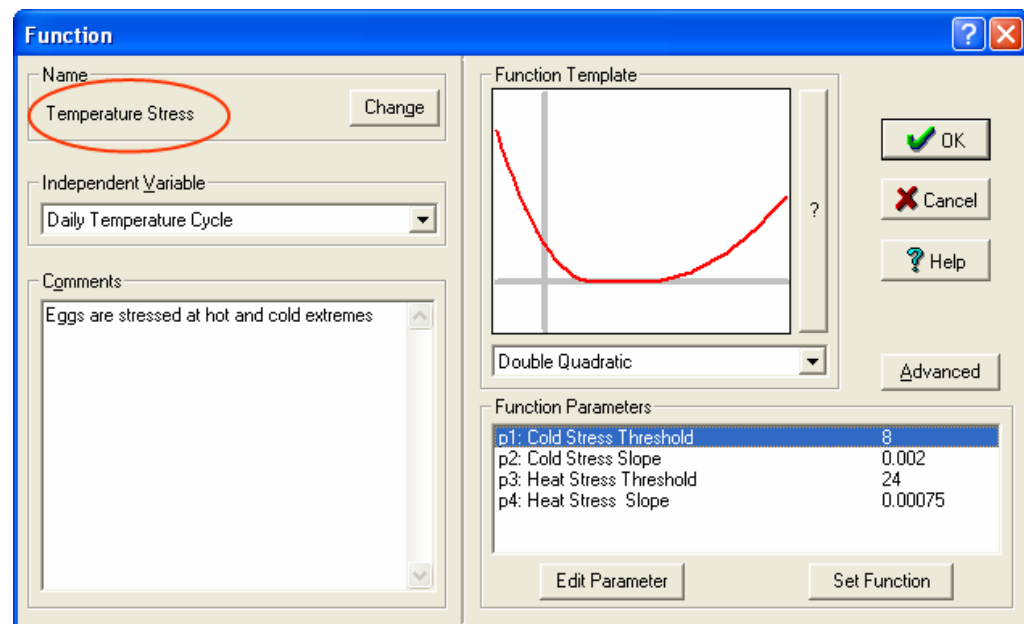



Fig. 6-48 shows an example of a function used to determine the accumulation of a Stress variable.

Fig. 6-48 Example function dialog box for a user defined cohort variable.



In Fig. 6-48 stress is determined by temperature and uses a “Double Quadratic” function to accumulate stress when temperature is at extremes. This stress cohort variable can be used later in the adult lifestage, for example to determine potential fecundity.

6.19 The Resource Variable and Density Calculations

The Resource Variable is specified from the “Other lifestage properties” icon () in the Lifestage diagram (Fig. 6-49). Specifying this variable allows DYMEX to conveniently calculate a measure of density within cohorts or lifestages. It is not necessary to use a Resource Variable for this purpose in most models, however. For example, if we are simulating a population of plants in a model domain (defined as a plot size in hectares), then density (in plants/hectare) could easily be derived from the numbers of plants in a stage and the plot size using (say) an **Equation** module. Using a Resource Variable in this situation has the advantage of needing one less module. There are situations, however, where the density is required in terms of a Cohort Property. An example of a situation where this may be useful is when modelling buffalo flies, where eggs are laid in discrete cattle dung pads as soon as the pads are deposited, and the production of dung pads and eggs are modelled separately, and the dung pads’ volume (and its “decay”) is modelled as a Cohort Property. Competition between larvae takes place in the pad, so a measure of larvae/pad (or per litre) is useful as the density measure. If the dung pad volume is then used as the Resource Variable, the lifestage output **Average Cohort Density** could then be used to obtain information such as the average number of eggs per litre of dung. This information could not be obtained in any other way.

The following measures of density are available when a Resource Variable is specified:

- **“Cohort” Density** – the predefined Cohort Property **Density** (d_i) is calculated as follows for the i -th cohort:

$$d_i = n_i/R_i,$$

where R_i is the value of the **Resource Variable** for the i -th cohort and n_i is the number of individuals in the i -th cohort.

- **Average Density** – this lifestage output variable is the density per individual (i.e., the density experienced by the average individual in the lifestage). This weights the final density towards cohorts with larger numbers of individuals, and is calculated as follows:

$$\text{Average Density} = \sum \left[d_i \times \left(\frac{n_i}{\sum n_i} \right) \right]$$

Thus Average Density is calculated in the same way as all “Average” Cohort Property output variables. Note that even though this output is available if required, it may not be useful in many situations.

- **Average Cohort Density** – this calculates a density that is an average per cohort (as explained in the example on the previous page). If N_C is the total number of cohorts within a lifestage, then

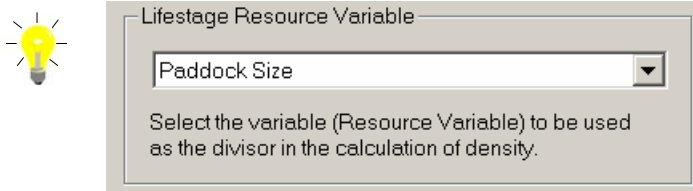
$$\text{Average Cohort Density} = (\sum d_i) / N_C$$

- **Total Density** – this lifestage output would be used in the common situation, where the density “resource” is (say) the area of a farm (and thus $R_i = R$ for all cohorts):

$$\text{Total Density} = (\sum n_i) / R$$

Note that the **Total Density** is closely related to the **Average Cohort Density**, with the former being the latter multiplied by N_C if R is constant.

Fig. 6-49 Setting the Resource Variable for a lifestage.



The **Resource Variable** could be used to determine deaths due to a competition for a resource. That resource could be a population of prey or substrate for laying.



*Note that the **Resource Variable** must never equal zero (as it is used directly as a divisor). If you are using a population as a resource variable add a low value (for example, 0.01) to the population using an expression module.*

6.20 Lifestage Output Variables



The Lifestage Output dialog box (Fig. 6-50), which is accessible from the lifestage output button, lists the outputs available from the selected lifestage. These outputs represent summaries of the values of Cohort Variables at any time, and the available outputs are listed in Table 6-4.

*Note that output variables are calculated from the lifestage at the **END** of the timestep. Therefore, say 100 individuals are born at the beginning of the timestep and there is a mortality process then the population output for that timestep will be less than 100.*



If there are no individuals within a lifestage, all output besides Total Number, Total Stage Mortality, Recruitment via Stage Transfer, Recruitment via Immigration and Total Graduates will be undefined. An undefined value will cause an error if it is used as input to a module.

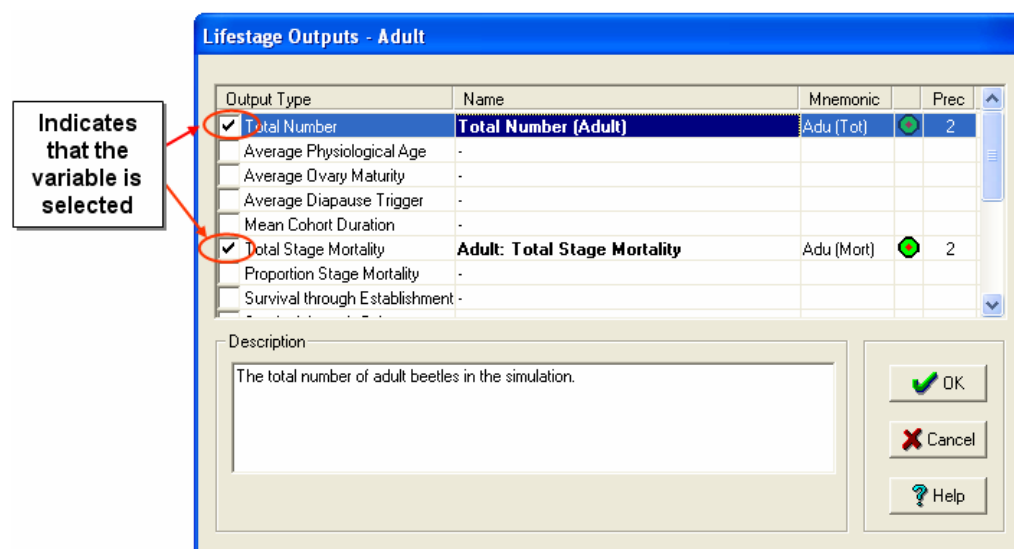


If the lifestage uses sub-populations, all except the “pooled” output variables will have sub-population structure.

Table 6-4 The Output variables and their definitions.

<i>Output variables</i>	<i>Definition</i>
Total Cohort Property (except <i>Number, Density</i>)	The total of the <i>Cohort Property</i> within the lifestage. Total Cohort Property = $\sum (\text{Cohort Property} \times \text{Number in Cohort})$
Total Number	The total number of individuals within the lifestage. Total Cohort Property = $\sum (\text{Number in Cohort})$
Total Density	Total Number per unit Resource Variable. This output is only available if the Resource Variable is <u>not</u> a Cohort Property.
Average Cohort Property	The average of the <i>Cohort Property</i> for individuals within the lifestage. Average Cohort Property = $\frac{\sum (\text{Cohort Property} \times \text{Number in Cohort})}{\text{Total Number in Lifestage}}$
Total Cohort Property (Pooled)	The value of the “Total Cohort Property” output variable, totalled over all sub-populations (available only for lifestages that use sub-populations)
Average Cohort Density	Density of cohorts averaged over the <u>cohorts</u> in the lifestage.
Mean Cohort Duration	The time it takes for 50% of the cohort, which just entered the lifestage, to leave the cohort either by death or transfer.
Development Time	The time taken for the cohort to reach full development (i.e. the preset Physiological Age, usually 1.0).
Total Stage Mortality	The number of individuals in the stage that die during the timestep (from continuous and establishment mortality).
Proportion Stage Mortality	The proportion of individuals in the stage that die during the timestep (from continuous and establishment mortality).
Survival through Establishment	The percentage of individuals entering the stage that survive the establishment mortality process.
Survival through Cohort	The percentage of individuals entering the stage that exit via a transfer process (rather than mortality).
Survival through Stage	The percentage of individuals entering the stage that achieve entry into a succeeding stage. This differs from Survival through Cohort by including the “Exit” mortality.
Recruitment via Stage Transfer	The number of individuals entering the stage via a “Stage Transfer” process during the timestep.
Recruitment via Immigration	The number of individuals entering the stage via the “Immigration” process during the timestep.
Graduates to DestStage	The number of individuals transferring out of the stage via the link to DestStage during the timestep.
Total Graduates	The total number of individuals transferring out of the stage during the timestep.
Progeny Production	The average number of progeny produced <i>per successfully established individual</i> over the lifetime of the cohort created on the corresponding timestep.
Sub-population Proportion	The proportion of individuals in each sub-population (available only for lifestages that use sub-populations)

Fig. 6-50 An example lifestage outputs dialog box.



➤ **To select or unselect Lifestage Output Variables**

1. Open the **Lifestage Outputs Dialog** box (Fig. 6-50), either by clicking on the appropriate lifestage in the **Lifecycle Window** and selecting “Outputs...” from the pop-up menu, or by selecting “Outputs...” from the **Lifestage** menu of an active **Lifestage Window**.
2. To select a variable, click on the small box in front of the appropriate “Output Type”, so that a checkmark is shown in the box. To unselect a selected variable, click on the checkbox to remove the checkmark.
3. When an output is selected, DYMEX provides a suitable name in the “Name” field. This can be changed if required by double-clicking on it and typing the new name into the edit field.
4. A suitable “mnemonic” should be provided by double-clicking on the “Mnemonic” field and typing it into the edit box. The default “precision” for display can also be changed in this way. A description for the output variable can be supplied via the “Description” box.
5. Repeat steps 2 to 4 for each output variable that is required from this lifestage.

Note the variable type indicator shown in the second right-most field of the output variable type list. It indicates whether the output variable has sub-population structure (🎲) or not (🟢).

6.21 Lifecycle Input

If individuals are to be introduced into the population on any day other than the start of the simulation, the **Lifecycle Input** variables must be connected to the

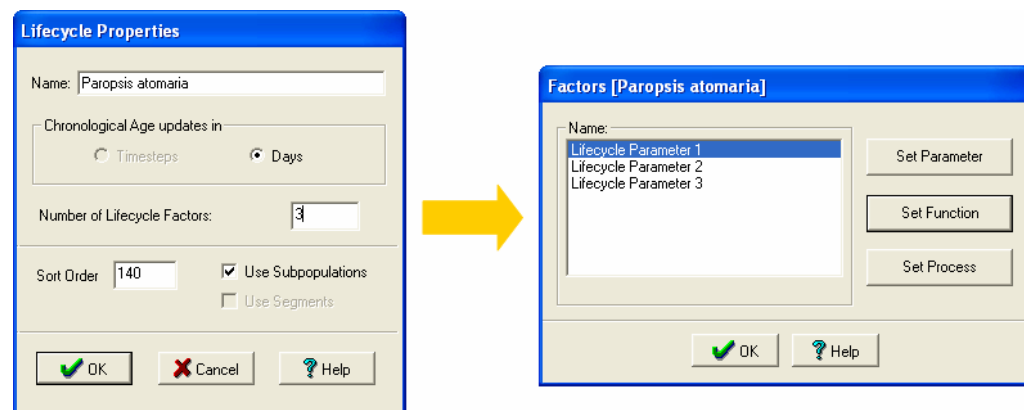
appropriate Timer outputs. The Lifecycle Input dialog is accessed from the Lifecycle menu and either or both of the **Day of Year** and **Simulation Date** inputs must be linked to the appropriate variables from the drop-down list.

6.22 Lifecycle Factors

There are times when a particular parameter may be required in more than one process within one lifestage or even within multiple lifestages in a lifecycle. The parameter could be specified separately in each location, but this is error prone because each time the parameter needs to be changed, each copy has to be changed. A better solution is to define the parameter in a separate **Function** module, and then use it in each process that it is needed via a *Direct* function. This also has disadvantages – it breaks the unity of the **Lifecycle** module by moving a part of it (the parameter definition) outside. The preferred solution is to use a *Lifecycle Factor*, which is a factor that belongs to the lifecycle as a whole rather than to a particular lifestage process.

Lifecycle factors are the same as factors within other modules, such as **Function** module. The number required is defined in the **Lifecycle Properties** dialog (Fig. 6-51, at left), which is accessed from the **Lifecycle Window** by choosing **Properties** from the **Lifecycle** menu. Up to 10 factors may be specified for any lifecycle.

Fig. 6-51 The Lifecycle Properties dialog (at left), showing three Lifecycle Factors specified for the lifecycle. These factors are shown in the Factors dialog at the right.



Once the number of factors has been set, each factor must then be specified as for any other module. This is done from the Factors dialog (Fig. 6-51, at right), accessed from the Lifecycle Window by selecting the “Lifecycle Factors” option from the Lifecycle menu. As is the case with all other module factors, a *Parameter*, *Function* or *Process* may be used for any particular lifecycle factor. Using a lifecycle factor in a lifestage process is as simple as selecting a *Direct* function as a process component and using the desired factor as its independent variable.

6.23 Other Uses of the Lifecycle Module



Note that lifecycle modules can be used to model phenomena that are not actual species lifecycles. One example is the case where the degradation of some specified resource needs to be modelled. This could be achieved by creating a lifecycle where one lifestage represents the resource and another is a degraded resource. Transfer would be the transition of an available resource into a degraded resource. The immigration process could be used to introduce the resource into the lifecycle.

Although the “Event” module (see *The Event Module*, page 127) is designed to cope with most types of management events (including decay of chemical treatment), there may be instances when it is not flexible enough to cope with a complex type of management. The Lifecycle then provides an option for dealing with this situation. Often, a single-stage lifecycle can provide the extra flexibility to deal with such a situation.

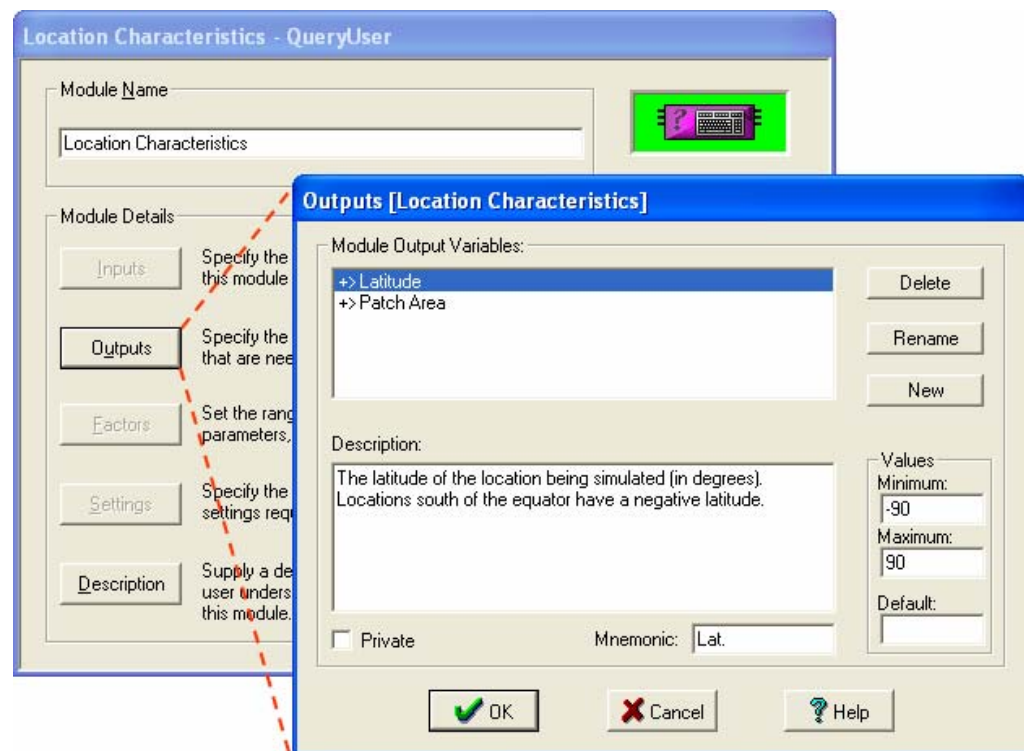
7. Model Input Modules

7.1 The QueryUser Module

What is the query user module?

The **QueryUser** module allows the user to set values at the beginning of a simulation for a number of “output variables” whose values do not change during a simulation (e.g. *Latitude*, *Cost of Pesticide*, etc). These variables are therefore very similar to parameters, but their values are not kept in the parameter file (see *Simulator* Guide). More than one **QueryUser** module is allowed per model, and as many values as required may be supplied in each **QueryUser** module. No input variables are needed for the module.

Fig. 7-1 Query User Module Dialog box with the Output variables dialog box.



Output variables

➤ To add output variables

1. Left click the **Outputs** button in the module dialog.
2. To create an output variable left click on the **New** button in the **Outputs** dialog box. This will create a new output variable for the module with a generic name derived from the module's name.
3. To use the new variable, left click on the **Select** button and you will be able to **Rename** the variable.

The “default” value is the value used for that variable in a simulation if no actual value has been set within the *Simulator*. If the “default” value is left empty the value **must** be set within the *Simulator* otherwise a simulation is not

possible. The “Minimum allowed value” and “Maximum allowed value” sets the Range that the variable can be changed to within the *Simulator* by a user.

Note that if the module has been set to use sub-populations, the output dialog will appear exactly the same in the *Builder*. However, the user will be able to set different values for each sub-population in the *Simulator*.



The **Query User** module could be used to provide the *Latitude* for subsequent use by a *Daylength* module or the module could be used to read in single temperature values for simulations that use constant temperature.

7.2 The QueryUser/Discrete Module

What is the query user module?

The **QueryUser/Discrete** module is very similar to the **QueryUser** module, in that it allows the user to set values at the beginning of a simulation for a number of variables. The difference is that the variables in the case of the **QueryUser/Discrete** modules may take only a defined set of discrete values. Each of these discrete values is labelled with a name. It is, therefore, useful in restricting inputs to sensible values. For example, we may allow a user to choose a herd size of 1, 2, 3, 4 or 5 animals in a cattle parasite model. Here obviously, the fractional values make no sense, and cannot be used. Another example is where we have 3 herbicides that can be used against a weed, and which cause known mortalities of 96%, 92% and 87.5%, respectively. The QueryUser/Discrete module can be used to allow users of the model to pick the herbicides by name and have the corresponding mortality rate automatically used. More than one **QueryUser** module is allowed per model, and any number of output variables may be supplied in each **QueryUser** module, with no restriction on the number of values available for each variable. No input variables are needed for the module.

Output variables are added to the module in the same way as for the UserQuery module (Fig. 7-1). An output variable must have been added before its list of values can be specified.

Fig. 7-2 QueryUser/Discrete Module Settings Dialog box

Variable	Values
Animal Breed	Zebu 0
	75% Zebu, 25% European 1
	* 50% Zebu, 50% European 2
	25% Zebu, 75% European 3
	European 4

➤ **To specify a set of values for an Output Variable**

- 1.** Left click the **Settings** button in the module dialog to obtain the module's Settings dialog (Fig. 7-2).
- 2.** Select the required Output Variable from the drop-down list (obtained by clicking on the small button at the right of the **Variable** box).
- 3.** To add a new valid value-name pair for that variable, click on the Add button and specify the name and its associated value (for example, *Zebu* and *0* as shown in the dialog above). Note that no two names may be the same within each Output Variables set, but values may be duplicated.
- 4.** Set a default value (i.e., the value to be used in the **Simulator** if the model user does not explicitly set a value) by selecting the required default in the **Values** list and clicking the **Default** button. An asterisk in front of the name indicates the currently set default value.

If the “default” value is not specified, the value **must** be set within the **Simulator** otherwise a simulation is not possible. Name-value pairs can be changed or deleted by selecting the required pair in the **Values** list and clicking the **Edit** or **Delete** buttons, respectively.

7.3 The QueryFile Module

What is a query file module?

Within the **Builder**, a **QueryFile** module at first appears the same as the **QueryUser** module. Like that module, its output variables cannot change within a simulation run. The main difference is that a **QueryFile** module can read the value of its variables from files as well as the keyboard. For example, it could be used to read *Latitude* from meteorological data files. If the file were changed between simulations (because the simulation is being run for a different location), a new *Latitude* value would automatically be read in from the new file. Like all data input modules, the **QueryFile** module has to be initialised within the **Simulator**. The **QueryFile** module can be linked to a specific **DataFile** or **Metbase** module when it is initialized within the **Simulator**, and it then reads its variables from whatever file is being accessed by that module.

Output variables

➤ **To create a new output variable for the module**

- 1.** Left click on the “**New**” button in the **Output Variables** dialog.
- 2.** To use the new variable, left click on the **Select** button and you will be able to **Rename** the variable.

