

Osmose ODD Description

Nicolas Barrier	Philippe Verley	Ricardo Oliveros-Ramos
Bruno Ernande	Wencheng Lau-Medrano	Criscely Luján
Alaia Morell	Morgane Travers-Trolet	Laure Velez
	Yunne-Jai Shin	

Table of contents

1	Introduction	4
2	Purpose and patterns	5
2.1	Purpose	5
2.2	Patterns	5
3	Entities, state variables, and scales	6
3.1	Entities	6
3.2	State variables	6
3.2.1	Simulation	6
3.2.2	Cell	7
3.2.3	Species	7
3.2.4	School	9
3.3	Scales	11
4	Process overview and scheduling	13
5	Design concepts	15
5.1	Basic principles	15
5.2	Emergence	15
5.3	Adaptation	15
5.4	Objectives	15
5.5	Learning	15
5.6	Prediction	16
5.7	Sensing	16
5.8	Interaction	16
5.9	Stochasticity	16
5.10	Collectives	16
5.11	Observation	16
6	Initialization	17
6.1	Seeding method	17
6.2	Using a NetCDF file	18
6.3	Using relative biomass	18

7	Input data	20
7.1	Configuration files	20
7.1.1	Key	20
7.1.2	Separators	21
7.1.3	Value	21
7.1.4	Decimal separator	22
7.2	CSV input file separator	23
7.2.1	Spatial maps	23
7.2.2	Accessibility and catchability matrix	23
7.2.3	Time series	24
7.2.4	By class time series	25
7.3	Forcing data	25
8	Submodels	26
8.1	Incoming flux	26
8.2	School initialisation	28
8.3	Resource update	28
8.4	Spatial distribution	30
8.4.1	Random distribution	30
8.4.2	Map definition	30
8.5	Mortality	34
8.5.1	Predation mortality	36
8.5.2	Starvation mortality	41
8.5.3	Migration mortality	41
8.5.4	Fishing mortality	42
8.6	Growth	59
8.6.1	Von Bertalanffy growth	60
8.6.2	Gompertz growth	61
8.7	Reproduction	63
	References	66

1 Introduction

2 Purpose and patterns

2.1 Purpose

The OSMOSE model represents the dynamics of fish communities. It is a multispecies and spatial model which lies on size-based predation, traits-based life history, and individual-based processes. The model aims to explore the functioning of marine ecosystems, the ecosystem effects of fishing and climate changes, the impacts of management measures (changes in fishing pressure and fishing strategies, implementation of marine protected areas).

The OSMOSE model represents the ecosystem dynamics of fish communities in marine ecosystems. It is an individual-based, spatially-explicit multispecies model accounting for explicit trophic interactions. The main characteristics of the model are opportunistic predation based on size and spatial co-occurrence of predators and preys

The aim of the model is to explore the functioning of marine trophic webs, notably in response to perturbations such as fishing or climate change.

2.2 Patterns

The model is evaluated by comparing the model outputs with observations of the marine ecosystem. These observations generally include catches, catches at length, biomass, abundance for different species.

3 Entities, state variables, and scales

3.1 Entities

In the current section, the different types of entities of the Osmose model are described.

- The `Configuration` entity describes the parameters and entities that will be shared among the different simulations.
- The `Simulation` entity describes a given simulation. Simulations differ from each other if stochasticity is enabled.
- The `Cell` entity describes the individual cells of the Osmose computation 2D grid (longitude, latitude, cell index, etc.)
- The `Species` entity represents the species whose life-cycle is fully represented. This entity contain **fixed** parameters, which are not modified during the simulation.
- The `BackgroundSpecies` entity represents the species whose life-cycle is not represented. These species can feed on and be eaten by `Species`, and can be fished. This entity contain **fixed** parameters, which are not modified during the simulation.
- The `ResourceSpecies` entity represents the low-trophic level species. These species cannot predate `Species` or `BackgroundSpecies` but can be eaten by them. This entity contain **fixed** parameters, which are not modified during the simulation.
- The `School` entity describes a group of `Species` individuals, whose life cycle is fully considered and which all share the same characteristics (age, weight, species, location, etc, trophic level).
- The `BackgroundSchool` entity describes a group of `BackgroundSpecies` individuals, whose biomass is provided as input and whose life-cycle is not simulated and which all share the same characteristics (age, weight, species, location, etc, trophic level).
- The `Resource` entity describes a swarm of `ResourceSpecies`, whose biomass is provided as input.

3.2 State variables

3.2.1 Simulation

The `Simulation` entity contains the following state variables:

Table 3.1: Simulation state variables

Variable	Description	Type	Units
i_step_simu	Times step of the simulation	int, $\in [0, N_{year} \times N_{step_year}]$	
rank	Index of the simulation	int, $\in [0, N_{simulation}]$	
schoolSet	List of the School inside the simulation	SchoolSet	
backSchoolSet	List of the BackgroundSchool inside the simulation	BackgroundSchoolSet	

3.2.2 Cell

The Cell entity contains the following state variables:

3.2.3 Species

The Species entities contain the following state variables:

Table 3.2: Species state variables

Variable	Description	Type	Units
name	name of the species	string	
index	index of the species	int, $\in [0, N_{species} - 1]$	
fileIndex	index of the species in the configuration file	int, > 0	
lifespan	species life span	int, > 0	time-steps
c	allometric factor	float	
bPower	allometric power	float	
sizeMaturity	Size at maturity	float	cm
ageMaturity	Age at maturity	float	year
eggSize	Size of eggs	float	cm
eggWeight	Weight of eggs	float	g
firstFeedingAgeDt	First feeding age	int	time-step

Variable	Description	Type	Units
larvaeToAdultsAgeDt	Age at which the school is considered as adult	int	time-step
TL_EGG	egg trophic level	float, = 3.0	

The BackgroundSpecies entities contain the following state variables:

Table 3.3: Background species state variables

Variable	Description	Type	Units
name	Species name	string	
index	Species index.	int, $\in [N_{species}, N_{species} + N_{bkg species} - 1]$	
fileindex	index of the species in the	int, > 0	
age	Age array	[float]	years
ageDt	Age array	[int]	time-step
c	allometric factor	float	
bPower	allometric power	float	
classProportion	Proportion of biomass attributed to each size-class	float[]	%
length	Length array	float[]	cm
nClass	Number of size-classes	int, > 1	
trophicLevel	Array of trophic level	float[]	
timeSeries	Time series of proportion of biomass attributed to each size-class		

Variable	Description	Units
accessMax	Maximum value for resource accessibility	0.99
accessibilityCoeff	Fraction of the resource biomass available to the fish	[0, 1]
fileindex	Species index in the configuration file	
index	Resource species index	
name	Resource species name	
sizeMax	Maximum size of the plankton size	Cm
sizeMin	Maximum size of the plankton size	Cm

Variable	Description	Units
trophicLevel	Trophic level	

3.2.4 School

The School entity contains the following state variables:

Table 3.5: School state variables

Variable	Description	Type
abundance	Number of fish in the school at the beginning of the time-step	float
biomass	Biomass of the school at the beginning of the school	float
instantaneousAbundance	Instantaneous number of fish in the school	float
instantaneousBiomass	Instantaneous biomass of the school at the beginning of the school	float
discardedBiomass	Biomass of the school that has been discarded by each fishing gear	float[]
fishedBiomass	Biomass of the school that has been landed by each fishing gear	float[]
nDead	Number of dead fish in the current time step for each mortality cause (predation, fishing, natural mortality, starvation)	float
predSuccessRate	Predation success rate	float
preyedBiomass	Biomass of prey ingested by the school at current time step	float
preys	List of preys eaten by the school	HashMap<Integer
uuid	Unique identifier of the individual	UUID
lat	Latitude of the individual	float
lon	Longitude of the individual	float
x	i-index of the individual in the 2D grid	int
y	j-index of the individual in the 2D grid	int
ageDt	Age of the school	int
eggRetained	Buffer variable that will temporarily retain some eggs inside a time step	float

Variable	Description	Type
length	Length of the individuals in the school	float
lengthi	Length of the individuals in the school at the beginning of the time-step	float
out	True if the school is out of the computation domain	boolean
starvationRate	Starvation rate of the school	float
trophicLevel	Trophic level of the school	float
weight	Weight of the school	float
species	School species	Species
abundanceHasChanged	True if abundance has changed in the current time-step	boolean
accessibility	Array of accessibility of the school to its preys	float[]
ageDeath	Age at which a school individuals die times the abundance	float[]

Table 3.6: Background school state variables

Variable	Description	Units
abundance	Number of fish in the school at the beginning of the time-step	
biomass	Biomass of the school at the beginning of the school	Tons
discardedBiomass	Biomass of the school that has been discarded by each fishing gear	Tons
fishedBiomass	Biomass of the school that has been landed by each fishing gear	Tons
nDead	Number of dead fish in the current time step for each mortality cause (predation, fishing, natural mortality, starvation)	
predSuccessRate	Predation success rate	%
preyedBiomass	Biomass of prey ingested by the school at current time step	Tons
preys	List of preys eaten by the school	
uuid	Unique identifier of the individual	
lat	Latitude of the individual	
lon	Longitude of the individual	

Variable	Description	Units
x	i-index of the individual in the 2D grid	
y	j-index of the individual in the 2D grid	
classIndex	Class index to which belongs the school	
bkgSpecies	School background species	

3.3 Scales

The basic units of OSMOSE are fish schools, which are composed of individuals that belong to the same species, and that have the same age, size (length, weight), food requirements and, at a given time step, the same geographical coordinates. From the school states (hereafter called individual states), biomass and abundance can be tracked at the population or community levels along with the size, age, and spatial dimensions.