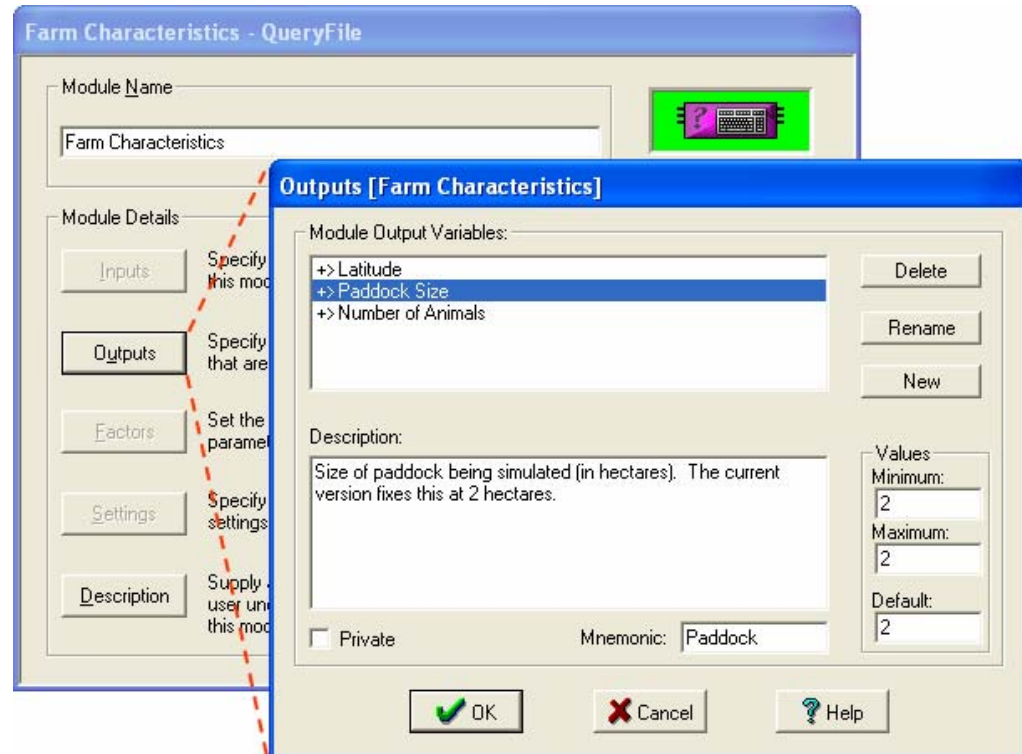
In the **Simulator**, a **QueryFile** module operates just like a **QueryUser** module for those variables whose value is input from the keyboard. For variables whose values are read from a file, however, any default values supplied using the **Output Variables** dialog are ignored, and the positions of the corresponding values in the file need to be specified.

Initialization Initialization is done in the **Model Simulator**.



The **QueryFile** module could be used to easily read in variables that change between locations such as soil pH, latitude, plant biomass and number of plants.

Fig. 7-3 QueryFile Module dialog box and Output Variables dialog box.



7.4 The DataFile Module

What is the data file reader module?

The **DataFile** module allows you to read in a series of values from a file at each timestep during a simulation run. There is no limit to the number of variables that can be read in per module, and more than one **DataFile** module is allowed per model.



The module differs from the **QueryFile** module in that the **DataFile** module reads in a sequence of data for each variable while **QueryFile** only reads in one value per variable for each simulation.

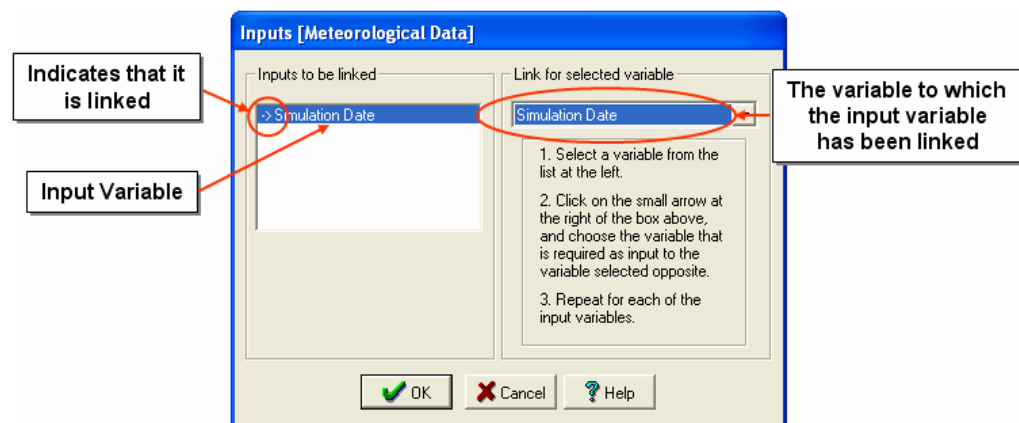
Input variable

The input variable (Simulation Date) can be linked to the *Simulation Date* variable provided by the **Timer** module. If this is done, the sequence of dates in the associated file will be checked during a simulation run. Alternatively, the 'Link for selected variable' can be left as '(none)', in which any date values in the file being read during a simulation will be ignored.

Model Timesteps and Data Timesteps

There is no requirement in DYMEX that the timestep of the model (set in the Timer module) and the timestep of data files must be the same. The **DataFile** module will interpolate or average values as required if these timesteps are different. This is discussed in more detail in the *Simulator* User's Guide.

Fig. 7-4 The Input Dialog from the DataFile Reader Module.



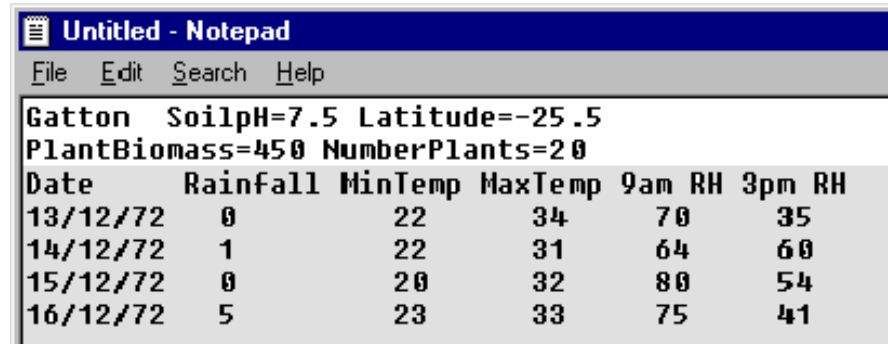
The output variables from this module correspond to the data items to be read from the associated file. These variables are created and named in the **Builder**, but the association with actual columns of data in a particular file is made during module initialization in the **Simulator** (as each run could potentially use a different file). See the section on initialisation of the DataFile module in the appropriate **Simulator** Chapter.

Initialization

➤ **To add and set up a DataFile module**

- 1.** Add the **DataFile** module by selecting the **Model/Add Module...** menu item and choosing **DataFile**.
- 2.** Double click on the new **DataFile** module in the Component Window.
- 3.** If the module is to be linked to the **Timer**'s "Simulation Date" variable, left click on the **Inputs** button and select **Simulation Date** from the drop-down box at the right of the dialog.
- 4.** Go back to the **DataFile** module dialog box and left click on the **Outputs** button.
- 5.** Left click on the **New** button to add the variables needed. Give the variables appropriate names by using the **Rename** button. If needed, also specify the maximum and minimum values that each variable can take. These values are used by the **Simulator** to check the file for data errors. In addition, the variables can be described in the Comment field – this description can aid the user of the model in correctly setting up the data file.

Fig. -7-5 Example data that could be read from a file with the data file reader (highlighted grey).

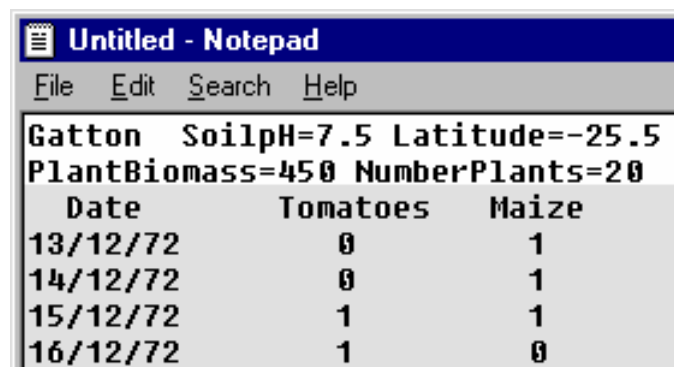


Untitled - Notepad						
File Edit Search Help						
Gatton SoilpH=7.5 Latitude=-25.5						
PlantBiomass=450 NumberPlants=20						
Date	Rainfall	MinTemp	MaxTemp	9am RH	3pm RH	
13/12/72	0	22	34	70	35	
14/12/72	1	22	31	64	60	
15/12/72	0	20	32	80	54	
16/12/72	5	23	33	75	41	



The **Data File Reader** module could be used to read in host availability for a pest insect. Within a text file, the presence of different hosts could be listed at different times of the year. For example (Fig. 7-6), two hosts exist for the pest and they appear at different times of the year: the text file could consist of 2 columns of values with 1s in the columns when the host is present and 0s when it isn't.

Fig. 7-6 Example file with host presence /absence data.



Untitled - Notepad		
File Edit Search Help		
Gatton SoilpH=7.5 Latitude=-25.5		
PlantBiomass=450 NumberPlants=20		
Date	Tomatoes	Maize
13/12/72	0	1
14/12/72	0	1
15/12/72	1	1
16/12/72	1	0

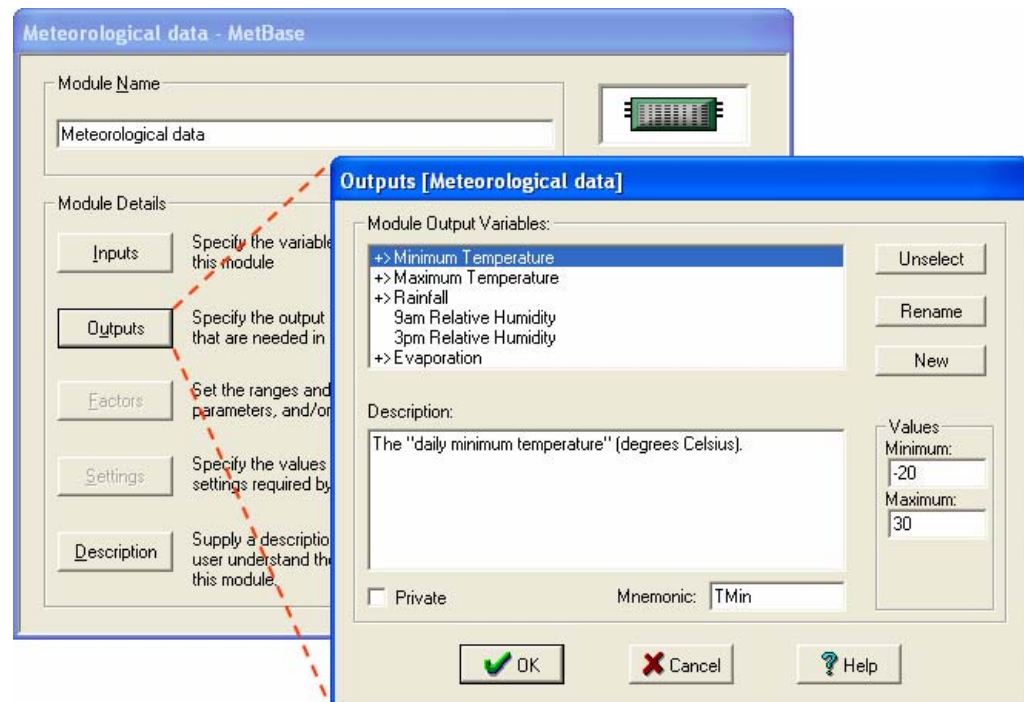
7.5 The Meteorological Data File Reader Module (Methase)

What is the meteorological data file reader module? The Meteorological Data File Reader module (**MetBase**) is a specialised **DataFile** module. It has six pre-defined output variables, which are Minimum Temperature, Maximum Temperature, Rainfall, 9am Relative Humidity, 3pm Relative Humidity and Evaporation (in that order). Additional variables can be added if required.

Input variables The input variable, Simulation Date, and its usage is the same as for the **DataFile** module described in the previous section.

Output variables The six predefined output variables are listed in the Output Variables dialog box and can be selected and renamed if required. Maximum and minimum allowed values, and a description of each variable can be supplied in the same way as for the **DataFile** module.

Fig. 7-7 Metbase Output Variables Dialog box.



There are two main reasons why MetBase should be used rather than DataFile for reading meteorological data. Firstly, the MetBase module is aware that rainfall and evaporation are recorded in the file as a total for the corresponding data timestep while the other preset variables are recorded as averages. If interpolation or aggregation of data needs to be done because the model and data timesteps are different, this is handled correctly without the need to set this up in the *Simulator* first. Secondly, the MetBase module does a look-ahead read of the next day's minimum temperature. This means that if this minimum temperature variable is used as input to a **Circadian** module (Section 9.1), that module can use the appropriate minimum temperatures for the two extremes of each day's cycle.

➤ **To add and set up a MetBase module**

- 1.** Proceed through steps 1 to 4 described for setting up the DataFile module.
- 2.** From the predefined variables, select those that are required by clicking on each in them in the list, and then clicking on the **Select** button. If required, rename the variable by clicking on the **Rename** button. The permissible range of each output variable can be set by entering values into the **Maximum** and **Minimum** text boxes, and a description can be provided in the **Description** box.
- 3.** If extra variables need to be read from file containing the meteorological data (for example, *Radiation*), click on the **New** button to add a new variable. The steps for selecting, renaming and specifying limits for this variable are the same as for the pre-defined variables.

Refer to the description of the DataFile module in the previous section for more information about usage of this module.

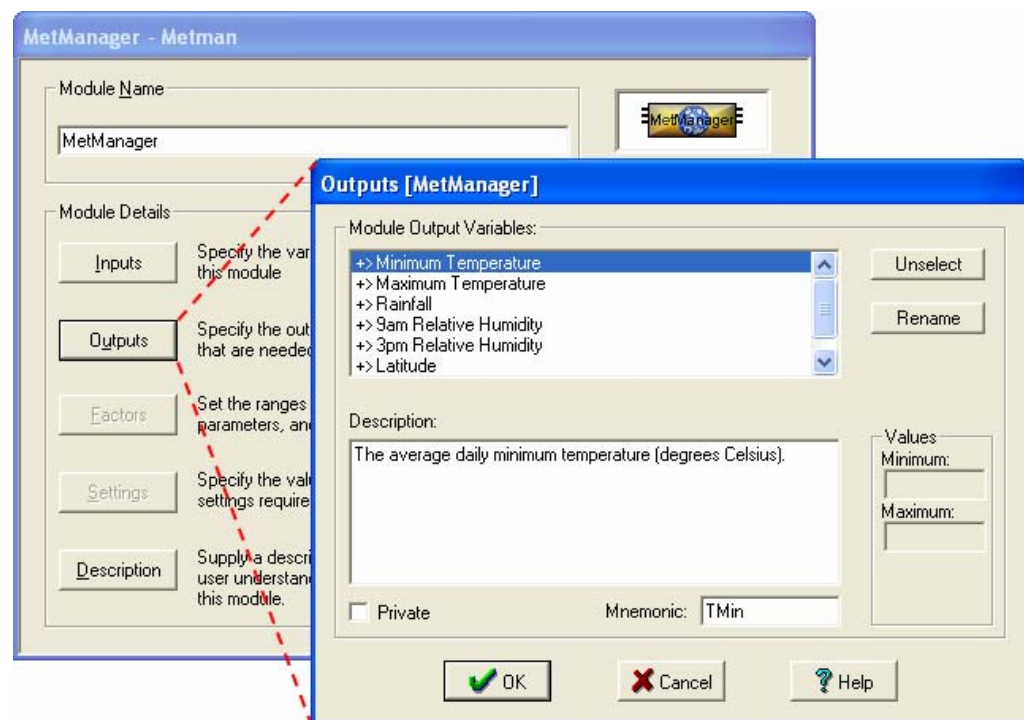


The **Metbase** module could be used to read in rainfall data to estimate survival of various lifestages. To do this, select “Rainfall” as an output variable within the **Outputs** dialog box. Go to the lifecycle and the lifestage that the mortality process linked to rainfall is to be added to, open the **Mortality** process, select continuous mortality and add a function component. Go to the **Independent Variable** drop down box and select “Rainfall”. Choose the shape of the function and set the parameters.

7.6 The Climate Database (MetManager) Module

The MetManager module is used to obtain long-term average meteorological data from the MetManager database. The database stores monthly or weekly values of various climatic variables, including maximum and minimum daily temperatures, rainfall and 9am and 3pm relative humidity. These data can be accessed through the MetManager module, which automatically converts the data into the appropriate timestep for use by the model (using simple linear interpolation). A common use of this module is to obtain climatic variables for many locations across a region for use in “equilibrium” type simulations. The usual caveats that apply to the use of long-term average (smoothed) data in situations where a model’s parameters have been fitted using actual met data must be kept in mind.

Fig. 7-8 MetManager Output Variables Dialog box.



The **MetManager** module uses a single input variable, *Day of Year*. This variable is generally derived from the corresponding **Timer** module output.

The following output variables are available from the MetManager module.

Output variables	Average Daily Minimum Temperature (degrees Celsius)
	Average Daily Maximum Temperature (degrees Celsius)
	Rainfall (mm)
	Average 9am Relative Humidity (%)
	Average 3pm Relative Humidity (%)
	Latitude (degrees)
	Longitude (degrees)

Note that all of these variables except 3pm relative humidity are available for each location in the database. For those locations that do not have 3pm humidity data, it is calculated from the 9am relative humidity using the formula,

$$3pm\ RH = 9am\ RH \times 0.85$$

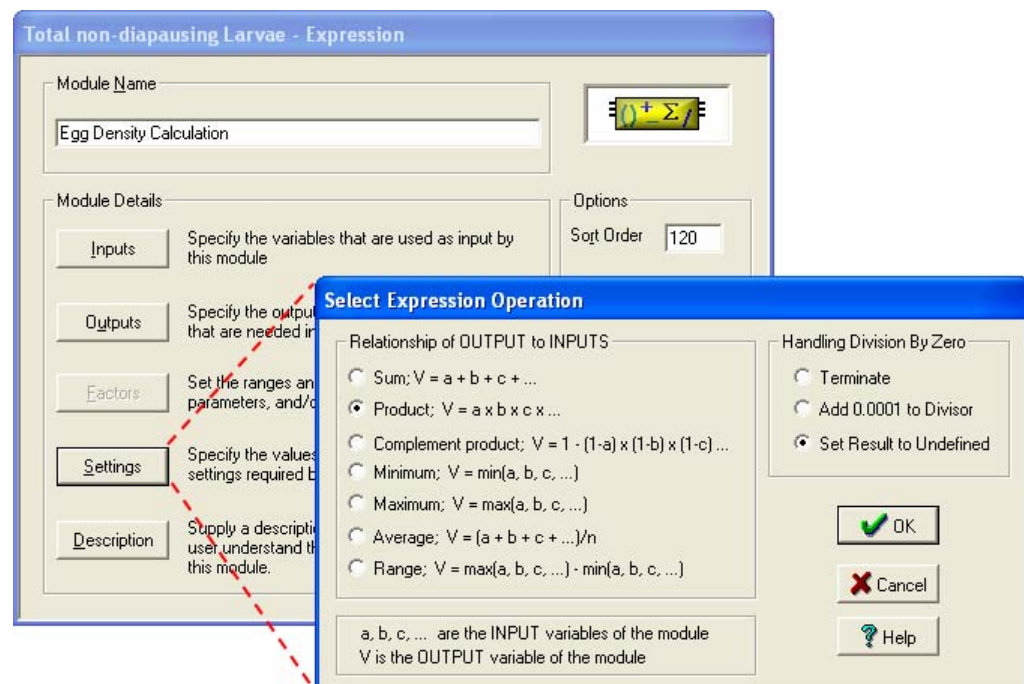
8. Data Manipulation Modules

8.1 The Expression Module

What is an expression module?

The **Expression** module is used to combine values of different input variables in some way to produce a single output variable. The input variables can come from any module, e.g. **Metbase**, **Lifecycle**, etc. As many input variables as required can be used. More than one **Expression** module is allowed per model. Expression modules are able to use sub-populations.

Fig. 8-1 The Expression Module's input dialog box.



Input variables

Input variables are added to the module and connected to their source variables in the input dialog box (Fig. 8-1). As many variables as required can be used. The **Negate** and **Invert** options can be used to manipulate the value of the corresponding input variable before its use in the nominated expression evaluation. If **Negate** is selected, the value used in the expression evaluation is actually $-x$, and if **Invert** is selected, it is $1/x$.

Setup

The **Settings** button is used to specify how the values of the input variables should be combined to yield the output variable value and how to treat arithmetic overflow condition. The available choices expressions and their results are listed in Table 8-1. If one or more of the inputs have the **Invert** option set, it is possible for one of those inputs to be zero and thus cause an arithmetic error during model execution (division by zero). To allow the user to specify the action that should be taken in such an eventuality, the **Handling Division By Zero** section of the dialog becomes available if any input has **Invert** set. One of three actions may be specified; (a) terminate the simulation, (b) Add 0.0001 to the input value (divisor), thus making it non-zero, or (3) set

the output variable of the module to “undefined”. *Note that if the output variable is used as input to another module, option (c) must not be selected.*

Table 8-1 Names of expressions and the summary of the expression.

<i>Name</i>	<i>Expression</i>
Sum	$V = A + B + C + \dots$
Product	$V = A \times B \times C \times \dots$
Complement Product	$V = 1 - (1-a) \times (1-b) \times (1-c) \times \dots$
Minimum	$V = \text{Minimum}(A, B, C, \dots)$
Maximum	$V = \text{Maximum}(A, B, C, \dots)$
Average	$V = (A + B + C + \dots)/N$
Range	$V = \max(a,b,c, \dots) - \min(a,b,c, \dots)$

Fig. 8-2 The Expression module’s Settings dialog box.

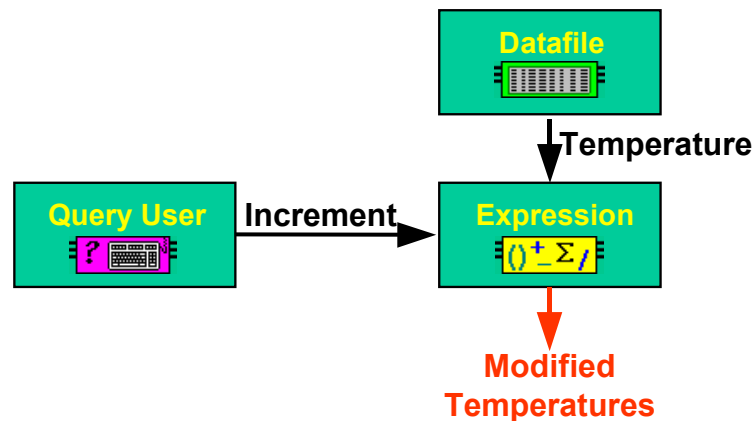


As an example, assume mean temperature is needed somewhere in the model to calculate a mortality. An **Expression** module would be added to the model with the *Sort Order* chosen to ensure that it appears before the module that uses the mean temperature.