Other variables can be reported such as the trophic level, the diets, the different sources of mortality, the catches from fishing operations. Because each school simulated in OSMOSE is represented from the egg stage to the terminal age, which necessitates high calculation and memory capacities, and because comprehensive information on entire life cycles needs to be parameterized, the selection of focus species is made parsimoniously, and usually between 10 and 20 high-trophic level species or functional groups are explicitly considered in OSMOSE applications.

The model operates on a weekly to monthly time step, and runs up to 100 years or more depending on applications and simulations.

For eggs (age 0), weight and sizes are provided as parameters. For the others, conversion from size to weight (and conversely) is obtained by using allometric relationships:

$$W = c \times L^b$$

$$L = \left(\frac{W}{c}\right)^{\frac{1}{b}}$$

where the c parameter is a condition factor, and b the allometric power.

Biomass to abundance conversion for a school is made by using the mean weight of the school:

$$B = N \times W$$

$$N = \frac{B}{W}$$

Variable	Description
species.length2weight.condition.factor.sp#	Allometric factor (<i>c</i>)
species.length2weight.allometric.power.sp#	Allometric power (<i>b</i>)
species.egg.size.sp#	Egg size (cm)
species.egg.weight.sp#	Egg weight (g)

4 Process overview and scheduling

The life cycle of each focus species included in the OSMOSE model is modelled, starting with the egg stage. At the first time step, eggs are produced and split into a number of super-individuals called schools. At each time step, OSMOSE simulates the main life history processes for these schools, starting with the release of fish schools within their distribution area which is specified in input for each species and by age when presence/absence data are available. Then different sources of mortality are applied including predation, fishing, starvation and other natural mortality. In OSMOSE, predation is assumed to be opportunistic and based on predator and prey size adequation and spatio-temporal co-occurrence. Depending on the predation success, somatic growth is then implemented and mature individuals spawn at the end of the time step and produce new eggs for the next step.

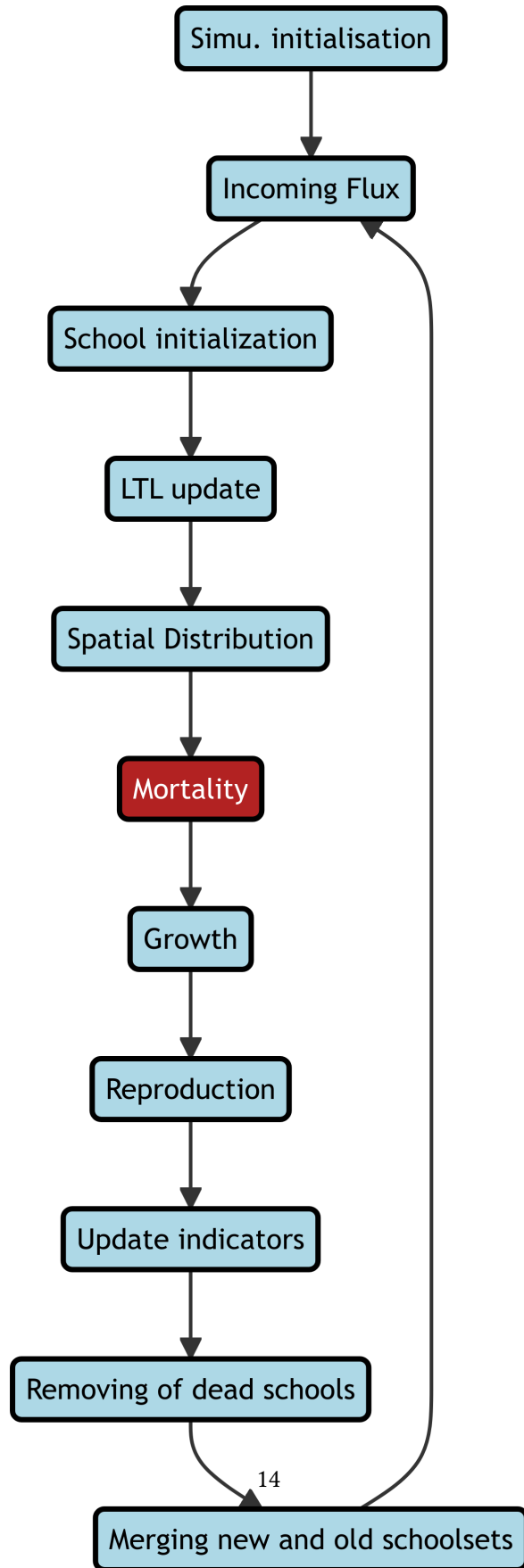


Figure 4.1: Scheduling

5 Design concepts

Following the ODD framework and terminology, we briefly present here some design concepts characterizing OSMOSE.

5.1 Basic principles

Dieta data, such as those of Shannon et al. (2003), suggest that patterns in fish diets include variability in time and space and prey switching, cannibalism and omnivory. Therefore, two species can be a predator or prey of one another depending on the life stages considered. Therefore, predation in Osmose is size-based, taking into account thresholds for predator/prey size ratio, and relies on the spatio-temporal co-occurrence of individuals.

5.2 Emergence

From the local interactions, population and community dynamics emerge. In particular, the whole food web structure emerges from size-based local predation interactions.

5.3 Adaptation

Adaptation is not implemented in Osmose.

5.4 Objectives

Objectives are not implemented in Osmose.

5.5 Learning

Learning is not implemented in Osmose.

5.6 Prediction

5.7 Sensing

Schools are assumed to know perfectly all the potential prey which are located in its vicinity, i.e. in the same cell of the grid. They also know the limits of their habitat.

5.8 Interaction

Super-individuals/schools interact locally through predation events.

5.9 Stochasticity

There are different sources of stochasticity in the model. First, the order at which schools act and interact. There is a randomization of the precedence of schools crossed with a randomization of different sources of mortality (predation, fishing, other natural mortality). In addition, fish movements are also randomized within their habitats.

5.10 Collectives

As the total number of fish (from eggs to adult fish) to be taken into account in the simulated system can reach a value of the order of 10^{12} , the model was not brought down to the fish level but to an aggregated level consisting of a group of fish having similar ecological attributes. The unit of action and interaction, i.e., the “super-individual” as defined by Scheffer et al. (1995), is a group of fish having the same size, the same spatial coordinates, requiring similar food, and belonging to the same species (therefore having similar physiological and morphological characteristics). For convenience, this super-individual is also called a “fish school” in the following sections.

5.11 Observation

For model testing and fitting, a variety of auxiliary state variables can be used in output of the model and compared with observations, data time series, and maps. Typically, species biomass and commercial catches, age or size distribution of abundance/biomass, diets can be confronted to observations.

6 Initialization

In Osmose, there are several initialisation methods.

6.1 Seeding method

In the seeding method, the system starts from a pristine state, with no schools in the domain. For a few years (user-defined), Osmose will release some eggs for every species. The eggs enter the different steps of the life cycle, and once the fish reach sexual maturity, the reproduction process takes over and Osmose stops the seeding, unless the spawning stock biomass gets depleted. In that case Osmose resumes the seeding by releasing some eggs until there are again mature individuals in the system for carrying on the reproduction process. Osmose completely ceases the seeding when the simulation reaches the maximal number of years for seeding (user-defined).

By following this approach:

- No assumption is made about the structure of the populations but it emerges from individual interactions
- it reduces computing requirements for the spin-up as the first years are the fastest to run
- it reduces the number of time steps for the spin-up
- it minimizes the amplitude of population oscillations

The initialisation process is controlled by two parameters:

- `population.seeding.year.max` defined the number of years for running the seeding process (from year 0 to `population.seeding.year.max`), and during which Osmose ensures that some eggs will be released even though there are no mature individuals in the system. Then, the seeding ceases completely until the end of the simulation. If the parameter is not specified, Osmose will set it by default to the lifespan of the longest lived species of the system.
- `population.seeding.biomass.sp#` defines the spawning stock biomass (SSB) that Osmose considers during the seeding period when there are no mature adults to ensure the reproduction process. SSB is then used to compute the number of eggs to be released in the system (cf. `{numref}reprod`).

6.2 Using a NetCDF file

Another possibility to initialize the Osmose model is by using a NetCDF file. This method is used if the `simulation.restart.file` or the `population.initialization.file` are set.

These parameters point to the NetCDF file that will be used to restart the simulation. If `simulation.restart.file` is used, then one file per replicate is expected (suffix ends with `.nc.#`, with # the index of the simulation).

6.3 Using relative biomass

Another method to initialize Osmose is to use relative biomass. This method is called if the `population.initialization.relativebiomass.enabled` is enabled. With this method, the user specifies, for each species:

- `population.initialization.biomass.sp#`: the initialization biomass for each species (B_s , float)
- `population.initialization.size.sp#`: the size-boundaries (float[]). Size is $N_{class} + 1$
- `population.initialization.tl.sp#`: the trophic levels for each size-class (float[], size N_{class})
- `population.initialization.relativebiomass.sp#`: the proportion of total biomass to attribute to each size-class ($P_{s,k}$, float[], size N_{class} , $\in [0, 1]$)
- `population.initialization.age.sp#`: the age to attribute to each size-class (float[], size N_{class})
- `population.initialization.nschool.sp#`: the number of schools to create for each size-class ($N_{s,k}^{int}[]$)

Using all these parameters, schools for a given species are initialized as follows.

- For each size-class, computes the biomass to release by multiplying input biomass and relative proportion: $B_{s,k} = B_s \times P_{s,k}$.
- Loop over the number of schools to create
- Randomly select a length between the lower and upper size-bonds of the given class. If age is 0, force length to be equal to species `eggSize`.
- Using length, computes the weight using allometric relationship. If age is 0, force weight to be equal to species `eggWeight`.
- Compute the number of individuals to put in school i : $A_{s,k,i} = \frac{B_{s,k} \times 10^6}{N_{s,k} \times W_{s,k,i}}$

The steps are summarized below:

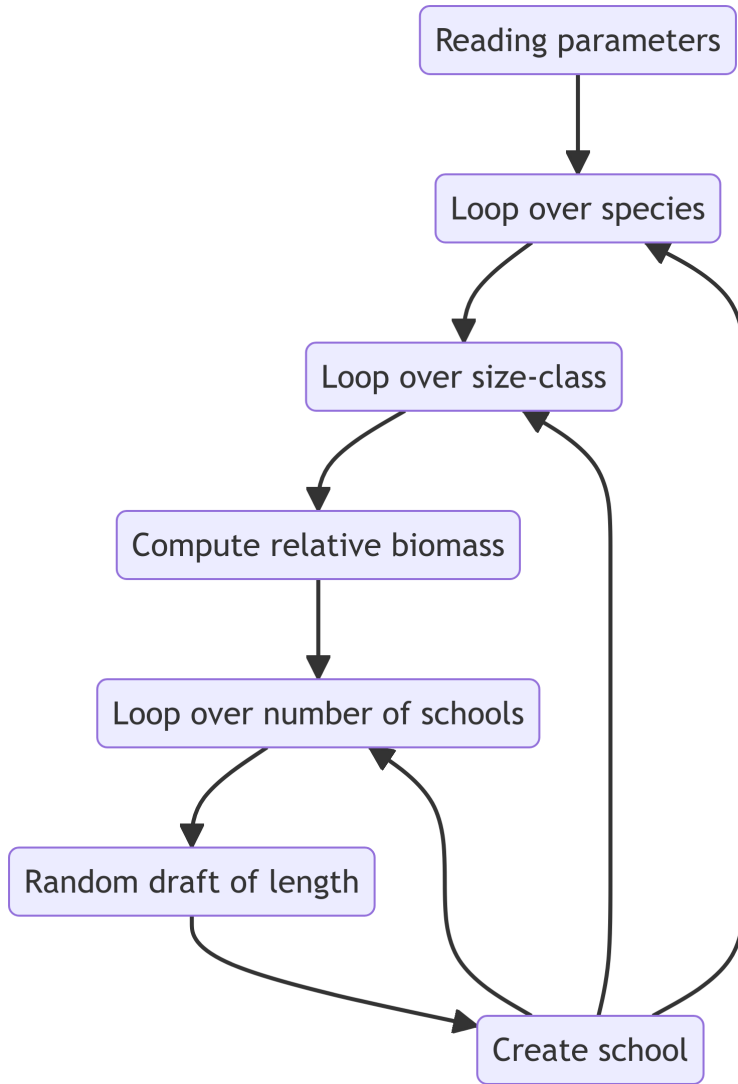


Figure 6.1: Initialization using relative biomass

7 Input data

This section describes the input files that are required to run the Osmose model.

7.1 Configuration files

Running Osmose requires specifying the path to an Osmose configuration file.

An Osmose configuration file is a text based file. The name of the file doesn't matter, but we recommend that you avoid any special characters and spaces in the name of the file as this may cause IO errors when you'll be running Osmose from the command line or calibrating the model in a UNIX environment. The extension of the file does not matter either.

We call the **main** configuration file the file that is passed as an argument. The main configuration file can contain comments, empty lines and parameters. Osmose scans each line of the main configuration file looking for parameters and automatically discards:

- empty lines (regardless of blank or tab characters).
- lines that start with a punctuation character, one of `!"#$%&'()*+,-./:;<=>?@[\\]^_`{|}~`

Tip

For comments, it is recommended to start the line with `#` or `//`

A parameter is formed by the juxtaposition of three elements: **key**, **separator** and **value**.

7.1.1 Key

The **key** can be any sequence of characters, without blank or any special characters (dot, hyphen and underscore are accepted). Example of keys:

```
simulation.ncpu  
predation.ingestion.rate.max.sp6
```

Osmose makes no difference between upper and lower case: `simulation.ncpu`, `simulation.Ncpu`, `Simulation.NCPU`, `SIMULATION.NCPU` designate the same key.

Keys starting with `osmose.configuration` have a special meaning for the configuration manager. It means that the value of this parameter is the path to another Osmose configuration file and the parameters in that file are to be loaded into the current configuration. This way, instead of having one big configuration file with all the parameters, it is possible to split the parameters into as many files as the user wants.

This process works recursively: a file can contain one or several `osmose.configuration` parameters that point to embedded configuration files, which may also contain one or several `osmose.configurationparameters`, and so on. Osmose handles the sub-configuration file exactly the same way as it handles the main configuration file (same convention for comments, special characters and naming of).

Relative path

Paths parameters are always defined relative to the configuration file in which they are defined

7.1.2 Separators

The separator can be any of the following characters:

- equals =
- semicolon ;
- comma ,
- colon :
- tab `\\t`

Parameters in the same configuration file can have different separators (although it is advisable to be consistent and use the same one). The configuration manager will work out what the delimiter for each parameter.

7.1.3 Value

The value can be any sequence of characters (even empty). The configuration manager does not attempt to interpret the value when loading the configuration files. It simply stores it as a string object. A value can be passed to the configuration manager as:

- a string
- an integer
- a float

- a double
- a boolean
- an array of strings
- an array of integers
- an array of floats
- an array of doubles
- a resolved path

An array of values is a sequence of values with a separator in between. Accepted separators for an array of values are the same characters listed above. The separator for an array of values can either be the same or distinct from the separator between the key and the value. The following examples are valid entries:

```
movement.map0.season;0;1;2;3;4;5
movement.map0.season=0;1;2;3;4;5
movement.map0.season = 0, 1, 2, 3, 4, 5
movement.map0.season : 0 ; 1 ; 2;3;4;5
```

and are equivalent for the configuration manager. It can be summarize as:

```
key separator1 value1 separator2 value2 separator2 value3 separator2 value4
```

with separator1 either equal or different from separator2.

7.1.4 Decimal separator

Osмосе is quite flexible in terms of separators for the configuration files (automatically detected), the CSV output files (user-defined by parameter `output.csv.separator`) and the CSV input files (automatically detected). On the contrary it restricts the decimal separator to dot, and only dot.

```
Example given: 3.14159265 or 1.618
```

Any other decimal separator (COMMA for instance as in French locale) will be misunderstood and will unmistakably lead to errors. One must be careful when editing CSV input files (either parameters or time series) with tools such as spreadsheets that may automatically replace decimal separator depending on the locale settings. Future Osмосе release might allow the use of specific locale but for now remember that DOT is the only accepted decimal separator.

7.2 CSV input file separator

Many Osmose parameters are paths to CSV file, for instance:

```
movement.map0.file
mortality.fishing.rate.byDt.byAge.file.sp#
reproduction.season.file.sp#
```

CSV input file separators can be any of the following characters:

- equals =
- semicolon ;
- comma ,
- colon :
- tab \\t

Osmose will detect the separator automatically and independently for every CSV file. It means that one CSV input file may be comma separated and another one may be tab-separated.

7.2.1 Spatial maps

The spatial dynamics of Osmose (species distribution, fishing dynamics, etc.) can be configured using CSV files. Each line of the CSV file represents a given latitude, and each column represents a given longitude.

In the CSV file, the land cell must either have negative or NA values.

! Warning

The land cells (NA values), the number of lines (longitudes) and columns (latitudes) must be consistent with the lower trophic level (LTL) forcing file!