➤ **To complete the above example**

- 1.** Add two variables to the **Inputs** variable dialog box by left-clicking on **Add Extra Input** twice.
- 2.** Link these variables to Minimum & Maximum Temperature by selecting each in turn.
- 3.** Return to the module dialog by clicking **Ok**, and select **Settings**. Then select *Average* from the **Settings** dialog. Note that the **Handling Division By Zero** options are all disabled because neither input is *inverted*.
- 4.** Go to the **Outputs** dialog box, select the Expression Output variable, and rename it to *Mean Temperature*.

Fig. 8-3 Example Expression Module diagram.



Another use for the **Expression** module could be to add a constant increment to a temperature variable (*Temperature*) read using a **DataFile** module (Fig. 8-3). To do this, add a **Query User** module, which will be used to supply the value of the constant (*Increment*). Then add an **Expression** module taking 2 inputs and link one to the *Temperature* and the other to the *Increment* variable. In the **Settings** dialog box set the expression operation to Sum.

8.2 The Equation Module

What is an Equation module?

The **Equation** module is a module that has one output variable, up to 9 input variables and up to 9 parameters. The value of the output variable reflects the result obtained by evaluating a user-supplied equation using the values of the module's input variables. Equation modules are able to take part in a model's sub-population structure.

The equation can be edited by clicking on the **Settings** button in the **Module Window** (Fig. 8-4). The syntax for the equation is the same as that for user-defined functions generally (see Section 10.2), with the exception that more than one independent variable can be specified. The independent variables are designated $[x1]$, $[x2]$..., up to $[x9]$. When the equation is evaluated, each equation variable will obtain its value from the corresponding variable specified (and linked) in the module's inputs. Hence, the number of independent variables specified in the equation must equal the number of module input variables.

Each equation parameter (designated by $[p1]$, $[p2]$, ...) corresponds to a module *factor*. As such, it may be designated as a model parameter or function. In the latter case, the equation parameter's value is obtained by evaluating the function. Note that the case of an equation that consists of only one parameter and no independent variable (i.e., "[p1]"), the Equation module is equivalent to a **Function** module.

A number of errors may occur during equation evaluation and these need to be handled in some way. Common examples of such errors are division by 0, logarithms of 0 or negative numbers, or equation inputs that are undefined at

some time. The following actions may be specified in response to the occurrence of such an error:

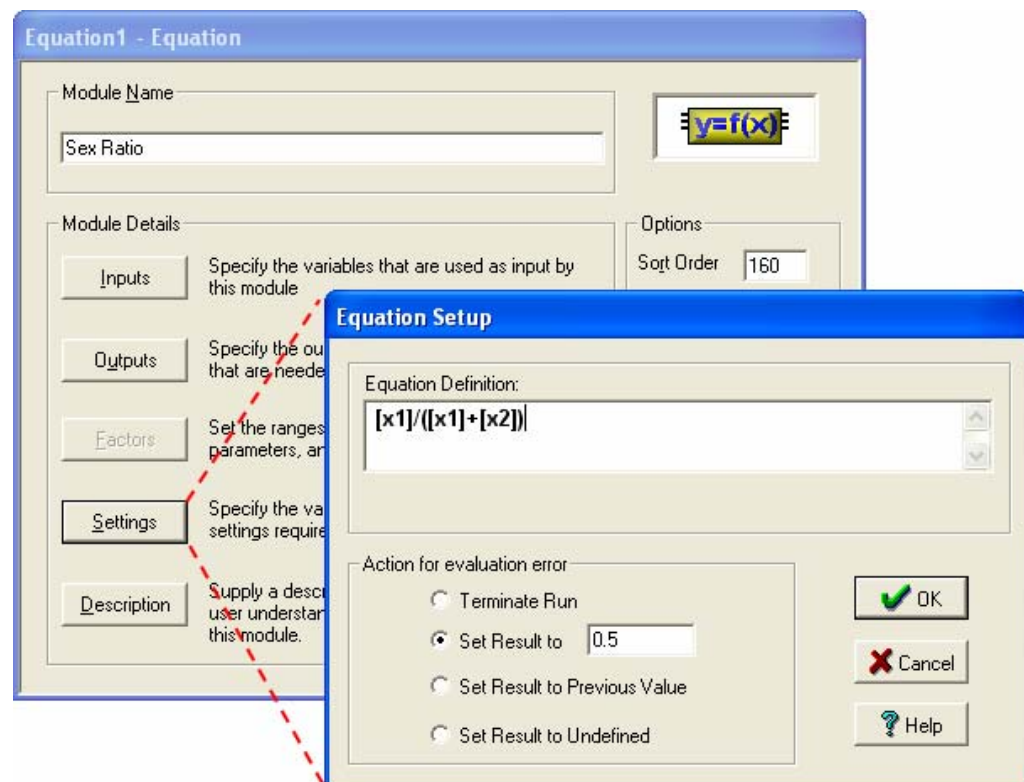
1. Terminate the simulation.
2. Set the output to a specified value.
3. Set the output to its value at the previous timestep.
4. Set the output to a special value that indicates “undefined”.

Note that in case of option 4, the equation output variable must not be used as input to another module or function. The example shown in Fig. 8-4 shows use of option 2, where the sex ratio is set to 0.5 (50:50) for those times when there are no individuals in the population. Note that it would be just as valid to use option 4 *if the output is not used as input anywhere else*.



*As an alternative to setting option 2 in the above example, the error condition (i.e., $[x1]+[x2]$ equal to 0) could be caught within the equation definition itself using an **if(...)** or **ife(...)** function. In that case, the user should set option 2.*

Fig. 8-4 The Equation Module’s “Settings” dialog box..



To complete specification of an Equation module

1. Left click on the **Settings** button in the Module Window, which opens the **Equation Setup** dialog box.
2. Type the required equation into the edit box (Fig. 8-5), using the syntax described in Section 10.2.
3. Decide what error conditions may occur in evaluating the equation during the simulation (for example, division by 0) and how such conditions should be handled. Select the required action from those

listed in the **Action for evaluation error** box. Click on **Ok** to close the dialog and return to the Equation module dialog.

4. Click the **Inputs** button to open the Input Dialog. Select each input variable in turn and link to one of the available variables listed in the drop-down box on the right of the dialog. Click **Ok** to return to the **Equation** module dialog box.
5. Open the **Outputs** dialog box, select the variable, and give it an appropriate name using the **Rename** button. Return to the **Equation** module dialog box.
6. Click on the **Factors** button to open the Factors dialog box.
7. Select each factor (equation parameter) listed in window at the left in turn. Click on the **Set Parameter** button to specify a parameter for that factor, or the **Set Function** button to specify a function. The former will open the Parameters dialog, where the parameters range and default value can be set. The latter will lead to the function specification dialog, where the appropriate function must be from the drop down list (e.g. Exponential), and the **Independent Variable** is selected.

8.3 The Counter Module

The **Counter** module is a specialized type of **Equation** module. In this module, the result of the equation evaluation is not used directly as output, but as a condition (gate) for incrementing a counter. During every timestep that the equation evaluates to a positive value, the counter is incremented by a specified value. During all other timesteps, the counter's value remains unchanged. The counter is reset to 0 at the start of a simulation. The **Counter** module's output at any time reflects the current value of the counter. The **Counter** module is able to take part in a model's sub-population structure.

The input and output variables are set up exactly as for the Equation module (Section 8.2). The "Settings" dialog box is also very similar – the form of the equation must be specified, and the **Action for evaluation error** selected. Note that only **Terminate** and **Set to Undefined** are available for the latter as options for the Counter module. The only extra item that needs to be specified is the increment to use for the counter. This is typed into an edit window below the equation definition.



An example of the use of a Counter module would be to count the number of times during a simulation the population of adult flies exceeded a threshold of, say, 1000. In that example, the input variable would be the "*Number of Adult Flies*", the equation to be used would be " $y = [x1] > 1000$ ", and the counter increment would be set to "*I*".

8.4 The Function Module

What is a function module?

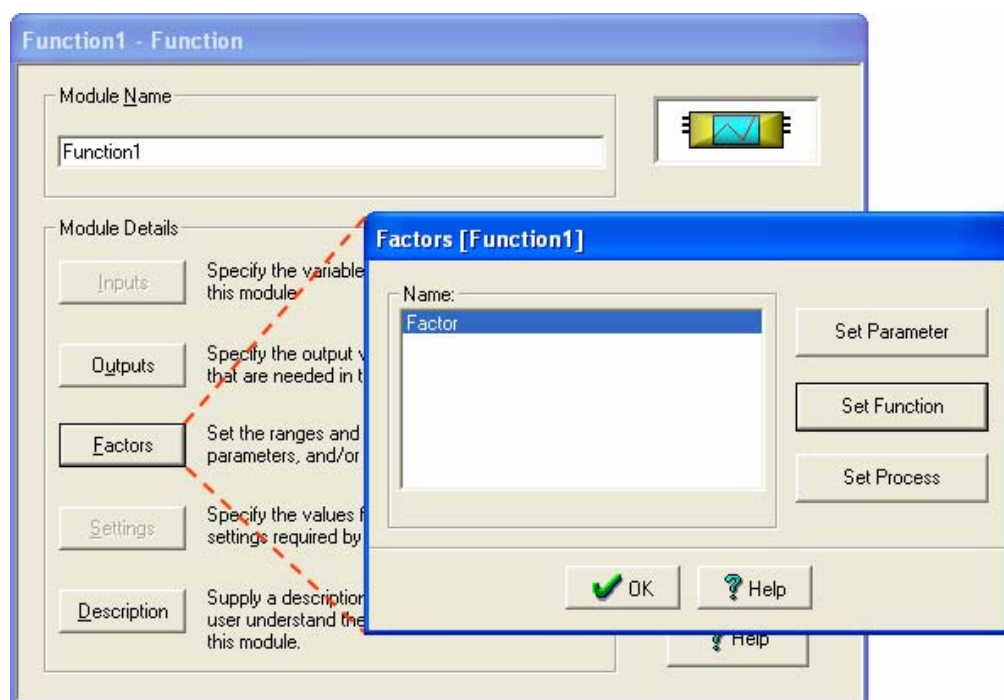
The **Function** module is a simple module that has no input variables, one (**Function**) to five (**Function5**) output variables, and a single factor corresponding to each output variable. When a **Function** module is created, the number of outputs must be known – i.e., it is not possible to change (say) a **Function2** module to a **Function3** module without deleting the existing module and creating a new one in its place. Multiple output **Function** modules behave exactly the same as an equivalent number of single output **Function** modules – the multiple output capability merely allows the module count to be reduced by grouping together logically related functions. Any of the **Function** modules can take part in a model's sub-population structure.

In its simplest form, the factor is a Parameter and the output variable reflects the value of that Parameter. Then the module behaves in much the same way as a **QueryUser** module, with the difference that the value of the output variable is modified from the Parameter dialog rather than a module initialization dialog. Generally, however, the parameters in a **Function** module will be replaced with functional relationships (see Section 2.3). In this case, the module has implicit input variables (the independent variables in the functional relationships), and a number of parameters appropriate to the Function Templates used.



Any of the pre-defined functions may be used within the Function module. For a list of these functions see the *Reference Guide or Help System*. If none of the pre-defined functions are suitable, you can define your own function and then use it within this module. See Section 10.1 for details on how to do this.

Fig. 8-5 The Function Module's Factor dialog box.





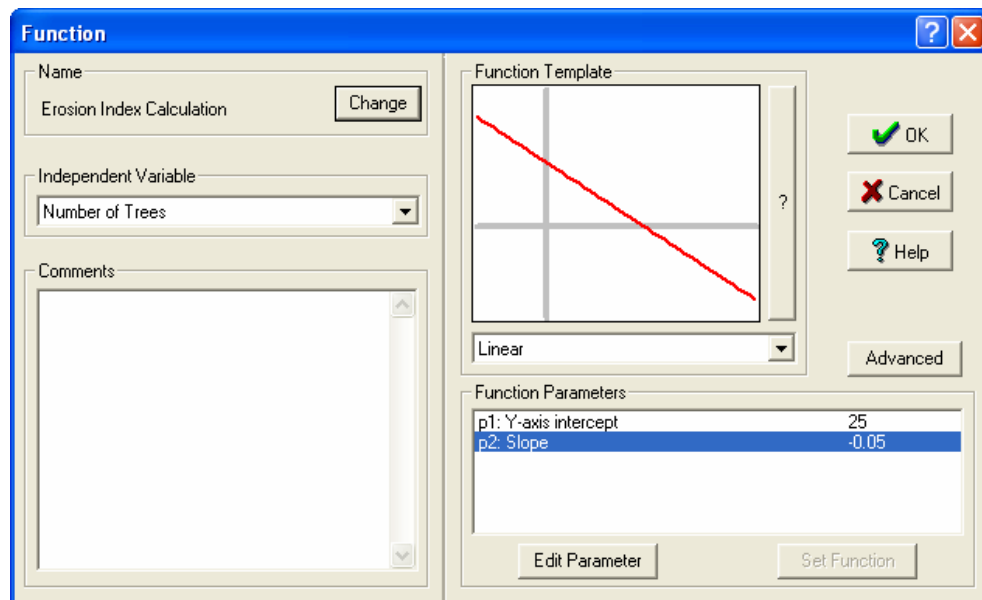
A **Function** module has as many Factors as it has outputs (from 1 to 5). As many Function modules as required may be used in a model.

The function module could be used to tie mean number of trees to an erosion index (ER index) with a linear relationship.

➤ **To specify a Function module**

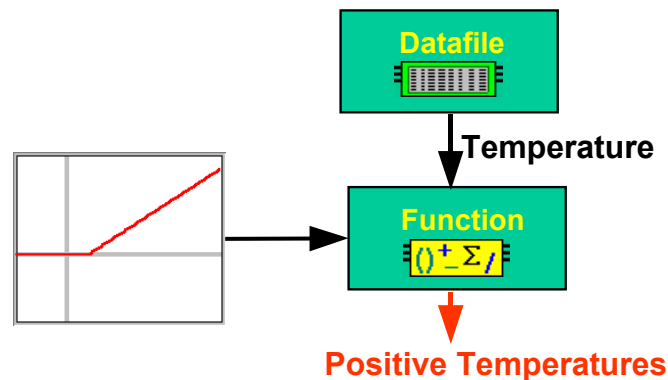
1. Create the module by selecting the **Add Module** menu option from the **Model** menu. Choose **Function** to create a module with a single output, or **Function n** where n is the number of outputs required (2 to 5).
2. Double-click on the module symbol in the **Model Components** window to get to the **Module** dialog.
3. Left click on the **Factors** button and open the Factors dialog box.
4. Left click on the **Set Function** button and open the function dialog box (Fig. 8-6). Choose the appropriate function from the drop down list (e.g. Linear) and select the **Independent Variable** (e.g. Number of Trees).
5. Go back to the **Function** module dialog box and open the **Outputs** dialog box, select the variable, and give it an appropriate name (such as Erosion Index) using the **Rename** button.

Fig. 8-6 A Function dialog box.



A **Function** module could be used to filter temperature values so only positive temperatures are input in to the model using an above threshold function with the threshold at zero degrees (Fig. 8-7).

Fig. 8-7 An example Function module diagram.

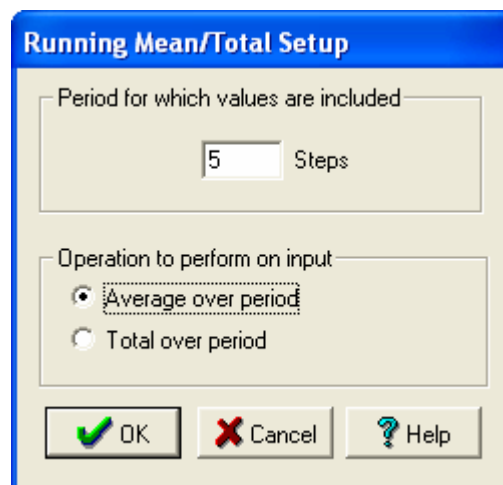


8.5 The Running Mean Module

The **Running Mean** module has a single input variable. The modeller specifies an averaging period, and then the output variable gives a running average (or total) of the input variable taken over that period. The period is specified in module steps (either timesteps, or segments if the module is in segmented timestep mode). The period used includes the current step and the (*period-1*) previous steps. Fewer steps will be used for averaging or totalling at the start of the run. The module is useful for smoothing variables, where values can fluctuate widely from one step to the next. For example, low temperatures may induce diapause in an insect, but rather than use an actual minimum temperature value directly, it may be better to use a running mean over a period of (say) a week, so that one isolated, unusually cold night is not sufficient for the diapause to trigger.

Setting up this module is very simple – the Settings dialog is shown in Fig. 8-8. Only two items need to be specified, the number of steps to be included in the period, and whether to average or total over that period.

Fig. 8-8 The Running Mean Settings dialog.



8.6 The Difference Module

What is a Difference module?

The **Difference** module is a simple module that has one input variable and one output variable. It has no parameters, nor any user-adjustable settings. The value of the output variable (y_t) is the difference between the input variable (x) at the current timestep (x_t) and at the previous timestep (x_{t-1}). It therefore provides us with the change in the variable between timesteps.

$$y_t = x_t - x_{t-1}$$

To complete specification of a Difference module

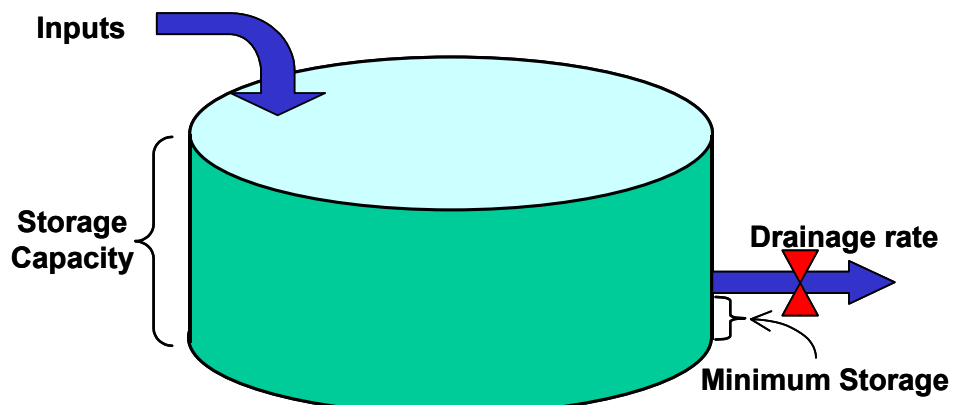
1. Click the **Inputs** button in the Module Window to open the Input Dialog. Select the input variable and link to one of the available variables listed in the drop-down box on the right of the dialog. Click **Ok** to return to the **Difference** module dialog box.
2. Open the **Outputs** dialog box, select the variable, and give it an appropriate name using the **Rename** button. Return to the **Difference** module dialog box.

8.7 The Storage Container Module

What is a Storage module?

The **Storage Container** module simulates a storage device with a nominated capacity (in any unit, eg m^3), which is filled (or emptied) using rates determined by the input variables and a “Drainage rate” parameter. If it is filled above capacity, the excess overflows, while a minimum storage capacity can also be set, beyond which the container cannot be emptied. Fig. 8-9 illustrates this module schematically.

Fig. 8-9 Schematic diagram of the operation of the Storage Module.



Up to 4 input variables may be used. Each of these inputs is a “Fill Rate”, the amount (in the same units as the Storage Capacity) being added to the store

during a timestep.

Two Output Variables are available from this module. The first, “Current Storage” is the current amount in the store, while the second, “Overflow”, is the amount lost during the current timestep due to the Storage Capacity being exceeded. These outputs can be in the same units as the Storage Capacity, or scaled as proportions of Storage Capacity, as selected by the model builder (see below).

Factors Three factors have to be set by the user. As usual, the factors can be parameters, functions or processes.

- **Total Storage Capacity** (permitted range: >0): The total capacity of the store, in any required units. Whatever units are chosen here, the input variables must be in the same units.
- **Minimum Storage** (permitted range: 0-**Total Storage Capacity**): The minimum level that the store may drop to if filled above that level. In other words, no drainage can occur until this minimum level is reached. The units for this factor must be the same as those for **Total Storage Capacity**.
- **Drainage Rate**: This factor determines the amount of store contents lost per timestep. If the “Method of Drainage application” (see Settings) is Direct, then the parameter is in the same units as the Total Storage Capacity. Otherwise, the amount drained from the Storage is evaluated as the difference between the current Storage and the Minimum Storage, multiplied by the value of the factor.

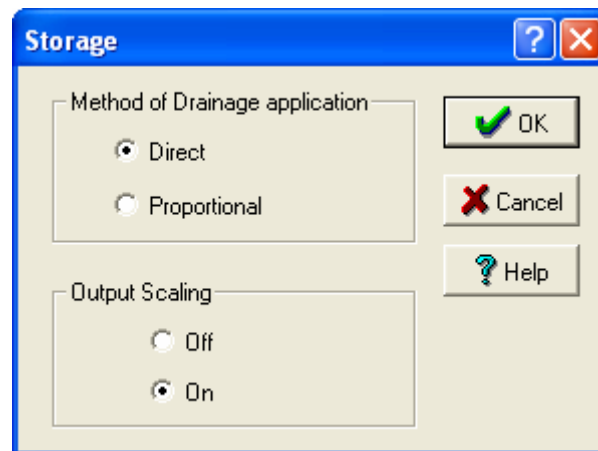


One use of this module could be to keep a tally of fuel (dead plant material) in a plant population model for determining the intensity of fires. One or more of the inputs could be derived from the Cohort Output “Total Stage Mortality” (Table 6-4) of the plant lifecycles.

➤ **To complete a Storage module**

- 1.** Left click the **Inputs** button.
- 2.** Link the **Fill1** input by left-clicking on the variables and using the drop down menu to find the appropriate variable. Link as many of the other three inputs as required.
- 3.** Go back to the **Storage** dialog box and left click on the **Outputs** button.
- 4.** Select either of the output variables and rename it to an appropriate name. Do the same with the other output variable if it is needed. Return to the **Storage Module** dialog box and left click on the **Settings** button to obtain the Storage Settings dialog.
- 5.** Choose the appropriate **Method of drainage application** (see discussion of **Drainage Rate**, above). Then choose how the outputs should be scaled - either directly in units of Total Storage Capacity (**Output Scaling** is **Off**), or as a proportion of the Total Storage Capacity (**Output Scaling** is **On**).