7.2.2 Accessibility and catchability matrix

Accessibility and catchability matrix are provided as a CSV file. The first line specifies the name of the predator (either fish species or fishing gear), while the first column specifies the name of the prey (either fish species or biogeochemical forcing).

In accessibility matrix, the size information is provided by appending the < character before the upper bound of the class value.

For example, if the predator column is cod , the values will be used for all cod schools. On the other hand, if the CSV file contains three columns:

cod < 0.25	cod < 2	cod
0.25	0.5	0.7

The 0.25 value will be used for cod schools smaller than 0.25 cm, the 0.5 value will be used for cod schools between 0.25 and 2cm, and the 0.7 value will be used for all the other school size.

In catchability matrix (used for fishing mortality and discards when fisheries are enabled), the predator names are the fishing gears.

7.2.3 Time series

Time series are provided in CSV files that contain two columns and a header. The first column contains the time value, which are not used by Osmose. The second column contains the values that will be used. Below is an example of CSV time series:

Table 7.2: Example of time series CSV file

Time	Value
0	0.05
0.083333336	0.15
0.16666667	0.15
0.25	0.15
0.33333334	0.05
0.41666666	0
0.5	0
0.58333333	0
0.66666667	0
0.75	0
0.83333333	0
0.91666667	0

If the number of values is less than the total number of simulation time steps, the same values will be repeated over and over. For example, in a 100 year simulation with a bi-monthly time-step (24 time steps per year), the user can either provide 2400 values (one value per time step) or 24 values (the same values will be used 100 times).

7.2.4 By class time series

7.3 Forcing data

Secondly, the modelled system is driven by prey fields which are not explicitly represented as focus species in OSMOSE. For example, biomass fields of phytoplankton and zooplankton varying in space and time are typical input to OSMOSE. In the different OSMOSE applications, these prey fields were usually produced from coupled hydrodynamic and biogeochemical (BGC) models such as ROMS-NPZD (Travers-Trolet, Shin, and Field 2014), ROMS-PISCES (Oliveros Ramos 2014), NEMOMed-ECO3M (Halouani et al. 2016) or are derived from observational data (Grüss et al. 2015; Fu et al. 2013). Benthic resources can also drive the dynamics of the system as in Halouani et al. (2016).

8 Submodels

8.1 Incoming flux

Some species might not do the full life cycle within the simulated domain (reproduce outside the domain for example). For such species, one way to take them into account is to include a flux of schools with user-defined age or length at specific time of the year. This is done by setting either the `flux.incoming.byDt.byAge.file.sp#` or `flux.incoming.byDt.bySize.file.sp#` parameters, which are the paths of the CSV files containing the input flux.

It provides the biomass (in tons) for the given size or age classes and must be as follows:

Time step / Age	0	2	3	4
0	0	500	800	0
1	0	500	800	0
2	0	400	700	0
3	0	400	700	0

The values of the class intervals (first row) are automatically scanned by Osmose. If `flux.incoming.byDt.byAge.file.sp#` is provided, the corresponding length will be computed from age classes using the `growth ageToLength` method. Conversely, if `flux.incoming.byDt.bySize.file.sp#` is provided, the corresponding age will be computed using the `lengthToAge` method (cf. Figure 8.1).

In the above example, there are 4 age classes: `[0 2[`, `[2 3[`, `[3 4[` and `[4 lifespan[`. Osmose will compute the incoming age at the middle of the interval (i.e. 1 year, 2.5 year, 3.5 year, etc). For the first time step, Osmose will therefore input 500 tons of 2.5 year-old school and 800 tons of 3.5 year school.

The values of the time step (leftmost column) does not matter. Osmose assumes there is one line per time step. The number of time steps in the CSV file must be a multiple of the number of time steps per year. If the time series is shorter than the duration of the simulation, Osmose will loop over it. If the time series is longer than the duration of the simulation, Osmose will ignore the exceeding steps.

The state variable that is updated is the `Simulation.schoolSet` variable, to which is added these newly created schools.

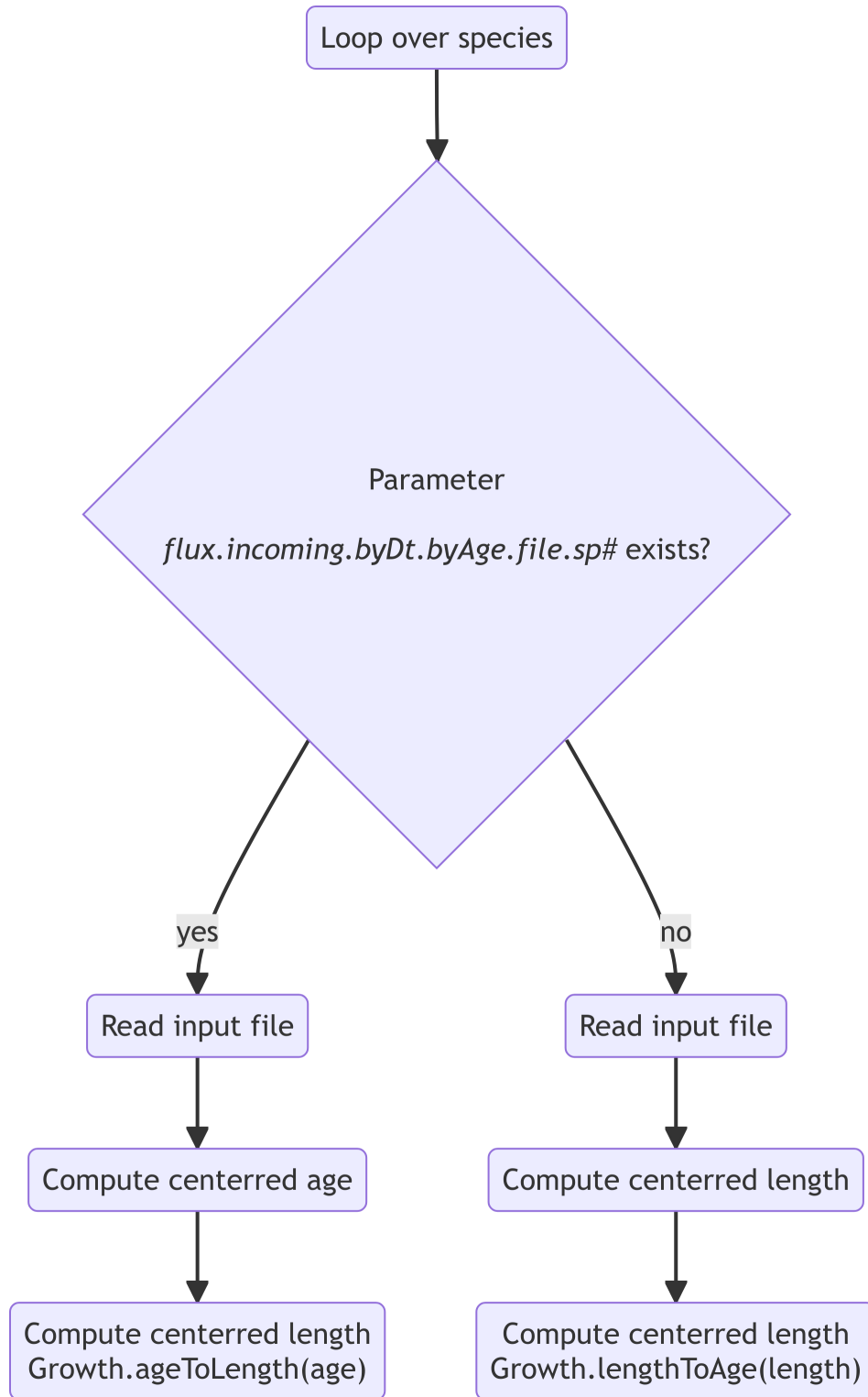



Figure 8.1: Input flux initialisation

The number of schools created for each species and each size-class is controlled by the `simulation.school.sp#` (also used in the reproduction processes). If the abundance is less than the number of schools, one school of abundance A is created. Else, N_{school} of abundance $\frac{A}{N_{school}}$ are created (cf. Figure 8.2).

Size-classes are handled the same way than age classes, except that the last size-class covers the $[4, \text{Linf}[$ interval.

 Warning

The incoming biomass should be calibrated.

8.2 School initialisation

The school initialisation process allows to reset some variables at the beginning of the time steps.

These variables and their value after the initialization are listed below:

Variable	Value
<code>out</code>	false
<code>abundance</code>	instantaneous abundance at previous time-step
<code>biomass</code>	abundance x weight
<code>lengthi</code>	length at previous time step
<code>preys</code>	empty listed
<code>preyedBiomass</code>	0
<code>predSuccessRate</code>	0
<code>nEggs</code>	0
<code>ingestion</code>	0
<code>nDead</code>	array of 0
<code>ageDeath</code>	array of 0
<code>fishedBiomass</code>	array of 0
<code>this.discardedBiomass</code>	array of 0

8.3 Resource update

The resource update consists in updating the resource forcings by reading the proper time-steps in the input NetCDF files. This update is done for both resource and background species.

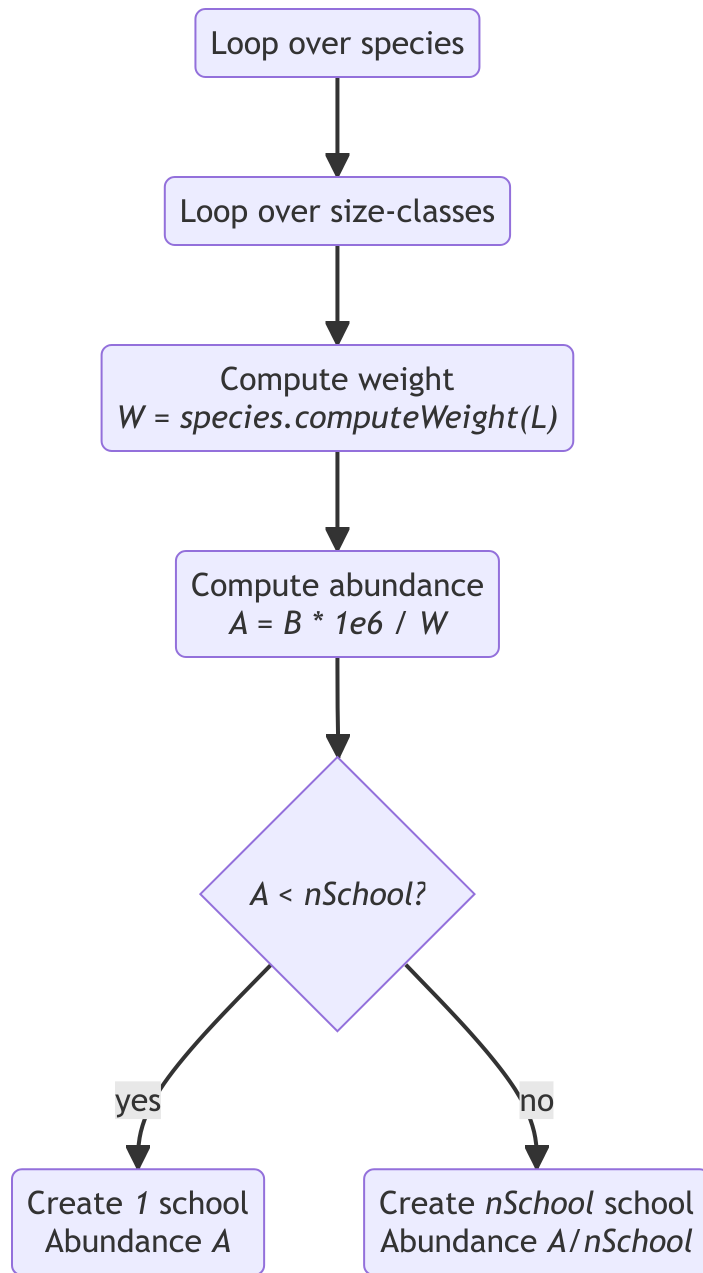


Figure 8.2: Influx process

8.4 Spatial distribution

At each time-step, the spatial distribution of fished is randomly changed based either based on distribution maps or based on random walk processes.

The displacement mode is defined through the `movement.distribution.method.sp#`, whose values are either `random` or `maps`.

8.4.1 Random distribution

For random distribution, two parameters need to be defined:

Table 8.3: Random distribution parameter

Parameter	Description
<code>movement.distribution.ncell.sp#</code>	Number of cells in which the species is allowed to move. If undefined, the whole domain is used.
<code>movement.randomwalk.range.sp#</code>	Number of adjacent cells a species can use during random walk (foraging)

At the beginning of the simulation, during the initialization process, the areas where the schools can live is initialized:

- If `ncell` is not set or is equal to the total number of ocean cells, this areas is set equal to all the ocean cells of the domain.
- Else, one cell is first randomly picked up in the whole domain, and the domain is extended from neighbours to neighbours until the number of cells reaches `ncell` (cf. Figure 8.3)

Then, at each time-step, schools are moved following a random walk method within this domain, with the range defined in the parameters. Unlocated schools are randomly put in one of the cell of the defined domain.

8.4.2 Map definition

Another way to define species distribution is to use distribution maps, which can generally be obtained from niche modelling.

A single map distribution can be defined either using CSV (Table 8.4) or NetCDF (Table 8.5) files

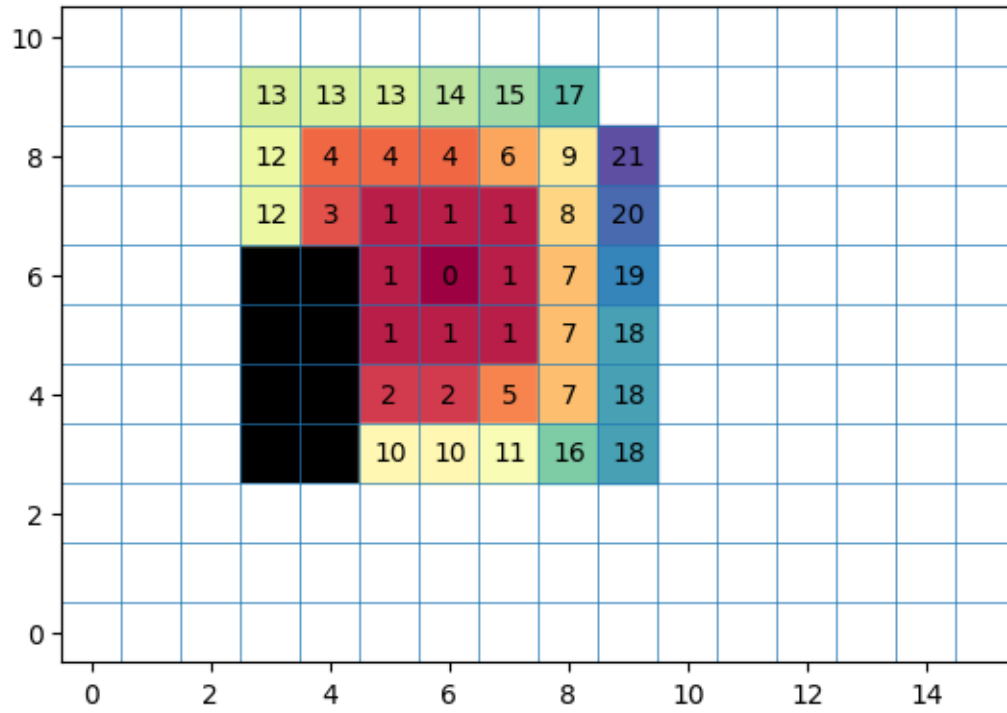


Figure 8.3: Initialization process of species domain when random distribution is used with a limited number of cells. Black squares indicate land cells, colors and numbers indicate the iteration step when the cell is included in the domain.

Table 8.4: CSV map distribution parameter

Parameter	Description
movement.randomwalk.range.sp#	Number of adjacent cells a species can use during random walk (foraging)
movement.file.map#	Name of the CSV file that contains the distribution map
movement.species.map#	Name of the species associated with the map.
movement.initialAge.map#	Minimum age (in years) when to use the map.
movement.lastAge.map#	Maximum age (in years) when to use the map.
movement.initialYear.map#	Minimum simulation time (in years) when to use the map.
movement.lastYear.map#	Maximum simulation time (in years) when to use the map.
movement.years.map#	Array of years during when to use the map. (instead of setting initial and final year)
movement.steps.map#	Array of time-steps during when to use the map

Table 8.5: NetCDF map distribution parameter

Parameter	Description
movement.netcdf.enabled	True if NetCDF maps are provided
movement.species.map#	Species to which the movement map is associated
movement.file.map#	NetCDF file containing the spatial distribution maps
movement.nsteps.year.map#	Number of time steps per year associated with the file.
movement.initialAge.map#	Minimum age (in years) when to use the map.
movement.lastAge.map#	Maximum age (in years) when to use the map.
movement.variable.map#	Name of the NetCDF variable containing the spatial distribution file.

One map is associated to a unique species for a given age span, year span and season. The full spatial

distribution of a species can be represented using as many maps as necessary to cover different age spans and/or year spans and/or seasons. Let's now have a look at each parameter in detail.

Note that the CSV file has the same number of lines and columns as the OSMOSE grid. The distribution area can be defined using either a presence/absence map (1 for presence, 0 for absence) or a map of probability of presence (containing values ranging from 0 to values <1).

The same CSV file can be used to define different maps.

If the file path is set to null it means that the schools concerned by this map are out of the simulated domain (e.g., migrating species).

% See the parameter mortality.out.rate.sp for mortality rate of species momentarily out of the simulated area.

When a school comes back to the simulated area, it will be randomly located on a new map (the one corresponding to the species and age class of the school at the current time step of the simulation).