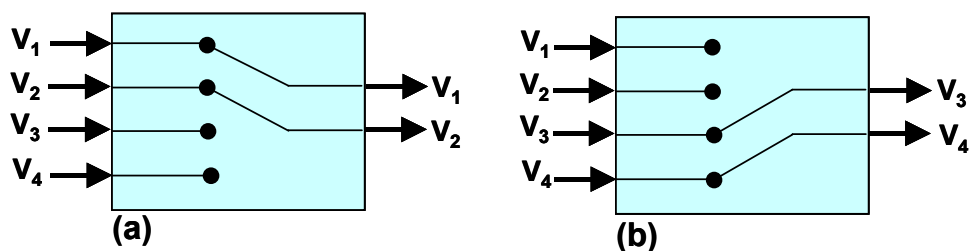
Fig. 8-10 Storage module Setup dialog box.



8.8 The Switch Module

The **Switch** module is a simple way of combining two or more sets of output variables into one single output stream. The effect is exactly like a multiple pole switch, as illustrated in Fig. 8-11. That example shows a Switch module with two output variables. These output variables are derived from two sets of input variables (V_1, V_2) and (V_3, V_4). Each of these sets constitutes an **input channel**. Either channel can be selected as the source for the output variables when the model is run in the *Simulator*.

Fig. 8-11 Diagrammatic representation of a 4-input, 2-output Switch model showing (a) the first set of outputs selected and (b) the second set of outputs selected.



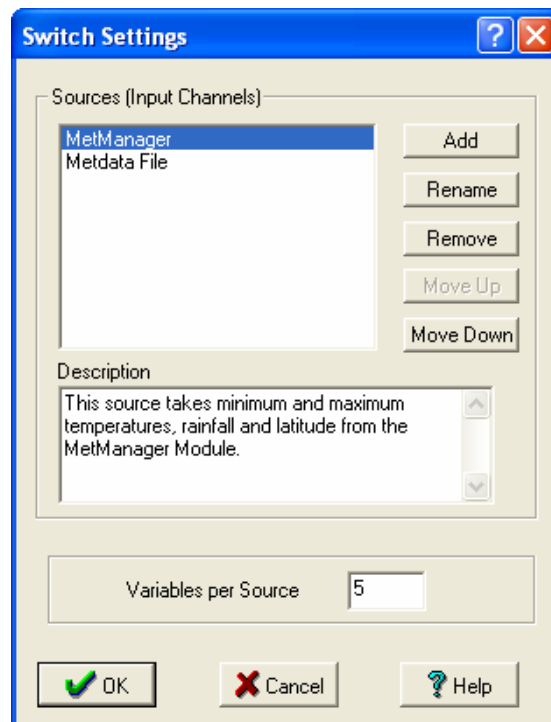
An example where the Switch module may be useful is for models that could use meteorological data read either from a **DataFile** or derived from a **MetManager** module. The Switch module would then be used to select between the two data sources. Note that the current version of the Switch module is implemented in a rather simplistic fashion. The output variables must have names that are different from the input variables (as all these variables belong to the model's global name space). If, for example, a minimum temperature is being supplied to input V_1 from a **DataFile** module and to input V_3 from the **MetManager** module, the first output variable from the Switch module might be named *Minimum Temperature*, with the variables

V_1 and V_3 named *Minimum Temperature (File)* and *Minimum Temperature (MetMgr)*, respectively.

➤ **To specify and set up a Switch module**

1. Create the module by selecting the **Add Module** menu option from the **Model** menu. Choose **Switch** to create the new module and add it to the end of the module list.
2. Double-click on the module symbol in the **Model Components** window to get to the **Module** dialog.
3. Click on the **Settings** button to open the **Switch Settings** dialog box (Fig. 8-12).
4. Click the **Add** button to create a new input “source” channel and give it a suitable name. A fuller description of the input channel can be provided in the **Description** panel when the source is selected in the list box.
5. Repeat step 4 for each required source. The module is currently limited to a maximum of 10 source channels.
6. In the **Variables per Source** box, specify the number of variables that will be provided by each source. This number is also the number of output variables from the module.

Fig. 8-12 The Switch module Settings dialog box.



7. With the sources correctly set up, return to the module dialog box and click on the **Inputs** button. The correct number of inputs will be shown in the left panel, and these can be connected to the appropriate variables in the usual way. If there are m sources and n

input variables per source, the input variables are listed in the following order

Source 1, Variable 1

Source 1, Variable 2

...

Source 1, Variable n

...

Source m, Variable 1

Source m, Variable 2

...

Source m, Variable n

- 8.** Return to the module dialog and click on the **Outputs** button. Select, rename and document the output variables as appropriate.



Always check the linkage of input variables when modifying a previously set number of variables per source, as input variables may have been reassigned to the wrong source.

9. Specialised Modules

9.1 The Circadian Module

What is a circadian module?

The Circadian module is designed to generate a variable that describes the diurnal change in the value of some quantity. This is particularly useful in situations such as the calculation of temperature-based development rates, where input data is available as minimum and maximum daily temperatures.

Variables in DYMEX generally are assigned a single value for each timestep. The **Circadian** module output variable is different in that it has a set of values (the exact number is determined by the modeller). For example, with the default 24 steps, there is one value for every hour over the period of a day. These values are obtained by interpolation, using one of 3 functions selected by the user. When this output is then used as input to a function, the function is evaluated for each of these values separately, *as if the model had a timestep of 1 hour*. When the output variable is used directly as input to another module, its average value over the 24 segments is used.

Two types of Circadian modules are available in DYMEX (**Circadian** and **CircadianAdjust**). These operate almost identically, the only difference being that the **CircadianAdjust** module has two factors that allow the supplied *Daily Minimum* and *Daily Maximum* values to be adjusted before their use in the calculation of the daily cycle. This is done via two factors. The first, **Adjustment for Minimum**, is evaluated during model execution and added to the Daily Minimum. The second, **Adjustment for Maximum**, provides a

similar adjustment for the Daily Maximum. In the following description, any procedure or explanation that relates only to the **CircadianAdjust** model will be qualified with the phrase [*CircadianAdjust only*].



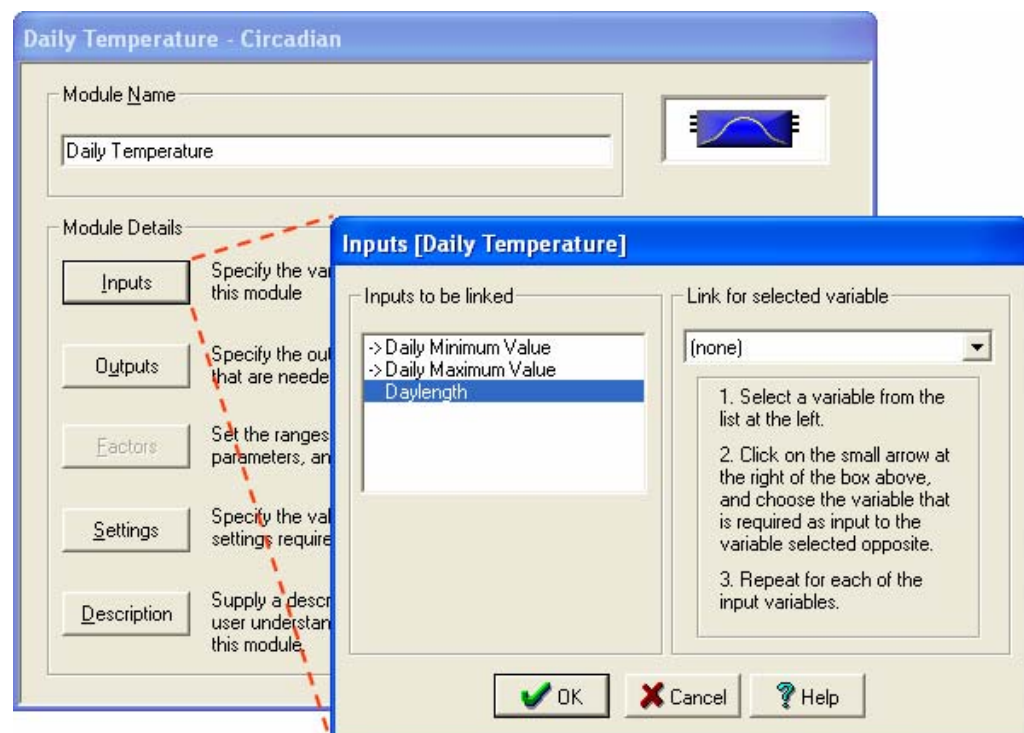
The **CircadianAdjust** module could be used to estimate the daily cycle of temperatures in the soil, by using the daily maximum and minimum air temperatures as inputs. The two factors would be used to adjust these inputs to provide daily soil maximum and minimum temperatures.

Input
variables

The module takes as its input the minimum and maximum values of the quantity during the timestep, as well as daylength. While the minimum and maximum values are necessary, daylength is not used for the **Sine** cycle shape. Daylength is used only in the **Composite (Sine+Sine)** and **Composite (Sine+Exponential)** (see *Settings below*). The next day's minimum (if available) will be automatically used for calculating the shape of the cycle after the maximum for the **Composite (Sine+Sine)** and **Composite (Sine+Exponential)** shapes.

*Multiple **Circadian** modules are allowed in a model.*

Fig. 9-1 Circadian module dialog box and its input dialog box.

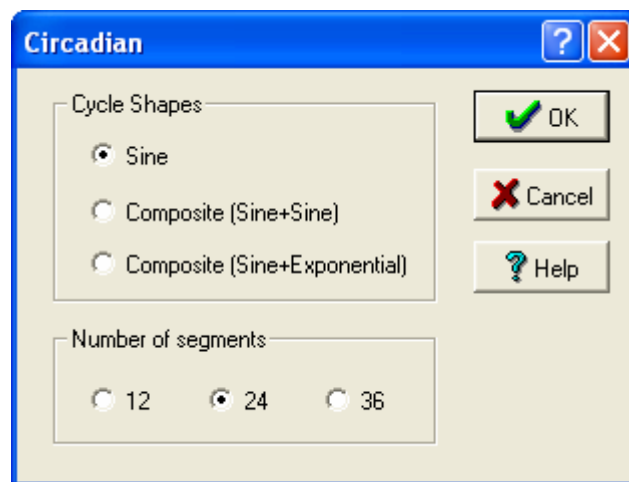


➤ To complete a Circadian module

- 1.** Left click the **Inputs** button.
- 2.** Link **Daily Minimum & Maximum Value** and **Daylength** if needed by left-clicking on the variables and using the drop down menu to find the appropriate variable (e.g. Maximum and Minimum Temperature).

3. [*CircadianAdjust only*] Go back to the **Circadian** dialog box and left click on the **Factors** button. Set the **Adjustment for Minimum** and **Adjustment for Maximum** factors as required. Note that the value of each factor will be added to the Daily Minimum and Maximum, respectively.
4. Go back to the **Circadian** dialog box and left click on the **Outputs** button.
5. Select the **Daily Cycle** variable and rename it to the appropriate name (e.g. Daily Temperature Cycle).
6. Go back to the **Circadian** dialog box and left click on the **Settings** button.
7. Choose the appropriate **Cycle Shapes** (e.g. Sine).
8. Select the required number of segments DYMEX should use to approximate the selected Cycle Shape (12, 24 or 36).

Fig. 9-2 Circadian module Setup dialog box.

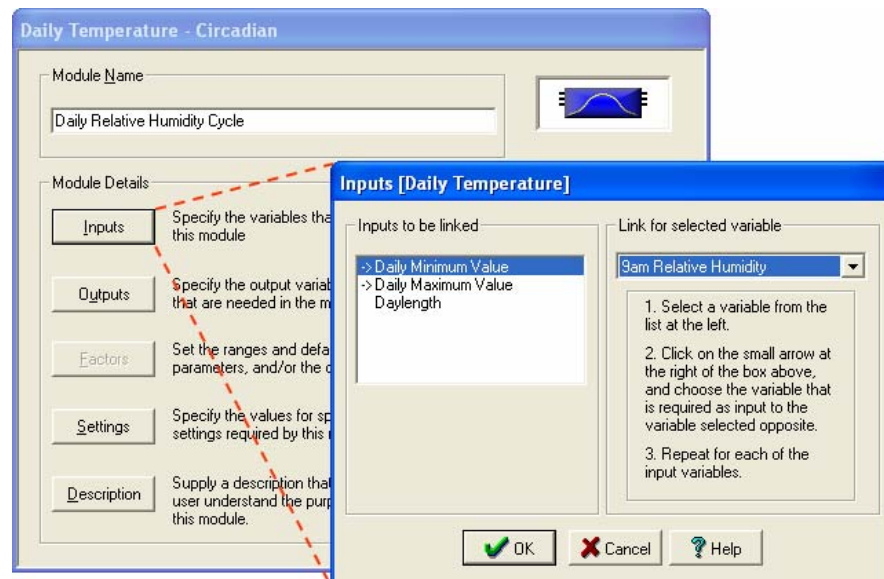


The **Circadian** module can be used to estimate 2-hourly temperatures during a day to calculate how many day-degrees are experienced by a lifestage. Alternatively, it could be used to model daily fluctuations in relative humidity. Note, however, the assumptions that must be satisfied when using other cycle shapes than the **Sine**.

Below in Fig. 9-3 is an example of using the **Circadian** module to model daily cycles in relative humidity. 9am RH has been used as the Daily Maximum Value, 3pm RH has been used as the Daily Minimum Value and Daylength is left unlinked.

Settings The **Settings** button allows you to change the cycle within the **Circadian** module, as well as the number of segments to be used each day. There are three cycle shapes available. Both the **Composite (Sine+Sine)** and **Composite (Sine+Exponential)** shapes assume that a minimum occurs before a maximum, and that the maximum occurs before sunset. For the descriptions that follow, the following definitions are assumed (refer to Figure 9-4):

Fig. 9-3 An example circadian module using RH.



a = the time (in hours) between mid-day (halfway between sunrise, t_r , and sunset, t_s) and the time when the modelled quantity is at its maximum (t_{max}).

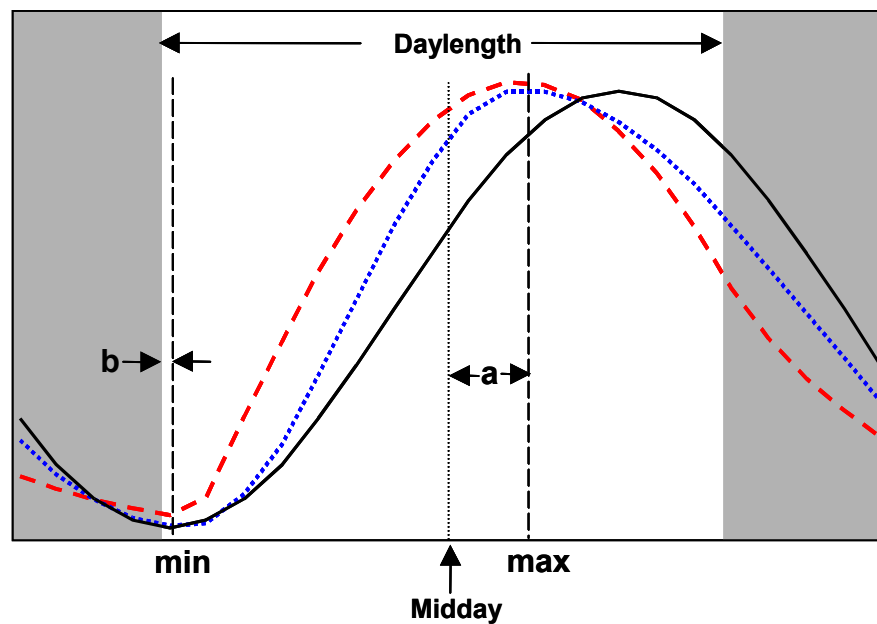
b = the time (in hours) between the time when the modelled quantity is at its minimum (t_{min}) and sunrise (t_r).

Daylength is the number of hours between sunrise (t_r) and sunset (t_s).

- **SINE** is a simple sine curve whose period is 24 hours. Note that the length of the increasing and decreasing parts of the cycle is not affected by daylength, each being always equal to 12 hours. This can give rather poor approximations of the daily change of quantities that are affected by daylength (such as temperature) over the course of a year, especially in locations far from the equator.
- **COMPOSITE (SINE+SINE)** is two sine curves joined together at the maximum value. The first curve models the rise in the quantity from the day's minimum value to its maximum, with a rise-time (or half-period) equal to the time between minimum and maximum. The second curve models the fall from maximum to subsequent minimum, with a half-period equal to the interval between these points. The model uses values of 1.86 and -0.17 for **a** and **b**, respectively. See Wann, M., Yan, D. & Gold, H.J. (1985) Evaluation of three models of daily cycle of air temperature. *Agric. For. Meteorol.*, 34: 121-128 for more details.
- **COMPOSITE (SINE+EXPONENTIAL)** uses a portion of a sine curve from the first daily minimum to sunset, joined to an exponential decay function from sunset onwards. The sine curve's period is chosen so that its maximum corresponds to the modelled variable's maximum, and its inflection point corresponds to the variable's minimum. The exponential (decay) curve is used to model the fall in value of the quantity being

modelled from sunset to sunrise. The model uses values of 1.86 and -0.17 for **a** and **b**, respectively, and 2.2 for the decay coefficient of the exponential portion of the curve. See Parton, W.J. and Logan, J.A. (1981) A model for diurnal variation in soil and temperature. *Agric. Meteorol.*, 23: 205-216, for a detailed description of this model.

Figure 9-4 Sine (—), Composite (Sine+Sine) (.....) and Composite (Sine+Exponential) (- - -) shapes, in relation to daylength and daily extremes



Circadian
output as
driving
variables



When using circadian output variables to drive functions it must be remembered that 24 calculations are performed each model timestep to obtain the final function output (if using 24 segments in the cycle). Assume a model has a “Linear above Threshold” function driven by daily temperature cycle. If the threshold of the function is 10° and the mean temperature for the day is 7° , it is nevertheless possible that the output from the function will be greater than zero for that day. This would occur if the maximum temperature for the day exceeded the threshold of the function. The proportion of the day above the threshold temperature will have an influence on the output of the function.

9.2 The Event Module

What is an
event
module?

The **Event** module allows for model operations that are triggered on set dates or by set thresholds. These operations would include management actions such as pesticide spraying and harvesting of a crop. The Event module has up to four input variables, and a varying number of output variables.

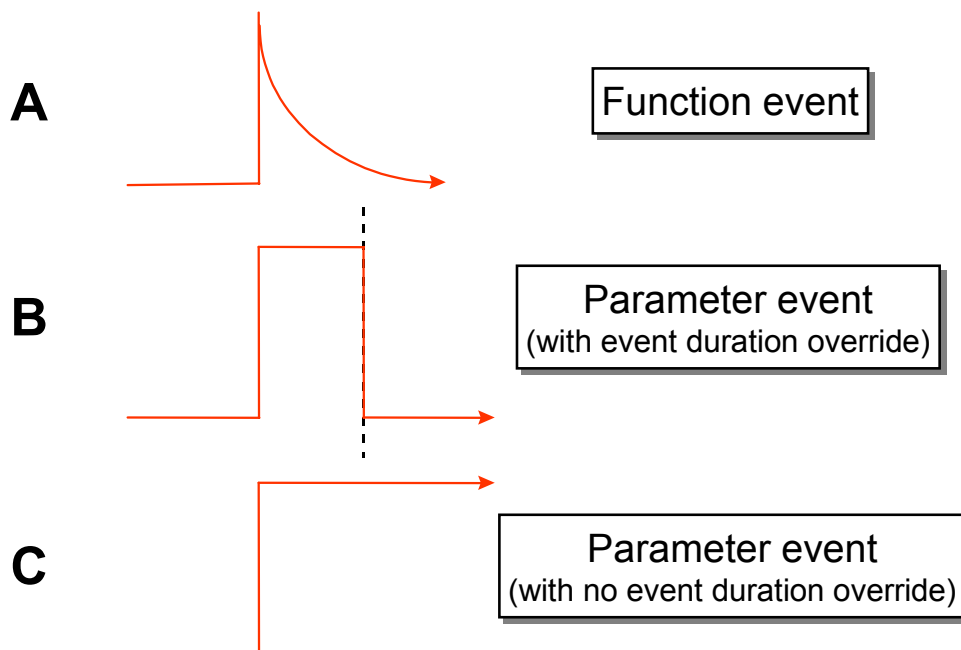
Several types of Event module are available. The simplest **Event** module has one Event Output Variable output and a corresponding factor (parameter,

function or process) that controls the way the output variable reacts to the event's occurrence. Other Event modules with 2 to 5 Event Output Variables are available (named **Event2** to **Event5**, respectively), with each of these modules having one factor determining each Output Variable's response. A further module, **EventWithDelay**, is similar to a simple **Event** module with a delayed event response, the length of the delay being controlled by a factor. In this section, any feature that applies only to a particular type of Event module will be noted.

Any of the **Event** modules can make use of the model's sub-population structure. To do this, the "**Use sub-populations**" in the **Module dialog** must be checked. When the model is subsequently run in the **Simulator**, different event triggers will be settable for the different sub-populations.

An event is triggered when the appropriate date or threshold is reached during a simulation run. A single parameter associated with the Event module, and the **Event Duration Override** setting, control the precise way in which the event is applied and runs its course after it has been triggered. A function or process can be substituted for the parameter to achieve effects such as an exponentially declining event effect. The **Event Duration Override** setting will force termination of the event.

Fig. 9-5 An illustration of three types of Events (showing the Event Variable output variable variable).

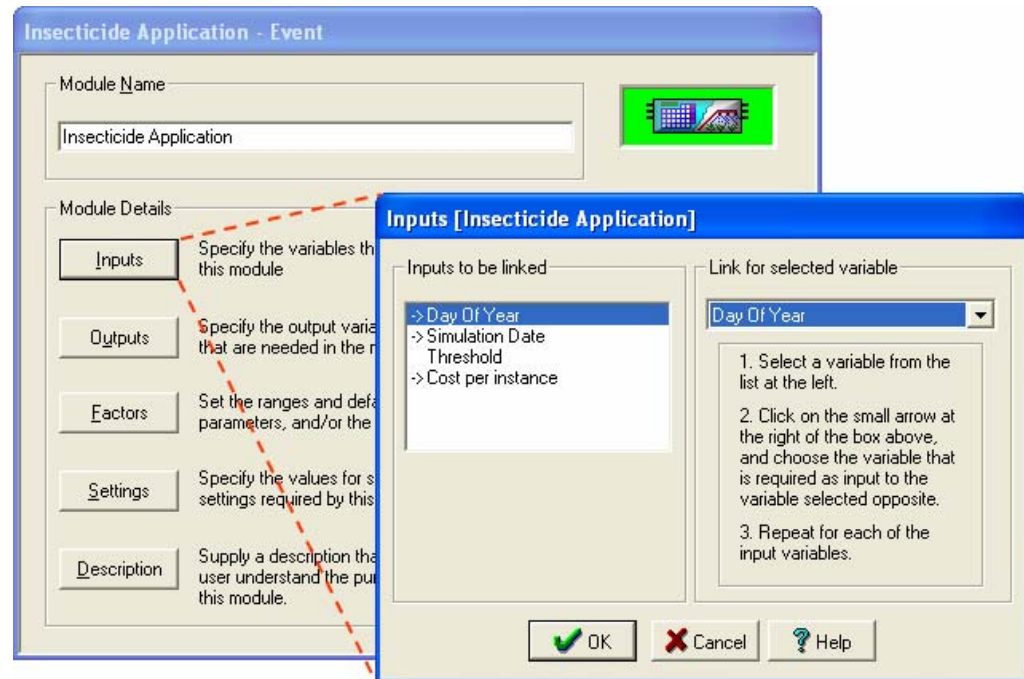


Input
variables

Event modules have four possible input variables: **Day of Year**, **Simulation Date**, **Threshold**, and **Cost per instance**. If **Day of Year** or **Simulation Date** are used, they need to be linked to the appropriate **Timer** module variables. **Threshold** is used when occurrence of an event depends on the value of a model variable (e.g. we may want to spray a pest only when numbers exceed a

predefined threshold). See the paragraph on **Settings** below for further information on Threshold conditions. **Cost per instance** is the cost of each occurrence of an event (for example, the cost of one spray application). If no variable is assigned to **Cost per instance**, it is assumed to be equal to 1.