Several maps can be defined for representing the spatial distribution of a single species. For example:

```
#Map 0
movement.map0.species = euphausiids
movement.map0.file = maps/mymap_euphau1.csv
movement.map0.age.min = 0
movement.map0.age.max = 0.2
movement.map0.year.min = 0
movement.map0.year.max = 40
movement.map0.season = 0;1;2;3;4;5;6;7;8;9;10;11;12;13;14;15;16;17;18;19;20;21;22;23

#Map 1
movement.map1.species = euphausiids
movement.map1.file = maps/mymap_euphau2.csv
movement.map1.age.min = 0.2
movement.map1.age.max = 1
movement.map1.year.min = 0
movement.map1.year.max = 40
movement.map1.season = 0;1;2;3;4;5;6;7;8;9

#Map 2
movement.map2.species = euphausiids
movement.map2.file = maps/mymap_euphau3.csv
movement.map2.age.min = 0.2
movement.map2.age.max = 1
movement.map2.year.min = 0
```

```

movement.map2.year.max = 40
movement.map2.season = 10;11;12;13;14;15;16;17;18;19;20;21;22;23

```

By increasing the number of maps, the description of the spatial distribution can be as detailed and refined as you want, as long as you have such information. It will allow for instance to create some maps for eggs (an egg in Osmose is a new school of age zero that is created during the reproduction process), some maps for the juveniles and some maps for the adults, as many as necessary to describe ontogenetic migrations.

From one time step to an other, the movement manager checks whether a given school remains in the same map or should “jump” to an other map (e.g. eggs map to juvenile map or adults in summer to adults in winter).

In the latter case (change of map), the schools are relocated randomly in the new map. The cell selection algorithm is as follows:

- A random cell ocean is selected, whose probability value is $P(i, j)$
- A random number is drafted, which is called R .
- If $P(i, j) < R \times P_{max}$, the operation is repeated. Else, the cell is selected.

In the former case (same map), the movement manager mimics foraging movement with a random-walk that moves schools to immediately adjacent cells within their distribution area.

8.5 Mortality

Within each time step, the total mortality of a given school is comprised of predation mortality caused by other schools, starvation mortality, fishing mortality, and diverse other natural mortality rate. The four different mortalities are computed so as to represent quasi simultaneous processes, and we consider that there is competition and stochasticity in the predation process.

Within each time step, OSMOSE considers each pair of school/source of mortality in turn in a random order. To ensure that the random order of the mortality sources and of the schools does not bias the resulting instantaneous mortality rates applied and effectively correspond to the mortality rates specified in input (for fishing and diverse natural mortality), all the mortality events are iterated within a time step over a fixed number of sub-time step.

Table 8.6: Stochastic mortality parameters

Parameter	Description
stochastic.mortality.seed	Integer to fix the random number generator
mortality.subdt	Number of mortality sub time-steps

Mortality processes are detailed below.

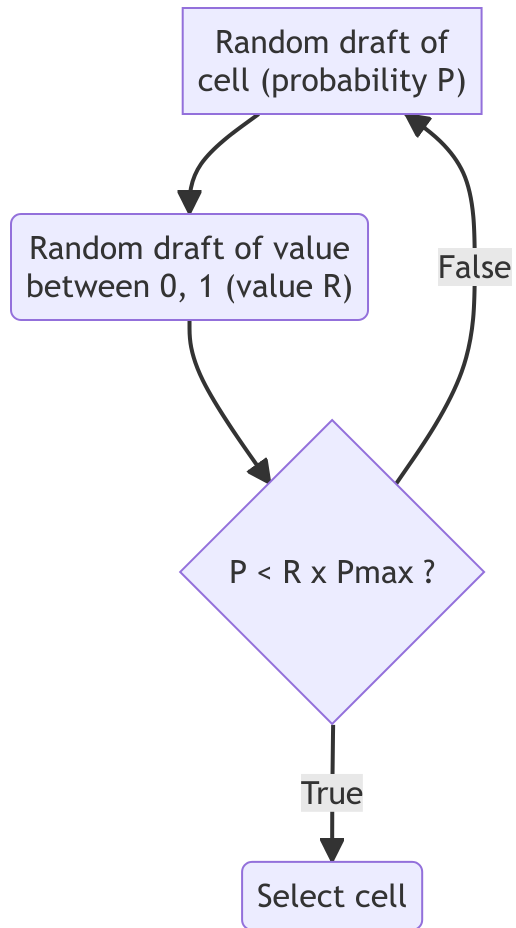


Figure 8.4: Movement process

8.5.1 Predation mortality

The central assumption in OSMOSE is that predation is an opportunistic process, which depends on:

- the overlap between predators and potential prey items in the horizontal dimension
- size adequacy between the predators and the potential prey (determined by **predator/prey size ratios**); and when the information is available
- the accessibility of prey items to predators, which depends on their vertical distribution (this being determined by means of **accessibility coefficients**). Thus, in OSMOSE, the food web structure emerges from local predation and competition interactions.

During the predation mortality process, the predation success rate (predSuccessRate attribute) is updated.

8.5.1.1 Size predation

Size-predation matrix is controlled by two parameters. The predator school S_{pred} can only feed on prey schools whose length belongs to a given interval:

$$R_{max} < \frac{L_{pred}}{L_{prey}} \leq R_{min}$$

with R_{min} and R_{max} the maximum and minimum predator/prey size ratios. Reorganizing this inequality, we obtain:

$$\frac{L_{pred}}{R_{min}} \leq L_{prey} < \frac{L_{pred}}{R_{max}}$$

Therefore, the minimum and maximum sizes of a prey that a predator can eat is given by:

$$L_{max} = \frac{L_{pred}}{R_{max}}$$

$$L_{min} = \frac{L_{pred}}{R_{min}}$$

Parameter	Description
predation.predPrey.stage.structure	Structure to determine thresholds for predator/prey size ratios (age or size)

Parameter	Description
<code>predation.predPrey.stage.threshold.sp#</code>	Array of age or size thresholds
<code>predation.predPrey.sizeRatio.max.sp#</code>	Array of R_{\max} values
<code>predation.predPrey.sizeRatio.min.sp#</code>	Array of R_{\min} values

To make sure that $L_{max} < L_{min}$, the `predation.predPrey.sizeRatio.max.sp#` and `predation.predPrey.sizeRatio.min.sp#` must verify $R_{min} > R_{max}$

Since resource groups are defined by a range of sizes, and not by a single sizes, the predator will feed on a given percentage of the resource:

$$R_{rsc} = \frac{\min(L_{max_{rsc}}, L_{max}) - \max(L_{min_{rsc}}, L_{min})}{L_{max_{rsc}} - L_{min_{rsc}}}$$

which is the overlapping range of the predator accessible range and of the resource size range.

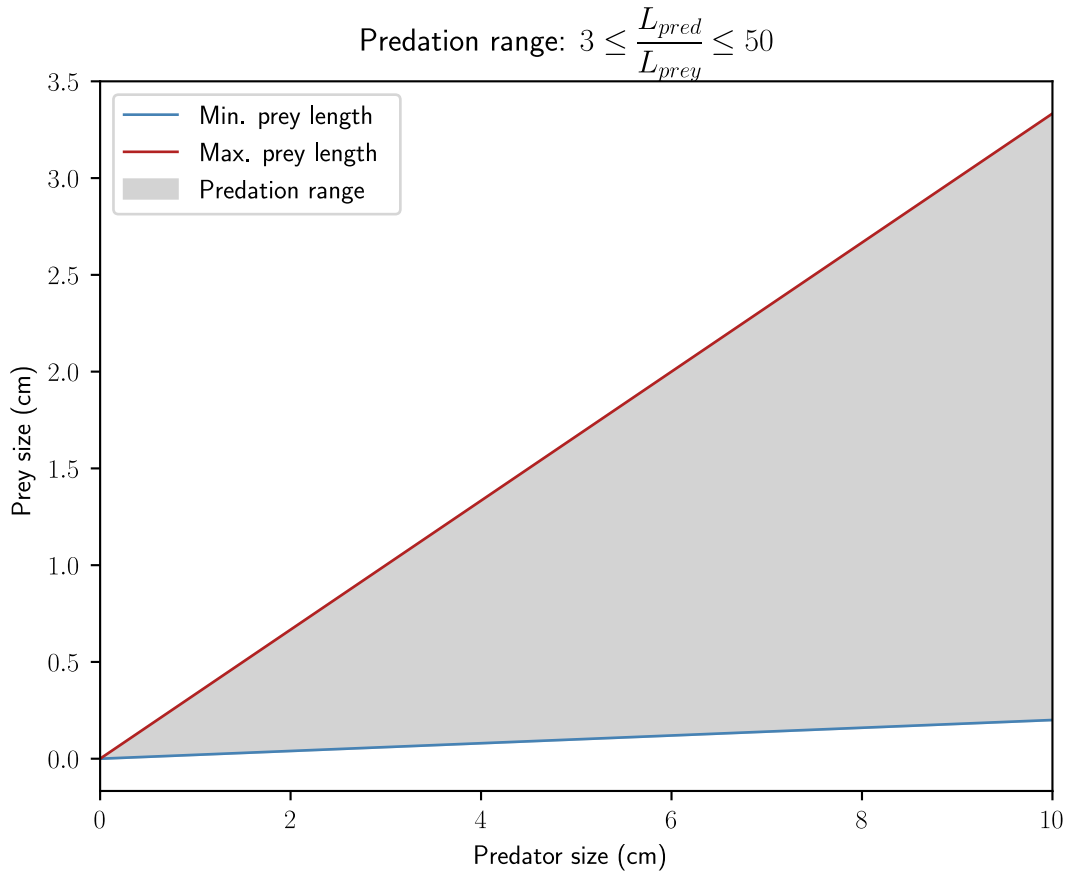


Figure 8.5: Size-based predation in Osmose

(predation-access)=

8.5.1.2 Accessibility

First, the accessibility of all the preys to a given school is determined from an accessibility matrix for every species and stages. This matrix must not be used to define diet preferences but rather to take into account for a difference of positions in the water column (meaning some schools might evolve around the same geographical area but never meet because they do not occur at the same depth).

(table-paros-accessfile)=

Prey / Predator	lesserSpotted < 0.45	lesserSpotted	redMullet < 0.25	redMullet
lesserSpottedDogfish < 0.45	0.05	0	0	0.05
lesserSpottedDogfish	0	0.8	0.4	0

Prey / Predator	lesserSpotted < 0.45	lesserSpotted	redMullet < 0.25	redMullet
redMullet < 0.25	0	0.4	0.8	0
redMullet	0.8	0.4	0	0.8
pouting < 0.25	0	0.4	0.8	0
pouting	0	0.8	0.4	0
whiting < 0.25	0	0.4	0.8	0
whiting	0	0.8	0.4	0
Dinoflagellates	0	0.5	1	0
Diatoms	0	0.5	1	0
Microzoo	0	0.5	1	0
Mesozoo	0	0.5	1	0
Macrozoo	0	0.5	1	0
VSBVerySmallBenthos	1	0.5	0	1
SmallBenthos	1	0.5	0	1
MediumBenthos	1	0.5	0	1
LargeBenthos	1	0.5	0	1
VLBVeryLargeBenthos	1	0.5	0	1
backgroundSpecies	0	0	0	0

Each line of the matrix corresponds to a prey (including plankton groups), and each column to a predator. The file must be understood as follow: lesserSpottedDogfish of age class less than 0.45 (line 1) are only accessible to young lesserSpottedDogfish (5%) and old redMullet (5%).

The class thresholds (age or size, defined with the predation.accessibility.stage.structure parameter) that are used to determine which row or column should be used are read directly from the CSV files by matching the < character. It is assumed that if there is no match, no threshold is provided. However, when < is matched, it is assumed that the number that follows is the upper bound of the class.

Furthermore, the column and row order is not important, since a match of the species name is performed.

Additionally, accessibility matrix can vary over time. To do so, one set of parameters must be defined for each accessibility matrix, as done for the parameterization of movements. The keys of these parameters must end with .acc#, with # the number of the accessibility matrix.

Parameter	Description
predation.accessibility.stage.structure	Threshold type. Must be age or size.
predation.accessibility.file	CSV file containing the accessibility matrix if constant over time
predation.accessibility.file.acc#	CSV file containing the accessibility matrix for the accessibility matrix #
predation.accessibility.initialYear	Start year when to use the accessibility matrix #

Parameter	Description
predation.accessibility.finalYear	Start year when to use the accessibility matrix #
predation.accessibility.years.access	End of years when to use the accessibility matrix # (instead of setting initial and final years)
predation.accessibility.steps.access	End of time steps when to use the accessibility matrix #

If the `predation.accessibility.file` (with no `.acc` suffix) is found, Osmose will assume constant predation accessibility matrix.

8.5.1.3 Predation rate

Finally, the predation rate is computed as follows. First, the total accessible biomass for the predator school is computed:

$$B_{avail} = \sum_{p=preys} A(pred, prey) \times B_{prey}$$

where B_{avail} is the total accessible biomass of preys, $A(pred, prey)$ is the accessibility coefficient of the predator over the given prey (cf. {numref}predation-access) and B_{prey} is the biomass of the prey.

The total biomass that a predator can eat is also computed as follow:

$$B_{eatable} = \frac{B_{pred} \times I_{max}}{N_{mort}}$$

with N_{mort} the number of sub-step of mortality processes, B_{pred} the total biomass of predator and I_{max} the maximum ingestion rate for each species, expressed in grams of food per gram of fish and per year. It is assumed that predator eat as much as they can.

The effective biomass that will be eaten by the predator is

$$B_{eaten} = \min(B_{avail}, B_{eatable})$$

The success rate for the given time-step ($S_R(t)$) is incremented by then the value computed for the given sub time-step as:

$$S_R(t) = S_R(t) + \frac{B_{eaten}}{B_{eatable}}$$

Finally, for each prey, the biomass eaten by the predator is given by:

$$B_{lost\ prey} = B_{eaten} \times \frac{A(pred, prey) \times B_{prey}}{B_{avail}}$$

and used to increment the number of dead individuals by predation (nDead attribute).

Table 8.10: Predation mortality parameter

Parameter	Description
predation.ingestion.rate.max.sp#	I_{max} (grams of food per gram of fish and per year)

8.5.2 Starvation mortality

Starvation mortality is applied as follows:

$$N_{starv} = N \times (1 - e^{-M_{starv}})$$

with M_{starv} the starvation mortality rate, which is computed as follows:

$$M_{starv} = M_{max} \times \left(1 - \frac{S_R}{C_{S_R}}\right) \text{ if } S_R \leq C_{S_R}$$

i Note

Starvation mortality rate applied at time step t is based on the predation success computed at time-step $t - 1$

Parameter	Description
mortality.starvation.rate.max.sp#	Maximum rate of starvation mortality :math:M_{\{max\}}
predation.ingestion.critical.sp#	Critical predation success rate :math:C_{\{S_R\}}

8.5.3 Migration mortality

When a school is out of the domain, none of the standards processes (predation, growth, fishing, natural mortality, growth, starvation) apply.

The migration mortality is used to simulate all sources of mortality outside the simulated area. It applies as long as the school is located out of the simulated area

$$N_{out} = N \times (1 - e^{-M_{out}})$$

Parameter	Description
mortality.out.rate.sp#	Annual mortality rate for species that move out of the simulated area (M_{out})

8.5.4 Fishing mortality

In Osmose, there are two implementations of the fishing mortality:

- One in which fishing mortality is by-species. It is the implementation that has been used so far. In this implementation, bycatch, discards and size-selectivities are not taken into account.
- One in which fishing mortality is by-gear. In this case, bycatch and discards can be taken into account through a catchability matrix. Additionally, size-selectivity can be explicitly considered.

8.5.4.1 By-species fishing mortality

On Osmose versions **previous** to 4.0.0, fishing mortality was species-specific. It was parameterized either by providing fishing rates or catches.

Parameter	Description
mortality.fishing.type	Whether fishing mortality is provided as :samp:rate or :samp:'catches'
mortality.fishing.recruitment.age.sp#	Age at which a species can be fished (years)
mortality.fishing.recruitment.size.sp#	Size at which a species can be fished (cm)

Warning

Osmose does not accept fishing mortality rates for some species and catches for other species.

8.5.4.1.1 By rate

If mortality rate is provided, the number of dead fishes in a school is computed as follows.

$$N_{fishing} = N \times (1 - \exp^{-F})$$

Osmose offers several degrees of refinement for inputting the fishing mortality: constant, seasonal, interannual and interannual with age or size class. Depending on the available information for each species of the configuration, one must choose the best way to input it. Each species can be parameterized independently from an other. For instance fishing mortality for species zero is a constant annual rate and fishing mortality for species three is provided as a time series per size class.

Parameter	Description
mortality.fishing.rate.byDt.byAge.file.sp#	CSV file containing the fishing rates by age and by time-step
mortality.fishing.rate.byDt.bySize.file.sp#	CSV file containing the fishing rates by size and by time-step
mortality.fishing.rate.byYear.file.sp#	File containing the fishing rates by year
mortality.fishing.rate.sp#	Annual fishing rate
mortality.fishing.season.distribution.file.sp#	File containing the seasonal distribution of fishing mortality. If not provided, assumes constant fishing rate

Osmose will first look for any of the first two parameters. If not found, it will look for the third one. If not found, Osmose will finally look for the fourth one. If the third or fourth parameter are found, it will also look for the fifth one.

8.5.4.1.2 By catches

If mortality is provided by catches, the number dead individuals is computed as follows:

$$N_{fish} = \min \left(N, C \times \frac{B_{fish}}{B_{fishable} \times W} \right) \text{ if } B_{fishable} > 0$$

with C the catches, B_{fish} the fish biomass, W its weight and $B_{fishable}$ the biomass that can be fished.