Fig. 9-6 Event module's dialog box.



Factors

Each response factor is a Parameter, a Function (response function) or a Process. If the Parameter option is used, the corresponding output variable (**Event Variable**) will reflect the Parameter's value after each event is triggered. If the Function or Process option is used the value of Event Variable at any timestep after an event has been triggered will be determined by the value of the current value of that factor. (See Section 2.3 for more information).

In the **EventWithDelay** module, the *Delay Time* factor can be a parameter (giving a fixed delay time), or it can be variable when a function or process is used. This factor's value is interpreted as the number of days (rounded upwards) that the event action will be delayed from the day that the event trigger occurs. The event with delay module is useful for simulating events when the decision to apply the event must precede it, such as when a decision to burn a paddock involves destocking ahead of the fire to allow sufficient fuel to accumulate. In this case, the factors driving the decision to burn (e.g., weed presence) may occur a year in advance of when it is logistically feasible to burn.

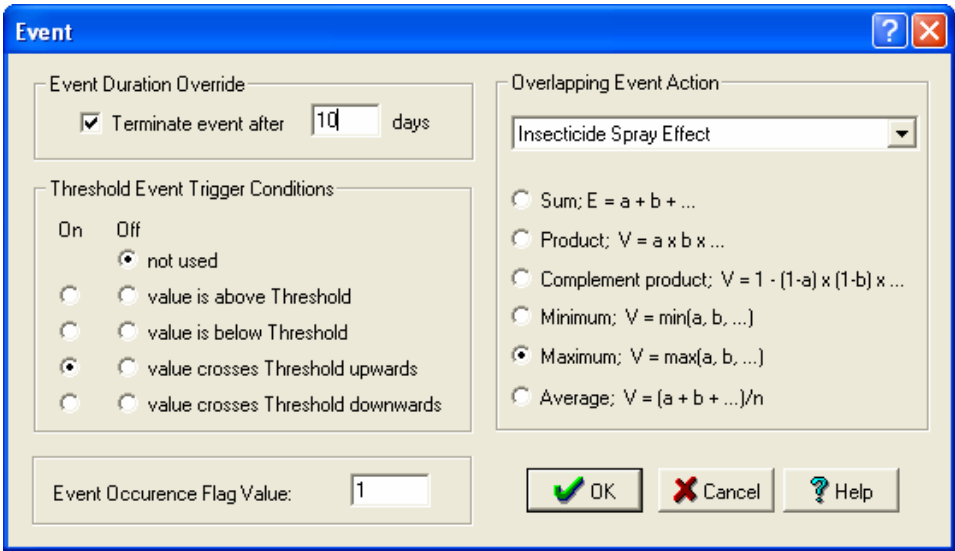
When you create a response function for the event there is a local variable available that is produced by the event: **Days Since Event**. This variable could be used to model the lessening effect of a treatment after the day of application.

Settings

Event Duration Override is used to force termination of an event. Without this override, and if a Parameter controls the Event Effect, the event will be effectively continuous (i.e., the triggering of the event acts like a switch (Fig. 9-5). To revert to the pre-triggering state you must set the **Event Duration Override** to the number of days you want the event to last. In multi-output Event modules, the Event Duration Override resets every Event Effect output to its pre-triggered state.

The **Threshold Event Trigger Condition** defines what type of condition is used to trigger the occurrence of an event, and what type of condition will terminate it. On conditions (i.e., conditions that initiate an event action) and Off conditions (conditions that terminate an event action) may be specified. In Fig. 9-7 **value crosses Threshold downwards** indicates that the event will only be triggered during timesteps when the **Threshold** variable's value crosses over a value (which is defined in the *Simulator*) in a downward direction. Similarly, any current event action will be terminated when the **Threshold** variable's value crosses over another value (also defined in the *Simulator*) in an upward direction. The **Off** trigger condition can be disabled (i.e., set to **not used**), in which case the model's user cannot turn an event action off using a threshold condition.

Fig. 9-7 Event module's Settings dialog box.



The **Event Occurrence Flag Value** is the value that is used for the “**Event Flag 1**” output on those timesteps when an event action is triggered (see *Output Variables* below), and is set to 1 by default.

If two or more event actions are triggered in close succession, so that the first response has not yet “timed out”, the resulting response functions from each action must be combined in some way to give a single value to the Event Variable output. The **Overlapping Event Action** setting specifies how this situation is treated, with a choice of several ways of combining the responses.

Output variables

Table 9-1 shows the output variables available for each type of Event module and their descriptions. If a variable is not available in a particular module, its

output position is used by the next available variable (i.e., there is no “hole” in the output variable list). For example, in the **Event2** module, the output variables are: Event Variable 1, Event Variable 2, Event Cost, Event Flag 1, and Event Flag 2.

Each of the **Event Variables** describes the response to the corresponding factor. If the Event module describes a pesticide application, this variable could be used to drive the mortality in one or more lifestages, for example. The **Event Cost** is used to calculate the “cost” of the event. Its value is equal to the input variable **Cost per instance** during those timesteps that an event is *triggered*, and is 0 at all other times. This output could be used to sum the cost of the pesticide application over the duration of the simulation. The **Event Flag** outputs can be used to indicate that an event occurred. The first of these is especially suited for this purpose, as it takes the special value “undefined” when the event is not being triggered. In tables and point graphs produced in the **Simulator**, “undefined” values produces a blank field, which clearly highlights the occasions when event triggers occur. Note, however, that because of these “undefined” values, the **Event Flag 1** output is not suitable for input to another module or function. The **Event Flag 2** output should be used for this purpose (it could be used, for example, to count the number of times a threshold event was actually triggered during a simulation).

Table 9-1 Event Output variables and their descriptions.

<i>Output variables</i>	<i>Description</i>
Event Variable 1	Describes the effect of the event due to event action factor 1.
Event Variable 2	Describes the effect of the event due to event action factor 2. Available only in modules of type Event2 , Event3 , Event4 and Event5 .
Event Variable 3	Describes the effect of the event due to event action factor 3. Available only in modules of type Event3 , Event4 and Event5 .
Event Variable 4	Describes the effect of the event due to event action factor 4. Available only in modules of type Event4 and Event5 .
Event Variable 5	Describes the effect of the event due to event action factor 5. Available only in modules of type Event5 .
Event Cost	Equal to the input variable Cost per instance during those timesteps that an event is <i>triggered</i> , and is 0 at all other times.
Event Flag 1	Equals the value specified for the Event Occurrence Flag (Settings dialog) for those timesteps that an event is triggered, and is undefined at all other times.
Event Flag 2	Equals 1 for those timesteps that an event is triggered, and 0 at all other times.

➤ **To complete an Event module**

- 1.** Left-click on the **Inputs** button.
- 2.** Link **Day of Year** and **Simulation Date** to the relevant variables supplied by the **Timer** module by selecting each variable in the **Inputs to be linked**

box and using the drop down box, **Link for selected variable**, to link the variables.

- 3.** Link the **Threshold** variable to the appropriate variable (e.g. Total Number of Adults) from the drop down box if it is required.
- 4.** If needed, link the **Cost per instance** variable to an appropriate variable – this will often be an output from a UserQuery module.
- 5.** Go back to the **Event** dialog box and left click the **Outputs** button.
- 6.** Select the output variables by left-clicking the **Select** button. Then rename the variable to relevant name by left-clicking the **Rename** button.
- 7.** Go back to the **Event** dialog box and left click the **Factors** button and set the factor that determines the effect of each output event variable.
- 8.** If the module is of type **EventWithDelay**, set the factor that determines the delay between the event trigger and its action.
- 9.** Go back to the **Event** dialog box and left click the **Settings** button.
- 10.** In the **Settings** dialog box, change any of the module settings if the defaults are not adequate.

An example of a constant event with event duration override: An event that is a spray application affects the survival of an organism for 2 days then has no effect. The input variables must be set with both **Simulation Date** and **Day of Year** being linked. **Threshold** could be linked to the number of organisms in a particular lifestage. The output variable must be selected and ideally renamed to something meaningful, e.g. *Spray Effect*. The spray has a constant concentration over 2 days, therefore a “Parameter” is selected and the default value is set to the proportion mortality per timestep from contact with the spray. Since the spray only affects the population for 2 days the **Event Duration Override** (Fig. 9-7) must be selected and set to 2 days.



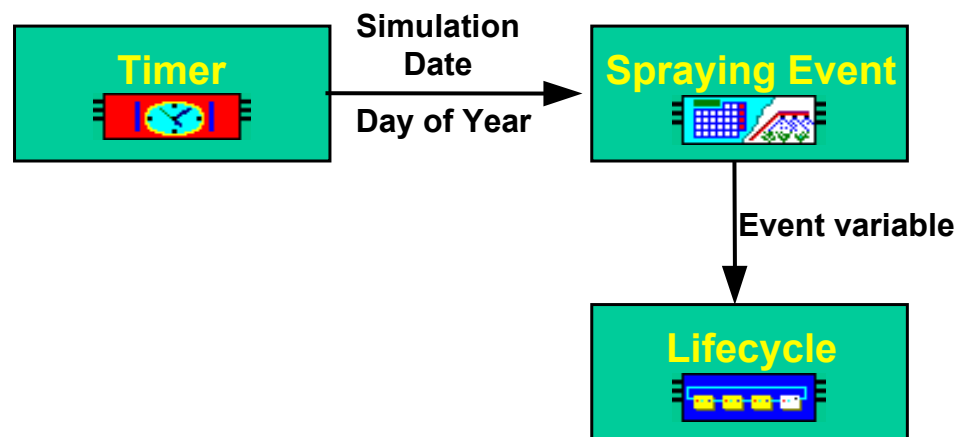
The **Event** module can be used to simulate any event that occurs on user-selected dates or in response to a threshold, such as spraying of a population of insects, harvesting of animals or plants or occurrence of a fire.

Fig. 6-33 gives an example of an event whose effect is determined by a function. This example also involves a spray treatment, but, instead of an abrupt halt to the effect, there is a gradual decline in the residual effect of the chemical.

The input variables: **Simulation Date** and **Day of Year** should be linked to the corresponding variables supplied by the **Timer** module. The output variable must be selected and renamed. Within the **Factors** dialog box choose a function and set the driving to variable **Days since Event**. From the function list, choose the “Exponential Decay” function. The parameters must be set for this function, including the initial effect of the spray, the number of days after the event that the effect of the spray starts to decline and the rate of decline.

In Fig. 9-8 the threshold value that is used to trigger an event can come from any one of the modules in the model. The number of individuals in a particular lifestage, a certain amount of rainfall, a particular temperature, etc may trigger the event.

Fig. 9-8 An Example Event module diagram.



9.3 The Daylength Module

What is the
daylength
module?

The **Daylength** module calculates the number of hours between sunrise and sunset given the **Latitude** of the location and either the **Day of Year** or **Simulation Date** (or both). Its second output variable (**Day Length Change**) gives the change in daylength that has occurred between the current timestep and the previous timestep (in hours). This will be positive if daylength is increasing and negative if it is decreasing. **Simulation Date** should be provided as input if the **Day Length Change** output is used, or that output will contain small discontinuities at the boundaries between normal and leap years. Commonly, either a Query User or Query File module provides the Latitude input required by the module.

➤ To complete the Daylength module

- 1.** Left click on the **Inputs** button and within **Inputs** dialog box link the **Latitude**, and either the **Day of Year** or **Simulation Date** (which are obtained from the **Timer** module).
- 2.** Go back to the **Daylength** dialog box and left click on the **Outputs** button.
- 3.** Select the **Day Length** variable by left-clicking on the **Select** button.



The **Daylength** module is commonly used to provide daylength for the **Evaporation** module but could be used for plant growth calculations, circadian temperature cycles, etc. see *The Circadian Module*, page 123 & *Plant Growth and Development*, page 63. The Day Length Change output is useful in triggering diapause in insects due to (say) decreasing daylength.

Fig. 9-9 Daylength Output variables dialog box.

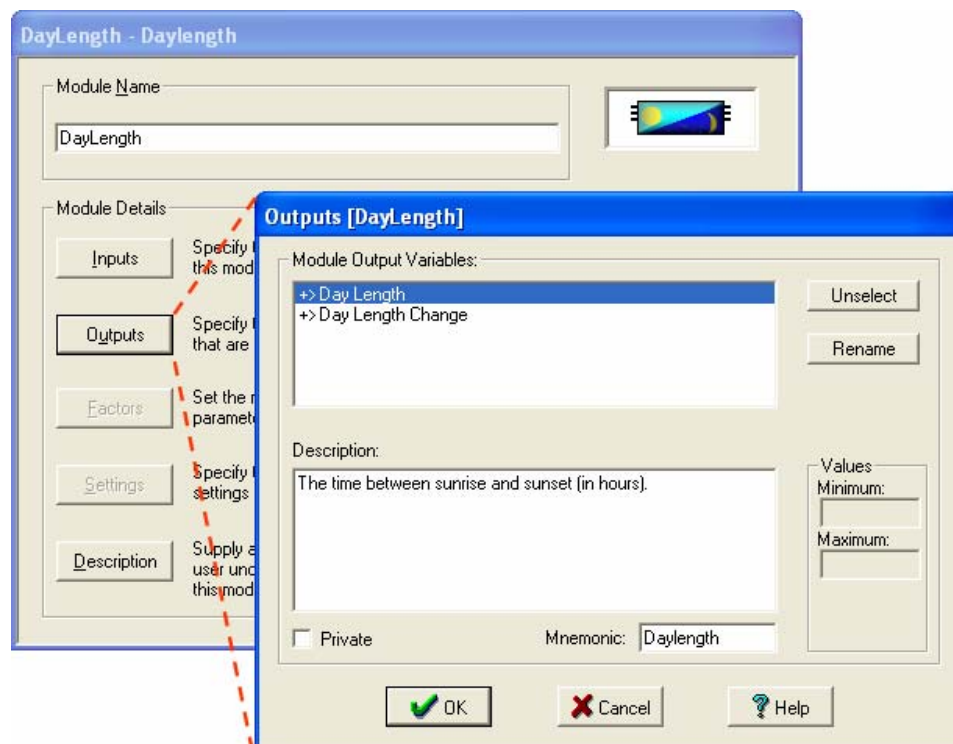
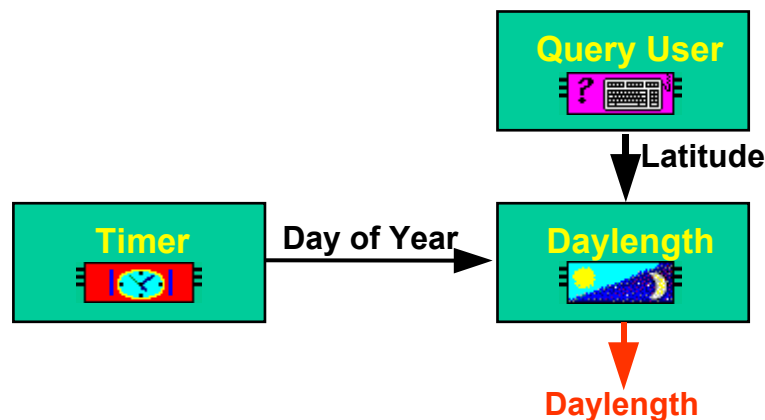


Fig. 9-10 Example Daylength module diagram.



9.4 The Evaporation Module

What is the evaporation module?

The **Evaporation** module calculates the pan evaporation (in mm/timestep) using the formula from Fitzpatrick, E.A. (1963), J. Appl. Meteorol., 780 as adapted by Sands, P. and Hughes, R.D. (1976), Agricultural Meteorology, 161. This algorithm is reasonable for timesteps of 7 days or greater, but should be used with extreme caution in daily timestep models. At that resolution, factors such as wind speed are very important in determining evaporation rates, and the estimates from the Fitzpatrick algorithm are likely to vary widely from actual values. Direct class A pan evaporation data should be used where available.

Input
variables

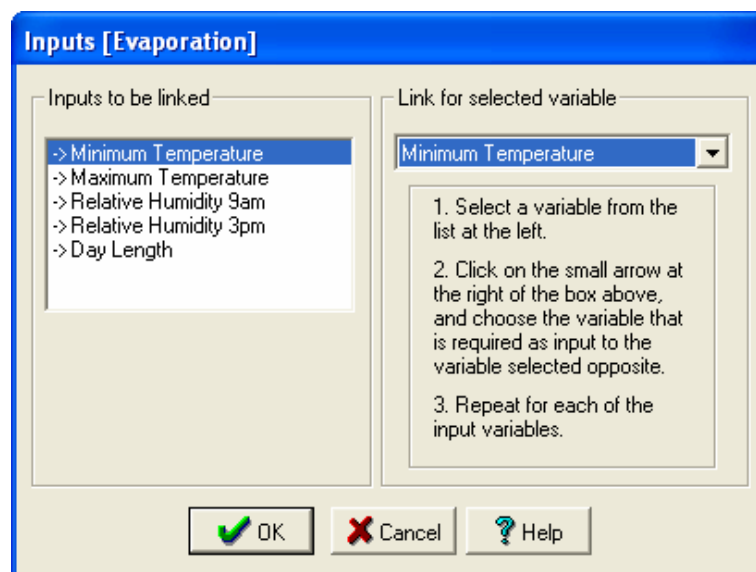
The input variables required and the modules that generally supply them are:

- **Minimum temperature** (Metbase module)
- **Maximum temperature** (Metbase module)
- **9am Relative Humidity** (Metbase module)
- **3pm Relative Humidity** (Metbase module)
- **Daylength** (Daylength module)

➤ **To complete the Evaporation module**

- 1.** Left click on the **Inputs** button.
- 2.** Link all input variables to the appropriate variables supplied by other modules using the drop down box.
- 3.** Go back to the **Evaporation** dialog box and left click on the **Outputs** button.
- 4.** Select the output variable by left-clicking on the **Select** button and rename using the **Rename** button if needed.

Fig. 9-11 The Evaporation module's input variable dialog box.



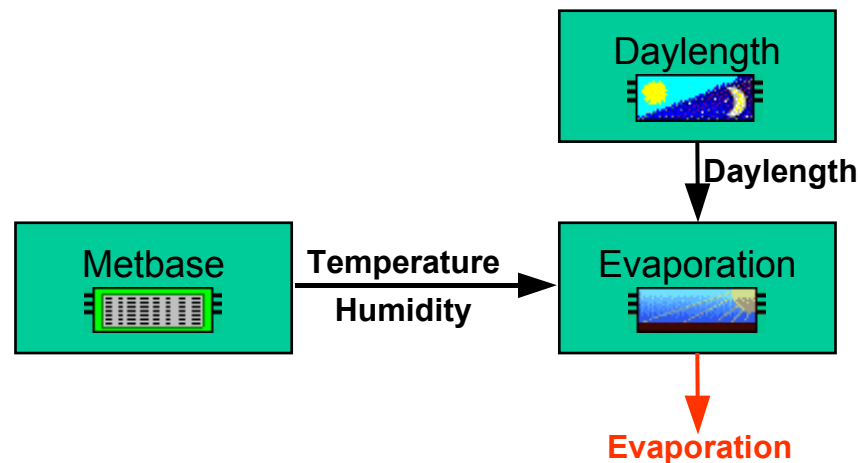
Output
variables

The module only outputs one variable: **Evaporation** (in mm/timestep).



The **Evaporation** module is most commonly used to supply evaporation values to the **Soil Moisture** module but could be used to drive any process, including mortality.

Fig. 9-12 Example Evaporation module diagram.

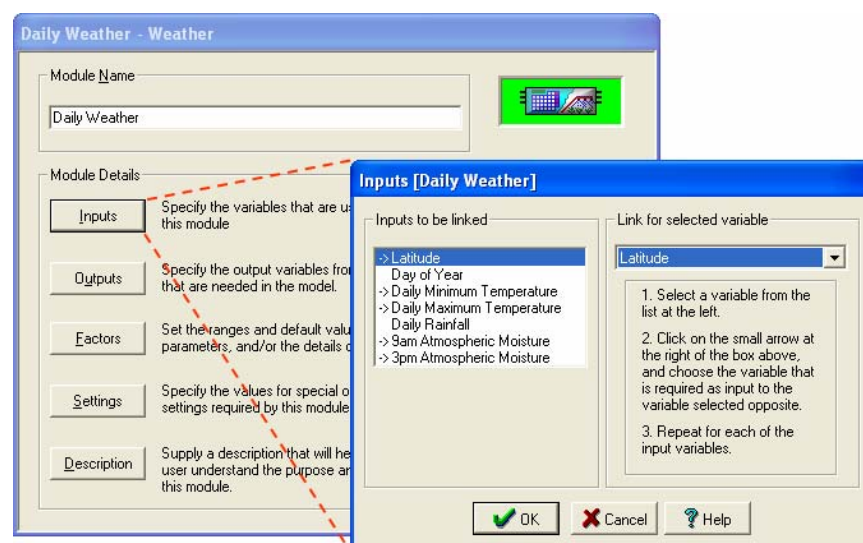


9.5 The Weather Module

What is the Weather module?

The Weather module is designed to generate output variables that describe several features of the daily weather. As such it can be used to replace several other modules in DYMEX (Circadian, Daylength and Evaporation). The module inputs values of latitude, the day of the year, temperature, rainfall and a measure of atmospheric moisture and outputs daylength, evaporation, daily temperature cycle and derived values of atmospheric moisture. In addition, the module allows the user to integrate the occurrence of preset values of temperature and RH to derive an “infection event” similar to that used by some modellers to predict the infection conditions for fungal pathogens (for example, Wang, Ryley and Meinke 2000).

Fig. 9-13 Weather module dialog box and its input dialog box.



Input variables

The module takes as its input the Latitude of the location being simulated, the Day of Year, the daily Minimum and Maximum Temperatures, daily Rainfall

and measures of atmospheric moisture at 9am and 3pm, respectively. The atmospheric moisture measures can be Relative Humidity (default), Vapour Pressure or Dewpoint. The user can specify which particular type of input is being provided in the “Settings” dialog. The rainfall and atmospheric moisture inputs are required only if evaporation or the “infection event” outputs are being used. All other inputs are mandatory.

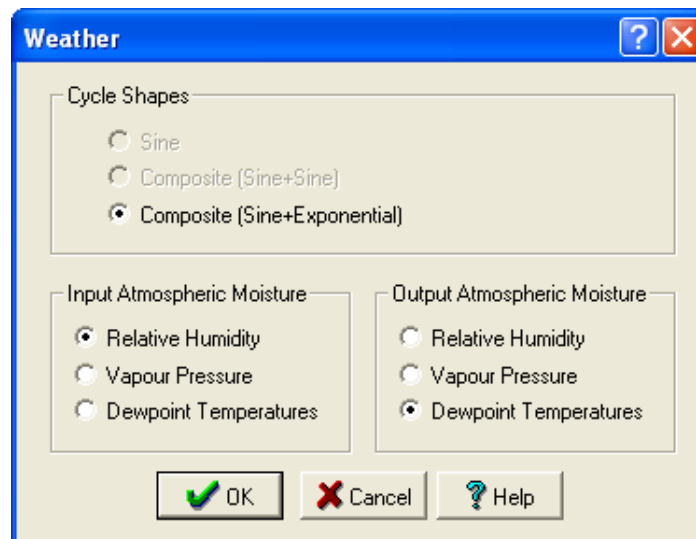
Settings

The “**Settings**” dialog allows the user to choose the particular cycle shape to be used for the daily cycle of temperature (output 1). A choice of **Sine**, **Composite (Sine+Sine)** and **Composite (Sine+Exponential)** are currently available. These are the same as the available shapes for the Circadian module, and a description of these is given in Section 9.1.

In the **Input Atmospheric Moisture** box, the user can choose the type of variable provided for inputs 6 and 7 of the module. The available choices are Relative Humidity (in %), Vapour Pressure (in millibars) and Dewpoint Temperature (in degrees Celsius).

In the **Output Atmospheric Moisture** box, the user can choose the type of variable that the module will provide for outputs 4 and 5 of the module. The available choices are again Relative Humidity (in %), Vapour Pressure (in millibars) and Dewpoint Temperature (in degrees Celsius). Thus these outputs can be used to convert from one measure of atmospheric moisture content to another.

Fig. 9-14 Weather module Setup dialog box.



Output variables

The following outputs are available from the Weather module:

1. **Daily Temperature Cycle** – This is the daily course of temperature, interpolated between minimum and maximum temperatures using the selected cycle shape (with 24 segments by default). The next day’s minimum temperature is used as the second minimum for all but the

Sine shape. This output variable is different to a normal output variable in that it has a number of values – one value for every segment (usually 24) over the period of a day. These values are obtained by interpolation, using one of 3 functions selected by the user. When this output is then used as input to a function, the function is evaluated for each of these values separately, *as if the model had a timestep of 1 hour*. When the output variable is used directly as input to another module, its average value over the 24 segments is used.

2. **Daylength** – time between sunrise and sunset (hours)
3. **Evaporation** – class A Pan Evaporation (method of Fitzpatrick), in mm/timestep.
4. **9am Atmospheric Moisture** – the atmospheric moisture at 9am, expressed as selected in the “Settings” dialog.
5. **3pm Atmospheric Moisture** – the atmospheric moisture at 3pm, expressed as selected in the “Settings” dialog.
6. **Night hours at 100% RH** – the number of hours during the previous night (i.e., the night that terminates in the current day’s minimum temperature) in which the Relative Humidity was 100%. This is, therefore, the number of hours in which the temperature was below the dewpoint.
7. **Night hours above 80% RH** – the number of hours during the previous night (i.e., the night that terminates in the current day’s minimum temperature) in which the Relative Humidity was above 80%. The method used to calculate this is taken from Wang, *et al.* (2000).
8. **Germination Event Flag** – This output is 1 if, during the previous night: (i) The temperature stays between 10° and 33° Celsius, (ii) there is at least 4 hours at 100% RH, and (iii) there is at least 10 hours above 80% RH

➤ **To complete the Weather module**

1. Left click the **Inputs** button.
2. Link **Latitude**, **Day of Year**, **Daily Minimum Temperature** & **Daily Maximum Temperature** by left-clicking on the variables and using the drop down menu to find the appropriate variable. Link the **Rainfall** and **Atmospheric Moisture** inputs to their appropriate variables if they are needed.
3. Go back to the **Weather** dialog box and left click on the **Outputs** button.
4. Select the **Daily Temperature Cycle** variable and rename it if desired.
5. Select any of the other output variables that are required, and rename to give meaningful names.
6. Go back to the **Weather** dialog box and left click on the **Settings** button.

- 7.** Choose the appropriate **Cycle Shapes** (e.g. Sine), and, if input or output atmospheric moisture variables have been selected in steps 2 and 5 above, select the correct type of variable from the given choices.
- 8.** Go back to the Weather dialog box and click on the Factors button.
- 9.** Choose appropriate ranges and default values for the 3 factors (you may rename them if desired).



The **Weather** module can be used to replace the Daylength, Evaporation and Circadian module in a DYMEX model by a single module.

9.6 The Soil Moisture (1-layer) Module

What is the soil moisture module?

The **Soil Moisture** module is a sub-model that simulates the water balance in a single layer of soil. It outputs an index between 0 and 1, with 0 being dry soil and 1 being a saturated soil. The input variables are **Rainfall** and **Pan Evaporation** (both in mm). Note that the Soil Moisture module can be set to use the model's sub-population structure, thus allowing simulations of (say) multiple fields with differing soil characteristics.

The soil storage capacity is given by a parameter, C (**Soil Moisture Capacity**), while a second parameter, r (**Evapotranspiration Rate**) gives the soil water loss due to evaporation and transpiration when the soil is saturated as a proportion of pan evaporation. A third parameter, E_0 (**Basal Evaporation**) allows the user to modify the way that water loss is handled (see equation). Then, if S_i is the current water content of the soil (as a proportion of the capacity) and E_i is the pan evaporation, the proportion of water lost by evapotranspiration during a timestep, W_i , is

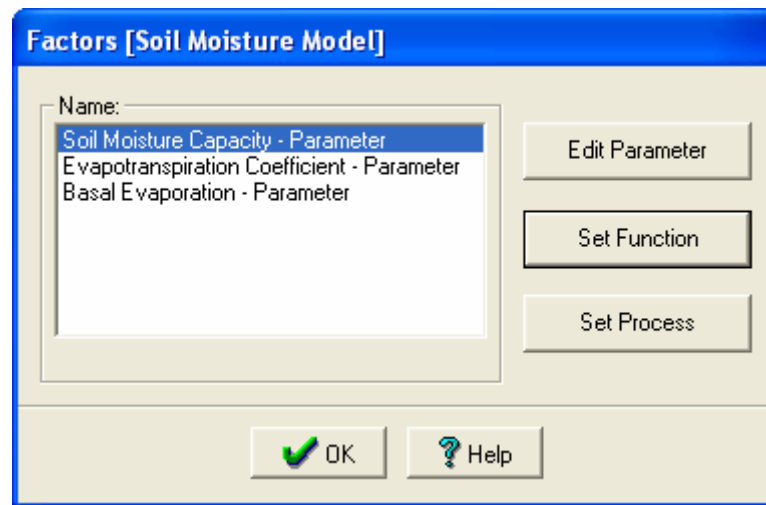
$$W_i = \min(1, \max(0, r(E_0 + S_i(E_i - E_0))/C))$$

When E_0 is set to 0, the effective evapotranspiration rate reduces linearly with the current soil moisture store (it equals rS_i). This follows logically from the observation that the energy required to extract moisture from the soil increases as the moisture level decreases and is a satisfactory scenario for many requirements. In situations where moisture is extracted more rapidly (for example, when simulating only a narrow soil profile at the surface), the E_0 parameter can be set to a small positive value to allow water to be drawn from the soil at a higher rate. Note that E_0 is expressed in the same units as Evaporation and Rainfall.

Finally, the new water store, S_{i+1} , is given by:

$$S_{i+1} = \min(1, S_i - W_i + R_i/C)$$

Where R_i is the rainfall for the current timestep.

Fig. 9-15 Soil Moisture (1-layer) module's Factor dialog box.

Factors Three Factors have to be specified by the modeller:

- **Soil Moisture Capacity** (permitted range: 50-200 mm): The water holding capacity of the defined layer of soil (say, 1 metre or to required rooting depth), in mm. Clay soils will have larger values than sandy soils. Sandy soils may store only 50 mm while loams may hold 150 mm and clays 200 mm within the rooting depth of most crops.
- **Evapotranspiration Coefficient** (permitted range: 0.5-1.2): Sum of evaporation and transpiration from vegetation expressed as a proportion of open Pan Evaporation. Generally between 0.5 and 1.2 with 0.8 typical.
- **Basal Evaporation** (permitted range: 0-5 mm): The portion of the evaporation input that is available for evapotranspiration regardless of soil moisture status (in mm). Values above 0 will empty water out of a non-saturated soil at higher rates than if it is set to the default of 0. Generally set to 0.

As usual, any of the parameters can be replaced by a function or process. For example, in many situations, the evapotranspiration rate may not be satisfactorily simulated using a fixed value parameter. Satisfactory fits to measured data have been derived using a decreasing evapotranspiration rate as the soil moisture level drops below the wilting point.

➤ **To complete the Soil Moisture module**

1. Left click on the **Inputs** button and link **Rainfall** and **Evaporation** to the appropriate variables supplied by other modules.
2. Go back to the **Soil Moisture** dialog box and left click on the **Outputs** button.
3. Select the output variable using the **Select** button and rename it if required, using the **Rename** button.

4. Go back to the **Soil Moisture** dialog box and left click on the **Factors** button.
5. Set the parameter or function that determines each of the soil moisture parameters.

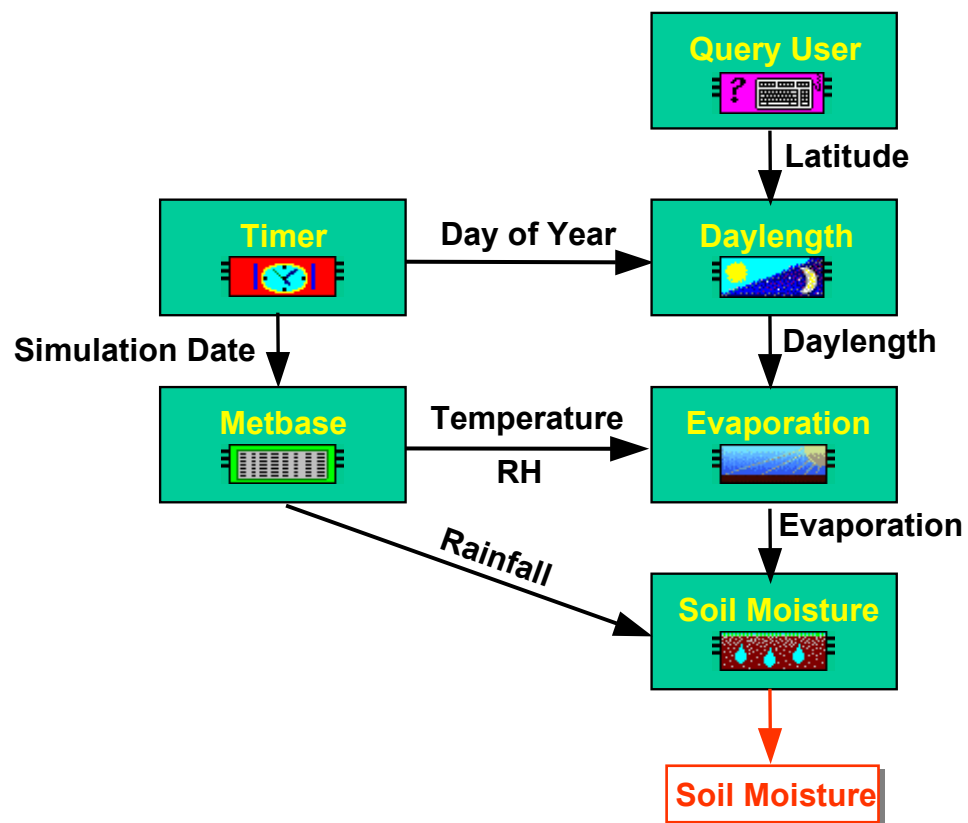


Note that all the variables set within the Factor dialog box have descriptions already provided for them, with the descriptions including the range of suitable values.



Soil moisture is a more useful measure of available moisture than rainfall. It can be used to determine plant growth, the survival of a lifestage of an invertebrate, etc.

Fig. 9-16 Example Soil Moisture Module Diagram.



9.7 The Degree-Day module

The **Degree-day** module has a single input (the daily temperature cycle) and a parameter that supplies a threshold temperature. The daily temperature cycle must be supplied by a **Circadian** (see Section 9.1) module. The output from the module is the number of degree-days above that temperature (i.e., the area bounded by the daily temperature and the threshold temperature when the daily temperature is above the threshold) that is accumulated during a timestep. A factor is used to supply the Threshold Temperature. This factor will normally

be a parameter, but a function or process may be used in those rare cases where the threshold temperature changes with time.

➤ **To complete the Degree-day module**

- 1.** Left click on the **Inputs** button and within **Inputs** dialog box link the **Daily Temperature Cycle** to the required daily cycle variable.
- 2.** Go back to the module dialog box and left click on the **Outputs** button.
- 3.** Select the **Day-degrees above Threshold** variable by left-clicking on the **Select** button, and rename if required.
- 4.** Return to the module dialog and click on the **Factors** button. Set the factor as required (usually this will be a parameter).

9.8 The Climate Change Scenario module

The **Climate Change Scenario** module takes as its inputs variables that represent current weather conditions (temperature, rainfall and evaporation) and outputs corresponding variables for weather conditions given a particular *climate change scenario*. The scenario is provided as a set of parameter values. Various simple scenarios are possible using this module. For example, maximum and minimum temperatures under a greenhouse scenario can be altered as an overall temperature change and/or as a change in temperature per degree latitude. Rainfall can also be altered as a change in percentage winter and summer rainfall and/or as a change in rainfall per degree of latitude. In addition, Evaporation may also be adjusted using a simple algorithm.

Up to 6 input variables are used by this module, as below.

<i>Day of Year</i>		Number of days elapsed since December 31
<i>Current Minimum Temperature</i>	I_{min}	The minimum daily temperature (current climate)
<i>Current Maximum Temperature</i>	I_{max}	The maximum daily temperature (current climate)
<i>Current Rainfall</i>	I_{rain}	Total Rainfall (current climate)
<i>Current Evaporation</i>	I_{evap}	Total Evaporation (current climate)
<i>Latitude</i>	l	Latitude of location being simulated

Day of Year is generally obtained from the **Timer** module. The temperatures are the daily averages over the model timestep, while the rainfall and evaporation variables are the total for the timestep. The evaporation input is required only if the evaporation output is selected for the module.

Four output variables are available from the module, these being the minimum daily temperature (O_{min}), maximum daily temperature (O_{max}), total rainfall (O_{rain}) and total evaporation (O_{evap}) to be expected under the specified scenario.

The Climate Change Scenario is specified using 18 factors, as listed in Table 9-2.

Table 9-2 Climate Change Scenario factors.

Min. Temperature Change (Winter)	W_{min}	The change in the minimum temperature that occurs during the ' <u>Winter</u> ' months <i>outside</i> the Equatorial zone.
Min. Temperature Latitudinal Change (Winter)	WL_{min}	The change in the minimum temperature for each degree of latitude away from the equator that occurs during the ' <u>Winter</u> ' months <i>outside</i> the Equatorial zone.
Min. Temperature Change (Summer)	S_{min}	The change in the minimum temperature that occurs during the ' <u>Summer</u> ' months <i>outside</i> the Equatorial zone.
Min. Temperature Latitudinal Change (Summer)	SL_{min}	The change in the minimum temperature for each degree of latitude away from the equator that occurs during the ' <u>Summer</u> ' months <i>outside</i> the Equatorial zone.
Min. Temperature Change (Equatorial Zone)	E_{min}	The change in the minimum temperature that occurs <i>inside</i> the Equatorial zone.
Max. Temperature Change (Winter)	W_{max}	The change in the maximum temperature that occurs during the ' <u>Winter</u> ' months <i>outside</i> the Equatorial zone.
Max. Temperature Latitudinal Change (Winter)	WL_{max}	The change in the maximum temperature for each degree of latitude away from the equator that occurs during the ' <u>Winter</u> ' months <i>outside</i> the Equatorial zone.
Max. Temperature Change (Summer)	S_{max}	The change in the maximum temperature that occurs during the ' <u>Summer</u> ' months <i>outside</i> the Equatorial zone.
Max. Temperature Latitudinal Change (Summer)	SL_{max}	The change in the maximum temperature for each degree of latitude away from the equator that occurs during the ' <u>Summer</u> ' months <i>outside</i> the Equatorial zone.
Max. Temperature Change (Equatorial Zone)	E_{max}	The change in the maximum temperature that occurs <i>inside</i> the Equatorial zone.
Rainfall Change (Winter)	W_{rain}	The change in the rainfall (in %) that occurs during the ' <u>Winter</u> ' months <i>outside</i> the Equatorial zone.
Rainfall Latitudinal Change (Winter)	WL_{rain}	The change in the rainfall (in % per degree latitude) for each degree of latitude away from the equator that occurs during the ' <u>Winter</u> ' months <i>outside</i> the Equatorial zone.
Rainfall Change (Summer)	S_{rain}	The change in the rainfall (in %) that occurs during the ' <u>Summer</u> ' months <i>outside</i> the Equatorial zone.

Rainfall Latitudinal Change (Summer)	SL_{rain}	The change in the rainfall (in % per degree latitude) for each degree of latitude away from the equator that occurs during the ‘ <u>Summer</u> ’ months <i>outside</i> the Equatorial zone.
Rainfall Change (Equatorial Zone)	E_{rain}	The change in the rainfall (in %) that occurs <i>inside</i> the Equatorial zone.
Evaporation Change (Winter)	W_{evap}	The change in the evaporation that occurs during the ‘ <u>Winter</u> ’ months <i>outside</i> the Equatorial zone.
Evaporation Change (Summer)	S_{evap}	The change in the evaporation that occurs during the ‘ <u>Summer</u> ’ months <i>outside</i> the Equatorial zone.
Evaporation Change (Equatorial Zone)	E_{evap}	The change in the evaporation that occurs <i>inside</i> the Equatorial zone.

The **Equatorial Zone** (z) is a band of latitude centred on the equator and its exact extent is defined in the **Simulator**. “Winter” and “Summer” seasons are defined with reference to the Equatorial Zone and the **Day Of Year** input. North of the Equatorial Zone, “Summer” is defined as that period when Day of Year is between 92 and 274 (i.e., March 2 – September 30, inclusive), with “Winter” being the rest of the year. South of the Equatorial Zone, these are reversed.

The temperature output variables are derived from the input variables and factors as follows (the “ t ” subscript can be either “ $tmin$ ” or “ $tmax$ ”):

$$\text{in the } \mathbf{Equatorial\ Zone}, \quad O_t = I_t + E_t$$

$$\text{elsewhere, in “Winter”,} \quad O_t = I_t + W_t + WL_t \times (abs(l) - z / 2)$$

$$\text{elsewhere, in “Summer”,} \quad O_t = I_t + S_t + SL_t \times (abs(l) - z / 2)$$

The rainfall output variables are derived from the input variables as follows:

$$\text{in the } \mathbf{Equatorial\ Zone}, \quad O_{rain} = I_{rain} \times \left(\frac{E_{rain}}{100} \right)$$

elsewhere, in “Winter”,

$$O_{rain} = I_{rain} \times \left(1 + \frac{W_{rain} + WL_{rain} \times (abs(l) - z / 2)}{100} \right)$$

elsewhere, in “Summer”,

$$O_{rain} = I_{rain} \times \left(1 + \frac{S_{rain} + SL_{rain} \times (abs(l) - z / 2)}{100} \right)$$

The Evaporation output is calculated as follows

$$\text{in the } \textit{Equatorial Zone}, \quad O_{evap} = I_{evap} \times \left(\frac{E_{evap}}{100} \times \Delta t \right)$$

$$\text{elsewhere, in "Winter"}, \quad O_{evap} = I_{evap} \times \left(\frac{W_{evap}}{100} \times \Delta t \right)$$

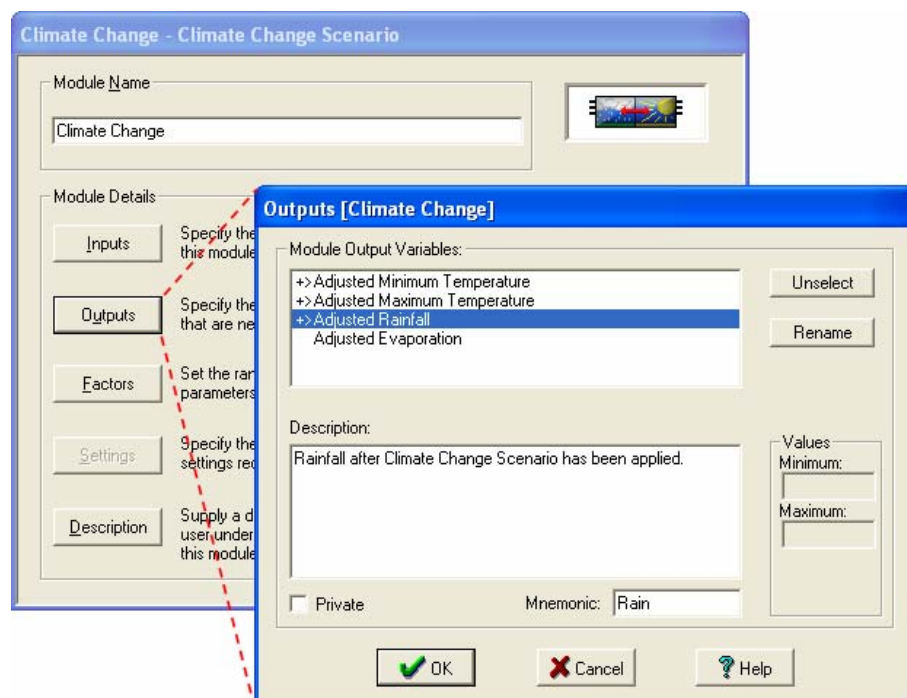
$$\text{elsewhere, in "Summer"}, \quad O_{evap} = I_{evap} \times \left(\frac{S_{evap}}{100} \times \Delta t \right),$$

where, Δt is the change in average temperature produced by the scenario (i.e., the evaporation change is dependent on the temperature change).

➤ **To complete the Climate Change Scenario module**

1. Left click on the **Inputs** button and within **Inputs** dialog box link the **Day of Year**. Link each of the other variables in turn (note, however, that the **Current Evaporation** input needs to be linked only if the **Evaporation** output is required).
2. Go back to the module dialog box and left click on the **Outputs** button. Select and rename any of the output variables that are required (Fig. 9-17). Return to the module dialog and click on the **Factors** button. Set the factors as required (usually these will be parameters, but more complex scenarios could be constructed by using functions or processes for some of the factors).

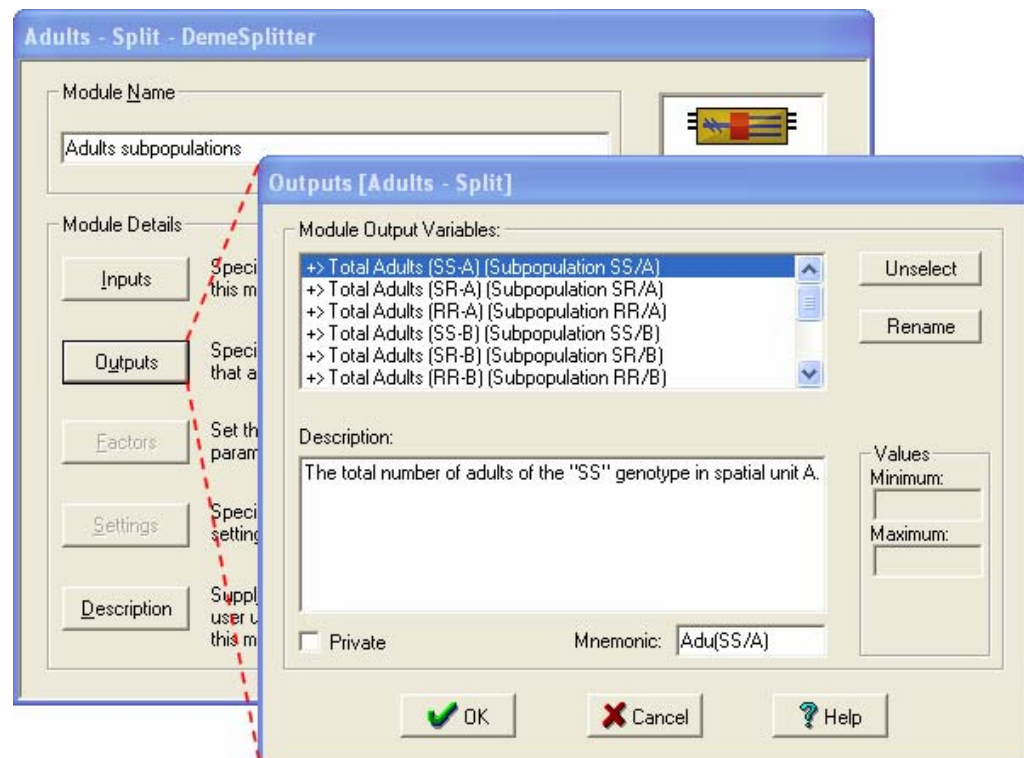
Fig. 9-17 The Climate Change Scenario module output dialog.



9.9 The DemeSplitter and DemeStatistics modules

These two modules are only available in models that use sub-populations. Their purpose is to take *demed variables* (see Section 2.1) as input and split the variables into normal (non-demed) variables. The **DemeSplitter** module takes a single demed variable as its input. There will be as many outputs as there are sub-populations. For example, if the model population consists of 4 spatial sub-populations (A, B, C and D), each divided into 3 genotypes (SS, SR and RR), there would be a total of 12 output variables (“SS/A”, “SR/A”, etc), these being the components of the input variable (Fig. 9-18).

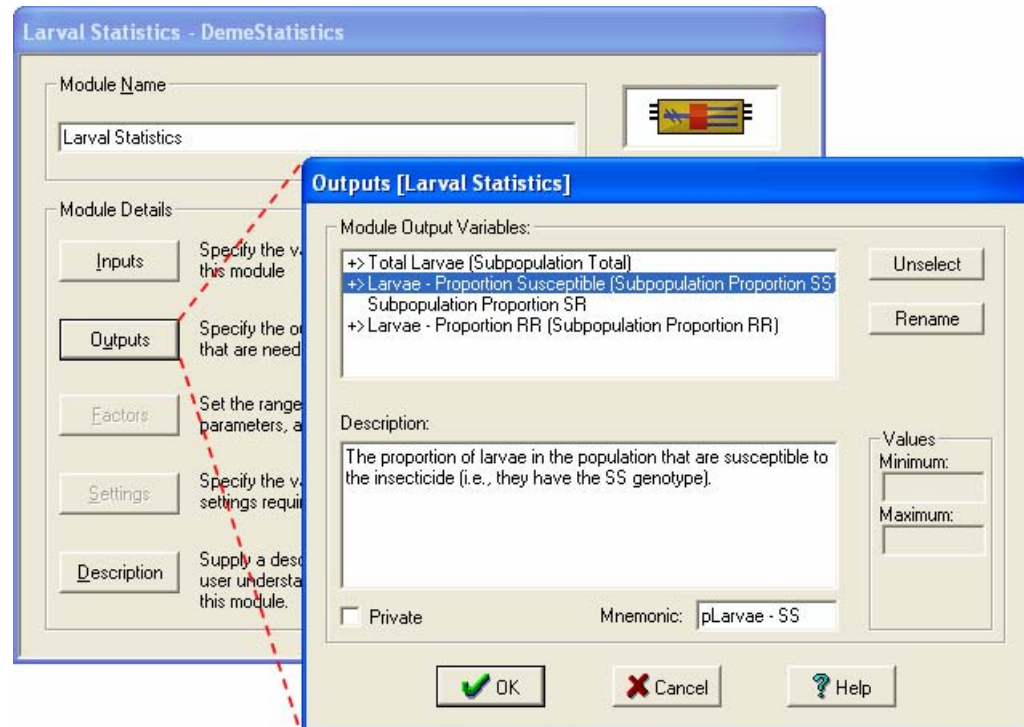
Fig. 9-18 The DemeSplitter module output dialog, showing subpopulation components for a model that uses both spatial and genetic subpopulations.



The **DemeStatistics** module is similar to the **DemeSplitter** module, but has a different set of output variables. If the model population is split into N sub-populations, the **DemeStatistics** module will have $N+1$ outputs. The first output variable is the total of the input variable (summed over all sub-populations). The other output variables each represent the proportion of the total that is represented by a particular subpopulation. The example shown in Fig. 9-19, which relates to a model with 3 genetic sub-populations (“SS”, “SR” and “RR”). The second output variable (highlighted) is the proportion of “Larvae” (the demed input variable) with the “SS” genotype. Note that in the example, only three of the outputs that are available are actually used in the model.

Setting up a **DemeSplitter** or **DemeStatistics** module is very straightforward. The user just needs to select the input variable and then select, rename and provide a description, etc. for each of the required output variables.

Fig. 9-19 The DemeStatistics module output dialog, showing subpopulation totals and proportions for a model that uses genetic subpopulations.



10. Function Templates

Mathematical functions are used throughout DYMEX to specify relationships between variables. The terminology surrounding functions in DYMEX can be confusing, and a few definitions here may help in removing some of the confusion.

Function Template. A mathematical definition of a function's shape, providing its equation and the symbolic parameters, but not providing values for the parameters.

Function. This will be used to refer to the actual use of a Function Template in a particular place. As such, parameter values (defaults and/or ranges) are provided.

A number of commonly used Function Templates are provided with DYMEX. Full descriptions of each of these functions can be found in the DYMEX Help system.

10.1 Adding new Templates

If none of the predefined templates are adequate for an application, a new template can be defined. These templates can then be used anywhere in the model in the same way as a built-in Function Template. A template can have up to 9 parameters and use any of the syntax described in Section 10.2.

Assume we need a new Function Template with the following equation:

$$y = \frac{p_1}{1 + e^{-p_2 x^2}}$$

In the equation, x is the independent (driving) variable, and p₁ and p₂ are parameters

➤ To add this template to the list of Function Templates

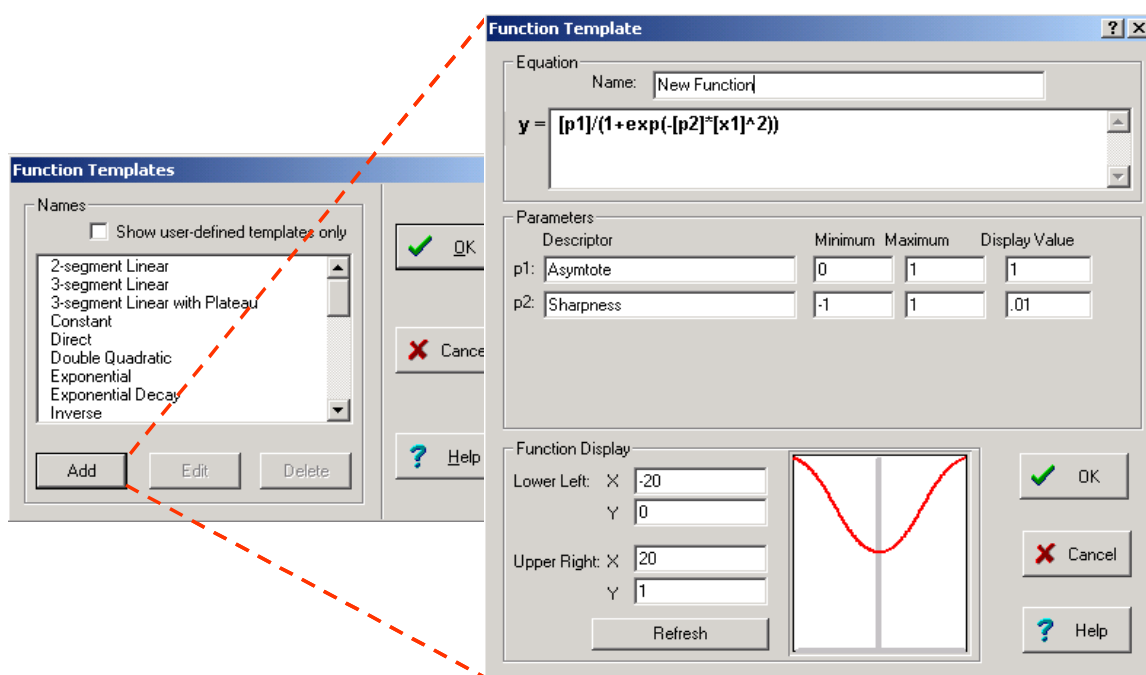
- 1.** Make sure the **Model Components** window is active, and click on the **Model|Function Library...** menu item. This opens the Function Templates dialog, listing all currently defined Function Templates (Figure 10-1).
- 2.** Click on the **Add** button to open the Function Template definition dialog.
- 3.** Click on the text box labelled “y=”, and type in the equation as follows:

$$[p1]/(1+\exp(-[p2]*[x1]^2))$$

This syntax is explained in detail in the next section, but note that the parameters and driving variable are within square parentheses and labelled sequentially.

4. Click on the **Refresh** button, and two lines of text boxes will appear in the dialog for entry of parameter details.
5. For the first parameter (**p1**), insert Asymptote as the **Name**, and set the **Minimum**, **Maximum** and **Display Value**, to 0, 1 and 1, respectively. These limits define the absolute limits for this parameter in any use of the Function Template in the model. The Display Value is used as the parameter value when the function shape is displayed throughout DYMEEX.
6. For the second parameter (**p2**), insert Sharpness Parameter as the **Name**, and set the **Minimum**, **Maximum** and **Display Value**, to -1, 1 and 0.01, respectively. (Note that if more than 5 parameters are used in a Function Template, a button will appear in the dialog that allows switching between groups of parameters).
7. In the **Function Display** panel, set the **Lower Left** X and Y values to -20 and 0, respectively, and the **Upper Right** X and Y values to 20 and 1, respectively. This has the effect of setting the display coordinates to be used in the graphic to the right of the panel, and for all such display graphics in DYMEEX when this template is illustrated.
8. Click on the Refresh button to see the new Function Template displayed.
9. Supply a name for the Function Template by clicking on the **Name** text box, and typing in an appropriate name.

Figure 10-1 Adding a new Function Template to the Function Library



10.2 Function Template Syntax

The name and equation defining a Function Template must be provided using the syntax outlined in this section.

Template Name



A unique and descriptive name should be chosen for the Function Template. The name may be the same as the name of one of the pre-defined Function Templates, in which case the pre-defined template will be replaced by the new template.

Equation Syntax

The equation definition may contain the operators and special symbols listed in Table 10-2.

When the equation is evaluated, the $[x1]$ token is replaced by the current value of the independent (driving) variable, while $[p1]$ is replaced by the value of the first parameter, $[p2]$ by the second, and so on. Note that $[p1]$ is not necessarily the first parameter in the expression – it refers to the first parameter in the functions parameter list. Both the $[x1]$ and any parameter token may appear in an expression more than once. Thus, $5+[p1]*[x1]+[p2]*[x1]^2$ is a valid expression. Spaces are allowed anywhere within an expression, and should be used to enhance readability. Alphabetic characters within the expression may be in either upper- or lower-case.



Operators have a precedence order, with evaluation proceeding in the following sequence (operators within the same order level are evaluated from left to right):

()

abs, exp, log, log10, sin, cos, tan, asin, acos, atan, sqrt, max, min, if, ife, not, in, rt, crt, rand, rang, try

^

* /

> < <= >= !=

& |

+ -

Some examples of valid expressions, and their corresponding equations are given in Table 10-1:

Table 10-1 Examples of valid Function Template expressions and their corresponding evaluations.

$[p1] + [p2]*[x1] + [p3]*[x1]^2$	$y = p_1x + p_2x + p_3x^2$
$1 - \exp(-[p1]*[x1])$	$y = 1 - e^{-p_1x}$
$[p1]/(1-[p2]*[x1]^2)$	$y = \frac{p_1}{1 - p_2x^2}$
$[x1] + 10 - [p2]*(\sin(\pi*[p1]*[x1]))$	$y = x + 10 - p_2(\sin(\pi p_1x))$
$([x1]>[p1])*([p2]*([x1]-[p1]))$	$y = \begin{cases} p_2(x - p_1), & \text{for } x > p_1 \\ 0, & \text{for } x \leq p_1 \end{cases}$
$\max(0, [p1] + [p2]*[x1]^2)$	$y = p_1 + p_2x^2$, or 0, whichever is the larger
$\text{if}([x1]>-1 \ \& \ [x1]<1, [x1]*[p1])$	$y = \begin{cases} p_1x, & \text{for } x > -1 \text{ and } x < 1 \\ 0, & \text{elsewhere} \end{cases}$
$\text{ife}([x1]<[p1], [p2], [p3])$	$y = \begin{cases} p_2, & \text{for } x < p_1 \\ p_3, & \text{for } x \geq p_1 \end{cases}$

Table 10-2 Available syntax items for a Function Template equation

Operator/Symbol	Action performed or value substituted
n	Any real number, for example, 5.75
pi	3.14159.... (i.e, the value of π)
e	2.71828.... (ie, the value of e)
[x1]	The independent (driving) variable in the equation
[pn]	The n-th parameter in the equation (eg, [p2])
+	addition, eg [x1]+12
-	subtraction (eg, 7.2-[p1]), or negation (eg, -[x1])
*	multiplication (eg, [p1]*[x1])
/	division (eg, [x1]/2)
^	power (eg, [x1]^2 is the same as [x1]*[x1])
&	the “and” operation; evaluates to 1 if and only if both sides of operator are positive, evaluates to 0 otherwise
	the “or” operation; evaluates to 1 if either side of the operator is positive, evaluates to 0 otherwise
>	“greater than” (eg, [x1]>5); evaluates to 1 if true, 0 if false
>=	“greater than or equal” (eg, [x1]>=[p1]); evaluates to 1 if true, 0 if false
<	“less than” (eg, [x1]<[p1]); evaluates to 1 if true, 0 if false
<=	“less than or equal” (eg, [x1]<=0); evaluates to 1 if true, 0 if false
=	“equal” (eg, [x1]=0); evaluates to 1 if true, 0 if false
!=	“not equal” (eg, [x1]!=0); evaluates to 1 if true, 0 if false
abs(t)	the absolute value (or magnitude) of expression t
exp(t)	exponential function, i.e, e^t (eg, $\exp(2-[p1]*[x1])$)
log(t)	logarithm with the base ‘e’ of t (eg, $\log(2.4*[x1])$)
log10(t)	logarithm with the base 10 of t
sin(t)	sine of t (t is in radians)
cos(t)	cosine of t (t is in radians)
tan(t)	tangent of t (t is in radians)
asin(t)	arcsine of t (in radians)
acos(t)	arccosine of t (in radians)
atan(t)	arctangent of t (in radians)

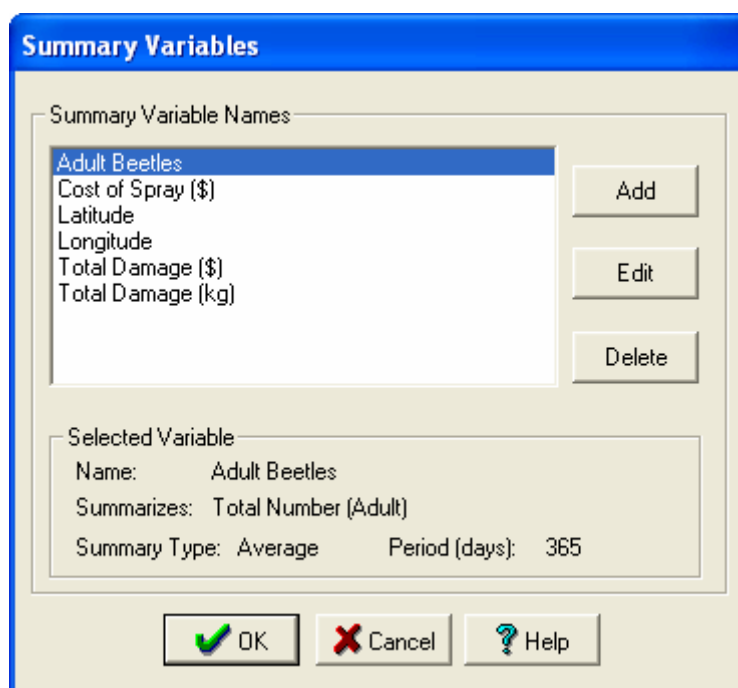
<code>sqrt(t)</code>	square root of t
<code>min(t₁,t₂)</code>	the smaller (minimum) of the two expressions, t_1 and t_2
<code>max(t₁,t₂)</code>	the larger (maximum) of the two expressions, t_1 and t_2
<code>mod(t₁,t₂)</code>	the fractional part of t_1/t_2
<code>if(c, t)</code>	evaluates to t if c is positive, or 0 if c is less than or equal to 0. This operator can be used to specify “segmented” functions.
<code>ife(c, t₁, t₂)</code>	evaluates to t_1 if c is positive, or t_2 if c is less than or equal to 0. This operator can be used to specify “segmented” functions.
<code>in(r₁, r₂, t)</code>	evaluates to 1 if $r_1 < t \leq r_2$, 0 otherwise
<code>not(t)</code>	evaluates to 0 if the expression in parentheses (t) is positive, 1 otherwise
<code>rt(t, p, r)</code>	The “running total” of t over the previous p timesteps (the total is restarted when $r \leq 0$) NOTE: DO NOT USE THIS FUNCTION IN COMBINATION RULES
<code>cnt(t, c, r)</code>	The “running total” of t , accumulated while $c > 0$ (the total is restarted when $r \leq 0$) NOTE: DO NOT USE THIS FUNCTION IN COMBINATION RULES
<code>rand(t)</code>	A uniformly distributed random number (n) in the range $0 \leq n < 1$. If $t < 0$, the generator is reseeded
<code>rang(t₁, t₂)</code>	A normally (Gaussian) distributed random number, where the distribution it is drawn from has a mean of t_1 and a variance of t_2
<code>try(t₁,t₂)</code>	Evaluates to t_1 if the value of t_1 is valid, otherwise it evaluates to t_2
<code>()</code>	parentheses – these may be used where required to force precedence

11. Summary Variables

Summary Variables (or **Run Summaries**) are variables that summarize the values of normal DYMEEX variables in some way over the course of a simulation run. They enable the results of different runs to be compared rapidly without the need to examine detailed tables or graphs. They are especially useful in models that will be used to run Sequences (see *Simulator User's Guide*).

To create, edit or delete a Summary Variable, select **Summary Variables...** from the **Model** menu. This will display the Summary Variables dialog box (Fig. 11-1). The dialog lists all the currently defined Summary Variables in the window at top left. Clicking on a variable name displays that variable's properties in the lower part of the dialog.

Fig. 11-1 A Summary Variables dialog box showing two summary variables.



➤ To create a new Summary Variable

- 1.** Click on the Add button in the Summary Variables dialog box to go to the Edit Summary Variable dialog box (Fig. 11-2).
- 2.** Click on the small button at the right of the **Variable to summarize** window and select the required variable from the drop-down list.
- 3.** Provide a name for the variable in the **Name** field. Note that Summary Variable names must be unique within themselves, but a Summary Variable may have the same name as a standard variable.
- 4.** Select the type of summary required by clicking on the appropriate selection in the **Summary Variable Statistic** box.

5. Select the period over which the variable is to be summarized (**Summary Period**). Note that this selection is not available if “*Last*” has been chosen as the statistic in the previous step.
6. A **Description**, **Mnemonic** and **Default Precision** may optionally be provided, as for standard output variables.
7. Click on Ok to complete the Summary Variable specification.

Fig. 11-2 The Edit Summary Variables dialog box variables.

Summary Variable Statistic The **Summary Variables Statistic** box lists a choice of 5 ways in which a variable can be summarized to give a single output value for the Summary Variable. The summary is taken for the period defined by **Summary Period**.

- **Total** - is the total of the variable’s values over the selected period.
- **Average** - is the average of the variable’s values over the selected period.
- **Minimum** - is the minimum value of the variable during the selected period.
- **Maximum** - is the maximum value of the variable during the selected period.
- **Last Value** - is the last value the variable attains during the simulation. Note that the period is irrelevant for this statistic.

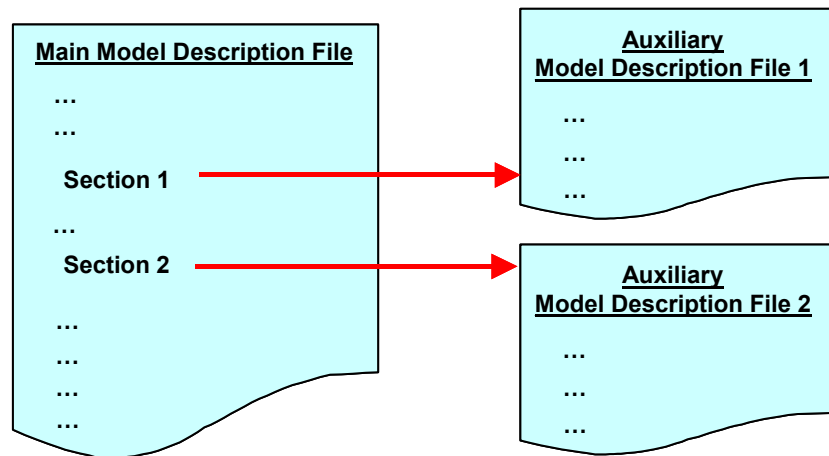
Summary Variables may be manipulated using modules (*Summary Modules*) in the same way as standard variables.

12. Working with Model Description Files

As briefly indicated in Section 1.2, the model that is built using the DYMEX **Builder** is stored in a file with extension “.gmd” – the *Model Description File*. For many simple models, a single file that stores all model components is quite adequate. However, when building complex modules it can be useful to break the model up into several smaller physical file units, each containing one or more modules. These units are termed model *Sections*, and the files they reside in are termed *Auxiliary Model Description Files* (generally given the extension “.gmi”). An example where this is useful may be with the modules used in conjunction with meteorological data. Several models may include use **MetBase**, **Circadian**, **Daylength**, **Evaporation** and **Soil Moisture** modules in the same way. These could be separated out as a section, to be included in all the models that use it. Thus any changes made to this group of modules would automatically apply to all the models. Another example would be multi-species models, where each species could be a separate section.

Figure 12-1 shows a schematic file layout of model that is organized into a main model “gmd” file, and two Sections, each stored in an Auxiliary “gmi” file. When either the Builder or the Simulator read the model, reading starts with the main “gmd” file. When a reference to “Section 1” is encountered, reading continues at the beginning of Auxiliary File 1. When the end of this file is reached, reading continues with the main file, and so on.

Figure 12-1 A model can consist of several Sections, each of which is stored in a separate Model Description File.



There are several caveats to the use of sections in *Auxiliary Model Description Files*:

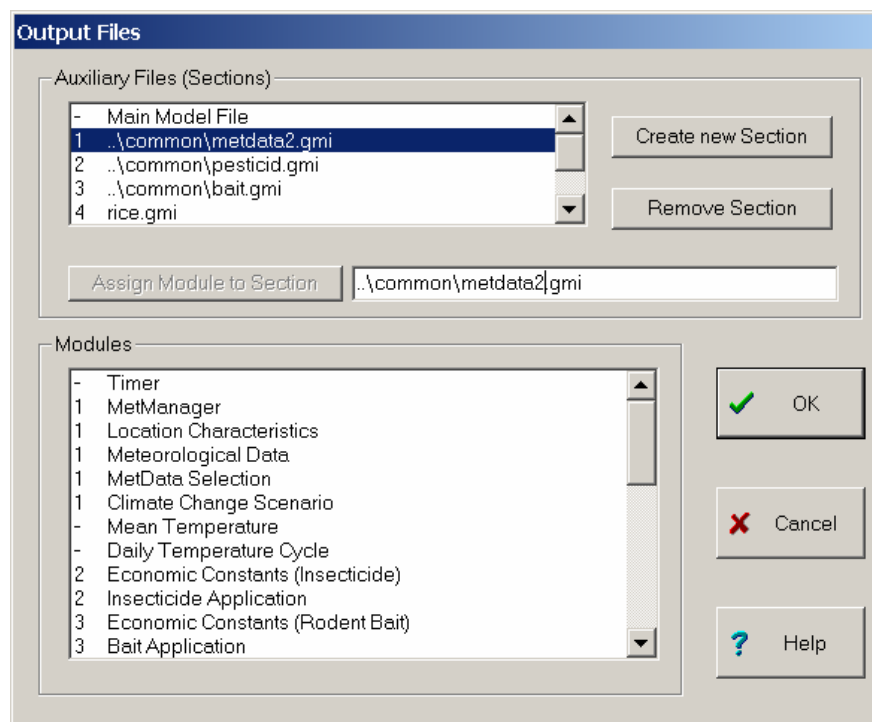
1. Sections may not be nested, i.e., an *Auxiliary File* may not include a reference to a section. The corollary to this is that the modules in an *Auxiliary File* must be contiguous.

2. Sections are read as if they were part of the main *Model Description File*. There is no translation or mapping of variable or module names at section boundaries. Therefore, output variable, module and function template names must be unique across all the files constituting a model. Input variable names in a section that do not have the matching output name elsewhere in the model will be considered as not yet set.
3. When a model is saved, DYMEX saves the *Main Model Description File* as well as any *Auxiliary Files* that have been changed. This could potentially cause another model using this Auxiliary File to fail to load. To avoid this problem, careful thought needs to be given to changes made to modules in *Auxiliary Files* (and they should always be backed up as a precaution).
4. All **Function Templates** that will be used by an *Auxiliary File* must be defined in the main file before the Auxiliary File can be imported.

➤ **To create a Section from a set of contiguous modules**

- 1.** Make sure the **Model Components** window is active, and click on the **Model|Files...** menu item. This opens the Output Files dialog, listing all modules currently in the model and their arrangement into files (Figure 12-2).
- 2.** Click on the **Create new Section** button to open the Auxiliary File dialog, and type in or browse for a filename.
- 3.** Click on “Save” to return to the Output Files dialog. The new Auxiliary File will be listed in the Auxiliary Files window, and selected.
- 4.** In the Modules window, click on one of the modules that is to go into this file. This module must currently reside in the Main Model Description File (not one of the Auxiliary Files).
- 5.** Click on the **Assign Module to Section** button (which will now be enabled) to move the module into the Section.
- 6.** If required, adjacent modules that are currently in the main file may be added to the same Section.
- 7.** Click **Ok** to complete the procedure.

Figure 12-2 The Model Output Files dialog.



➤ **To remove a Section from a model**

- 1.** Make sure the **Model Components** window is active, and click on the **Model|Files...** menu item (Figure 12-2).
- 2.** Highlight the section you want to delete by clicking on it in the **Auxiliary Files (Sections)** window.
- 3.** Click on the **Remove Section** button. This will remove the section, moving all its modules into the Main Model File.

➤ **To import an existing Section from a file**

- 1.** Make sure the **Model Components** window is active, and click on the **Model|Import Section...** menu item.
- 2.** In the resulting dialog, browse for the *Auxiliary File* name containing the required Section.
- 3.** If the section being imported contains one or more names that cannot be resolved (for example, module input variables, function driving variables or function templates), these names will be replaced by the special name “**(none)**”. A message informs the user that this has occurred. Any such names will need to be resolved manually before the model can be run.

12.1 Saving models using Auxiliary Files

When Auxiliary Files are used in a model, the Save and Save As operations become somewhat more complicated. DYMEX handles these operations as follows:

Save always saves the *Main Model File*, and attempts to save any *Auxiliary Files* that have been modified (either because component modules have been changed or moved between Sections). If an *Auxiliary File* cannot be saved, an error message indicating the reason is displayed (for example, the file may be write-protected).

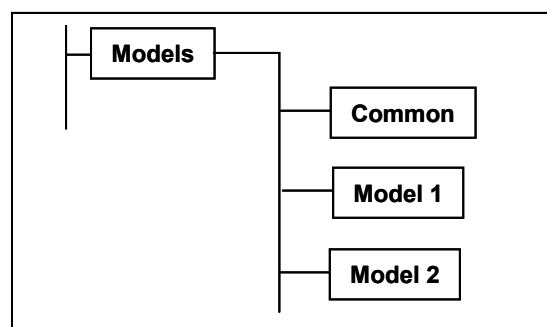
Save As can proceed in two ways, with the user able to choose which method is used:

- (a) The new *Main Model File* can contain references to the old *Auxiliary Files*, so that only the *Main Model File* is saved at the new location.
- (b) Copies of each of the *Auxiliary Files* are created in the same directory where the new *Main Model File* is placed, so that the saved copy of the model is completely independent of the old version.

12.2 Directory Layout for model files

DYMEX does not specify particular directories where models must be stored, but for the sake of consistency it is wise to follow a few rules. This becomes especially important if Auxiliary Files are used to store model sections, as the paths to the Auxiliary Files are stored in the main GMD file as relative paths. Moving model sections into different directories arbitrarily could cause a model to fail to load correctly. The scheme illustrated in Figure 12-3 is suggested for model directories. The “Models” directory would be the base directory for all DYMEX model files (not necessarily data files). Auxiliary files used in more than one model would be stored in the “Common” directory, while each model description file (and Auxiliary files used only by that model) would be stored in “Model 1”, etc.

Figure 12-3 Model directory layout.



13. **Appendix I. Transfer Functions and Transfer Rates.**

In DYMEX, a transfer process is applied at every timestep to individuals in a cohort. The result of evaluating that transfer process (the transfer rate, $r(t)$) determines the *proportion of those individuals currently in the stage* that move to the next stage during the timestep. The following procedure could be used to determine the type of transfer function to use in the simple case where the transfer is dependent on a single driving variable (commonly *Chronological Age*, *Physiological Age* or some measure of *Size*).

Let us assume we have 100 eggs of a particular species of beetle, which we monitor (at intervals of 10 degree-days) until they have all hatched. Perhaps we are keeping temperatures constant so that an interval of 10 degree-days might be a constant time interval such as a day. That is unimportant, as the same procedure could be followed even if the monitoring interval did not correspond to constant increments.

Let us say we obtain the results shown in Table A1-1 (and plotted in Figure A1-1):

Table A1-1 Accumulated degree-days of eggs and corresponding number hatching.

<u>Accumulated degree-days since laying</u>	<u>Number hatching during interval</u>	<u>Accumulated number hatched</u>
300	0	0
310	5	5
320	9	14
330	13	27
340	15	42
350	14	56
360	14	70
370	11	81
380	7	88
390	6	94
400	3	97
410	2	99
420	1	100
430	0	100

The DYMEX **Transfer** process determines what proportion of the number in the source stage transfer to the destination stage during a particular timestep. So let calculate this proportion for our data, as has been done in Table A1-2.

Figure A1-1 Number of eggs hatching plotted against accumulated degree-days.

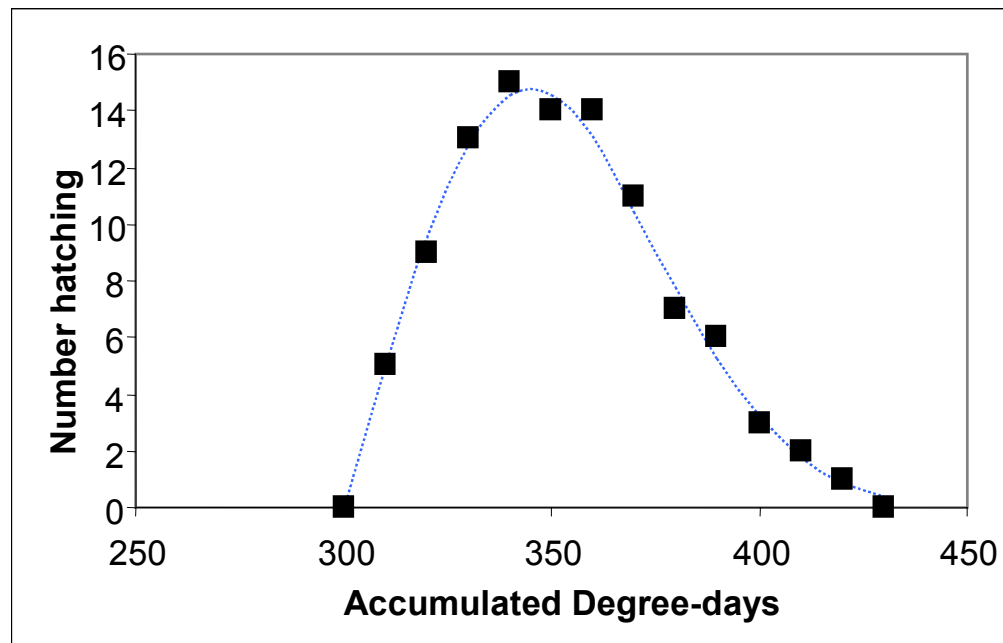


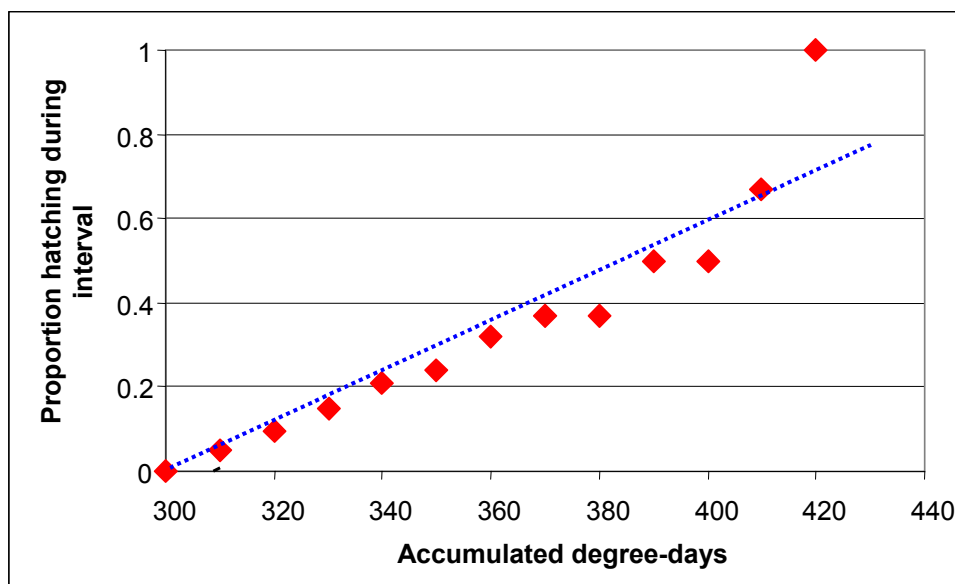
Table A1-2 Proportion hatching per time

<u>Accumulated degree-days since laying</u>	<u>Number hatching during interval</u>	<u>Proportion hatching</u>	<u>Number of eggs remaining</u>
300	0	0	100
310	5	0.05	95
320	9	0.095	86
330	13	0.15	73
340	15	0.21	58
350	14	0.24	44
360	14	0.32	30
370	11	0.37	19
380	7	0.37	12
390	6	0.5	6
400	3	0.5	3
410	2	0.67	1
420	1	1.0	0

Plotting the proportion hatching against the accumulated degree-days then gives the graph shown in Figure A1-2, indicating that a linear Transfer function would probably be a good choice for this situation. It must be noted, however, that the slope of the line is dependent on the choice of timestep for the model. This can be clearly seen if in the case where a large timestep is used, so that the eggs can accumulate 120 degree-days during the course of one

timestep. In this case, it is obvious that a very large slope would be needed (perhaps even infinity, corresponding to a **Step** function).

Figure A1-2 Proportion of eggs hatching during 10 degree-day interval plotted against accumulated degree-days.



◆ Index

- (none), 7
- , 4, 127, 129, 130, 131
- Action (function), 14
- Automatic Linking, 24
- Auxiliary File, 156–59, 159
- Auxiliary Parameter File, 28
- Basal Evaporation, 140
- Builder, 4
- C.V. Transfer, 92
- Chronological Age, 4, 33, **43**, **44**, 70, 160
- Circadian. *see* Module type, Circadian
- Cohort, 32, **39–45**, 49, 59, 65, 67, 79, 80, 82, 86, 90, 93, 94, 96, 98, 160
 - Grouping, **88**
- Cohort Duration, 6, 98
- Cohort Variable, 32, 65, 70, 79, 81, 90–95, 97
 - Output Operations, 94
 - Range, 94
 - Scope, 91
 - Update Method, 92
 - user-defined, **90–95**
- Combination Rule, 11, **57–59**, 61, 71, 79
- Competition, 64, 77, 96
- Complement-product, 58
- Component Window, 12–14, 17, 21
- Container Stage, 37
- Cost per instance, 128
- Current Average, 93
- Current Value, 93
- Cycle Shape
 - Composite (Sine+Exponential), 126, 137
 - Composite (Sine+Sine), 126, 137
 - Sine, 126, 137
- Data File Reader. *see* Module Type, DataFile
- Day of Year, 30, 100, 109, 132, 133, 142, 144
- daylength, 64, 124
- Daylength, 138
- Days since Event, 129, 132
- Days since Start, 30
- Degree-days, 60, 70, 125, 141
- Deme, 5, 7, 16, 20, 146
- Density, 96, 98
 - Average, 96
 - Cohort, 96, 98
 - Total, 97, 98
- Destination Stage, 36, 79
- Development, 31, 43, 44, 49, 50, 51, 53, 54, **59–64**, 82, 98, 123
- Development Time, 6, 35, 98
- Dewpoint Temperature, 137
- Dialog
 - , 81
 - Cohort Variable, 91
 - Combination Rule, 58
 - Environment, 47
 - Factors, 26, 116
 - Function, 53, 74
 - Function Properties, 26
 - Function Template, 148
 - Lifecycle Properties, 33
 - Lifestage Properties, 90
 - Module, 28, 102, 105, 136, 145, 146
 - Module Inputs, 23
 - Module Outputs, 25

- Parameter Properties, 26, 54, 55
- Parameter Set Properties, 28
- Process, 26
- QueryUser/Discrete, 103
- Running Mean, 118
- Timer Setup, 30
- User-defined Cohort Properties, 90
- Weather Module, 137
- Direct Update, 92
- Dispersal, 20, 39, 42, 43, 46, 49, 69, 71, 75, 79, **83–90**
- Drainage Rate, 120
- Endostage, 37
- Environment, 14, 15, 34, 35, **46–48**
 - Global, 46
- Equation Syntax, 150, **152**
- Equilibrium, 29, 109
- Equilibrium Variable, 29
- Evaporation, 138, 142, 145
- Evapotranspiration coefficient, 140
- Event Cost, 131
- Event Duration Override, 130, 132
- Event Threshold, 128, 132
- Event Variable, 131
- Exact Years, 31
- Factor, 11, 57, 141
- Fecundity, 43, 65, 66, 68, 70, 101
- Fecundity (E), 66
 - , 66
- Function, 11, 26, 52, 113, 115, 148
 - Direct, 71, 74
- Function Properties
 - Advanced, 56
 - High Limit, 57
 - Scale Factor, 56, 57
 - Y-offset, 56
- Function Template, 11, 63, 116, **148–50**, 157
- Global Scope, 81, 82, 91
- Immigration, 4, 83
- Inverted Update, 93
- Latitude, 133
- Lifecycle, 1, see Module type, Lifecycle,
 - Branch, 4, 31, 32, 35, 36, 38, 66, 80, 83
- Lifecycle Input, 100
- Lifecycle Window, 14, 18, 37
- Lifestage, 6, 15, 32, **34–39**, 50, 51, 66, 71, 73, 80, 81, 91, 96, 160
 - Name, 34
 - Output, 4, 34, 94, **97–99**
 - Reproductive, 36
 - , 34, 35
- Lifestage Output, 97
- Lifestage Window, 5, **39–43**
 - , 81, 82
- Low Limit, 57
- Meteorological Data File Reader, 107
- Mnemonic, 6, 25
- Model Description (GMD) file, 4, 156, 157, 159
- Model, Tree Diagram, 12
- Module, 2, 3, 4, 5, **7–9**, 13, 14, 17, 18, 21–27, 104, 105–7, 107, 108, 111, 113, 114, 115, 130, 142, 157
 - Create, 17
 - Description, 27
 - Input, 23–25
 - Link, 8
- Module dialog, 117, 122
- Module Library, 2
- Module Linkage, 8, 24
- Module type

-
- Accumulator, 4, *see* Running Mean
 - Adjustable Circadian, 4
 - Circadian, 61, 108, 123–27, 136, 137, 141
 - Climate Change Scenario, 4, 142, 143
 - Counter, 4, 115
 - DataFile, 25, 31, 104, 107, 108, 121, 6, 103, **133–34**, 136
 - Degree-day, 141
 - DemeSplitter, **146–47**
 - DemeStatistics, **146–47**
 - Difference, 4, 119
 - Equation, 4, 96, 113, 114, 115
 - Evaporation, 107, **134–35**, 136, 144
 - Event, 4, 71, 74, 101, 127–32
 - Expression, **111–13**
 - Function, 3, 113, **116–17**, 116
 - Lifecycle, 4, 15, **31–101**, 101
 - MetBase, 2, 107, 108
 - MetManager, 4, 109, 110, 121
 - QueryFile, 104–5, 104, 105
 - QueryUser, 25, 102, 104, 105, 116
 - QueryUser/Discrete, 4, 103
 - Running Mean, 118
 - Soil Moisture, 3, **139–41**
 - Storage, 4, **119–20**
 - Switch, 4, 121, 123
 - Timer, 17, 22, **29–31**, 142
 - Weather, 4, **136–39**
 - Module Type
 - Soil Moisture, 3
 - Mortality, 31, 44, 49, 50, 51, 58, **71–78**, 97, 98, 99
 - Continuous, 71, 73
 - Establishment, 71
 - Exit, 71
 - Naming Modules and Variables, 23
 - Number, 35, 43, 44, 49, 75, 97, 98, 99, 115, 160, 161
 - Operator Precedence, 150
 - Parameter, 4, 14, 26, 49, 52, 53, 54, 55, 115, 130
 - Limits, 55
 - Parameter Set, 27–29
 - Physiological Age, 6, 43, 44, 49, 59, 60, 61, 64, 80, 82, 83, 91, 92, 98, 160
 - Plant Growth, 63
 - Process, 11, 14, 26, 34, 35, 52, 66, 80
 - Establishment, 32
 - Exit, 32, 95
 - Process Component, 49
 - Process Components, 53
 - Progeny Production, 43, 65, 66, 67, 68, 98
 - Proportional Update, 93
 - Radiation, 64
 - Reproduction, 31, 50, 75, 101, 43, 44, 65, 66, 70
 - Resource, 14, 34, 35, 96, 97
 - Segmentation, 19, 30, 31, 118
 - Sex Ratio, 66
 - Simulation Date, 30, 100, 105, 107, 108, 128, 132
 - Simulation Id, 7
 - Soil Moisture Capacity, 140
 - Sort Order, 18, **21**, 22
 - Stage Link, 35
 - Stage Transfer, 32, 45, 49, 50, 51, 59, 70, 75, **78–83**, 98, 160
 - Sub-population, 5, 7, 8, 9, **20–21**, 24, 33, 35, 41, 46, 51, 54, 56, 68, 75, 97, 98, 99, 103, 111, 113, 115, 116, 128, 139, 146
 - Summary module, 9
-

- temperature, 64
- Time of Day, 30
- Timestep, 19, 30
- Transfer Process, 78
- Trigger, 130
- Update Method, 60
- Use Latest Inputs, 93
- Vapour Pressure, 137
- Variable, 6–7, 6, 8, 11, 14, 24, 25, 50, 53, 54, 94, 95, 96, 107, 108, 113, 114, 115, 154, 155, 157, 160
 - Cohort Property. see Cohort Variable
 - Cohort Variable. see Cohort Variable
 - Daily Cycle, 123, 137
 - Delay, 6
 - Demed, 7
 - Private, 26
 - Summary, 4, 7, 9, 154–55
- Variables Window, 15–16
- XE, 3