Parameter	Description
mortality.fishing.catches.byDt.byAge.file.sp#	CSV file containing the fishing rates by age and by time-step
mortality.fishing.catches.byDt.bySize.file.sp#	CSV file containing the fishing rates by size and by time-step
mortality.fishing.catches.byYear.file.sp#	File containing the fishing rates by year
mortality.fishing.catches.sp#	Annual fishing rate

Parameter	Description
mortality.fishing.season.distrib.file.sp#	File containing the seasonal distribution of fishing mortality

i Note

Catches are assumed to be in tons.

Osmose will first look for any of the first two parameters. If not found, it will look for the third one. If not found, Osmose will finally look for the fourth one. If the third or fourth parameter are found, it will also look for the fifth one.

8.5.4.1.3 Marine Protected Areas (MPAs)

The user can define as many MPA as he wishes.

Parameter	Description
mpa.file.mpa#	File containing the MPA definition
mpa.start.year.mpa#	First year when this MPA is active
mpa.end.year.mpa#	Last year when this MPA is active

The map is a CSV file similar to the movement maps. The CSV file has the same number of lines and columns as the OSMOSE grid. The MPA file must contain 1 where the MPA is defined, 0 elsewhere.

Start year and end year parameters define the time span when the MPA is enabled.

The MPAs are handled within the fishing process. Every time there is a new MPA to be activated or deactivated, Osmose updates the correction factor that will be applied to the fishing mortality rates in order to take into account the uniform redistribution of the fishing effort outside the MPAs.

8.5.4.2 By-gear fishing mortality

8.5.4.2.1 Fishing mortality rates

In the Osmose versions ≥ 4.3 , changes in the parameterization of fisheries have been implemented, although the spirit remains close to the one in versions 4.

Fisheries mortality time-series for each gear are now the product of three components:

- A vector of fishing mortality rates associated with fishing periods, F_{period}

- A vector of seasonality values, which provides the time-variation of the fishing effort during a given season, F_{season}
- A vector of multipliers, which provides a multiplication factor, F_{base} .

Therefore, for a given time-step t , the value of the fishing mortality for a given fleet would be:

$$F(t) = F_{period}(t) \times F_{season}(t) \times F_{base}(t)$$

Since the parameterisation is a bit tricky, a parameter (`fisheries.check.enabled`) allows to control whether the values of F , F_{period} , F_{season} and F_{base} should be saved. It allows to control that the given time-series are as expected.

8.5.4.2.1.1 Fishing base

The fishing base mortality multiplier (F_{base}) is provided as regime shifts.

Parameter	Description
<code>fisheries.rate.base.fsh#</code>	Fishing base for the different regimes.
<code>fisheries.rate.base.shift.fsh#</code>	Years of the regime shifts.
<code>fisheries.rate.base.log.enabled.fsh#</code>	True if fisheries are provided in logscale.

If one value is provided in `fisheries.rate.base.fsh#`, this value will be used during all the simulation. If N values are provided, then the `fisheries.rate.base.shift.fsh#` parameter must contain $N - 1$ values, which are the years of the regime shifts.

8.5.4.2.1.2 Fishing period

There is now the possibility to define a fishing period (F_{period}), i.e. a period when the fishery is active.

Parameter	Description
<code>fisheries.period.number.fsh#</code>	Number of fishing periods within one year (N_{per})
<code>fisheries.period.start.fsh#</code>	Start of the active fishing period (fraction of year, default = 0, Y_{start})
<code>fisheries.rate.byperiod.fsh#</code>	Fishing mortality rate. Must be in $year^{-1}$

Note that the number of values expected in the `fisheries.rate.byperiod.fsh#` parameter depends on Y_{start} .

If $Y_{start} = 0$, then $N_{per} \times N_{year}$ values are expected (one value for each fishing and non-fishing season).

If $Y_{start} \neq 0$, then $N_{per} \times N_{year} + 1$ values are expected.

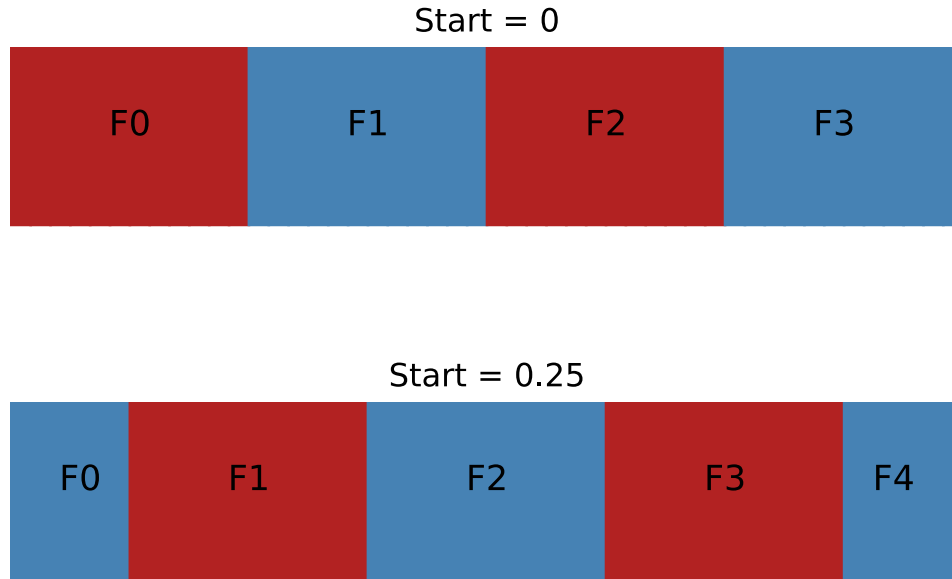


Figure 8.6: Fishing period for two values of Y_{start}

8.5.4.2.1.3 Fishing seasonality

In order to distribute the fishery mortality over the season, the user can define a seasonality vector, either as a file, or as a vector.

Parameter	Description
fisheries.seasonality.fsh#	Array of fishing seasonality. Must contain $\frac{N_{step/year}}{N_{season}}$
fisheries.seasonality.file.fsh#	File containing the fishing seasonalities (must contain N_{step} values)

In the first case, the same seasonality will be applied for each fishing season. Imagine that we have 24 time-steps per year and two fishing season (with no offset, top of figure {numref}fig-fperiod), then the seasonality provided should contain 12 values, which would apply for the active fishing period (green zone).

In the latter case, it is up to the user to generate the proper time series and to store it in a file.

 Warning

The sum of fishing seasonalities must equal one over the fishing seasons! **No automatic normalisation is performed by Osmose!**

8.5.4.2.1.4 Case studies

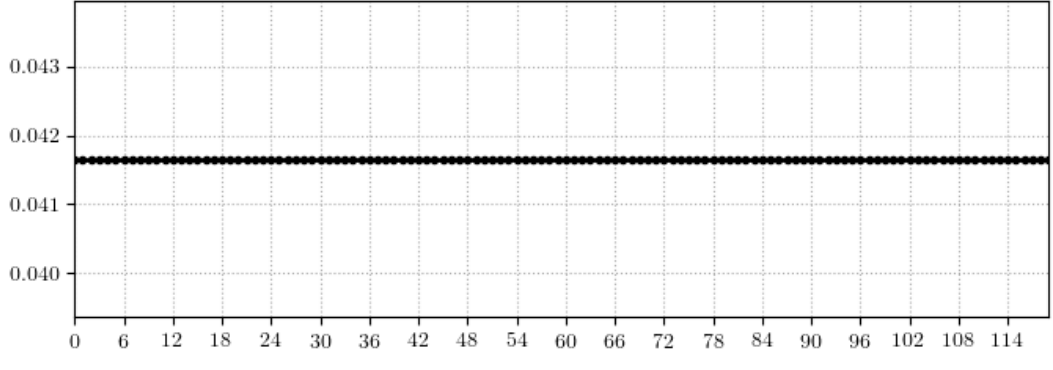
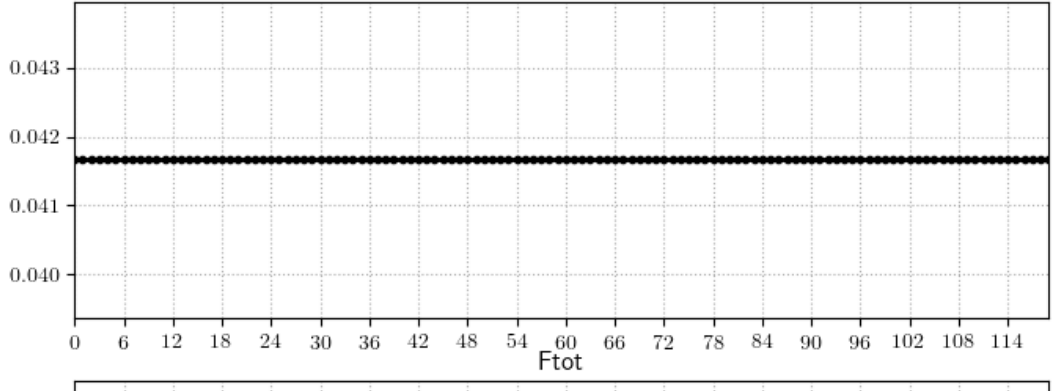
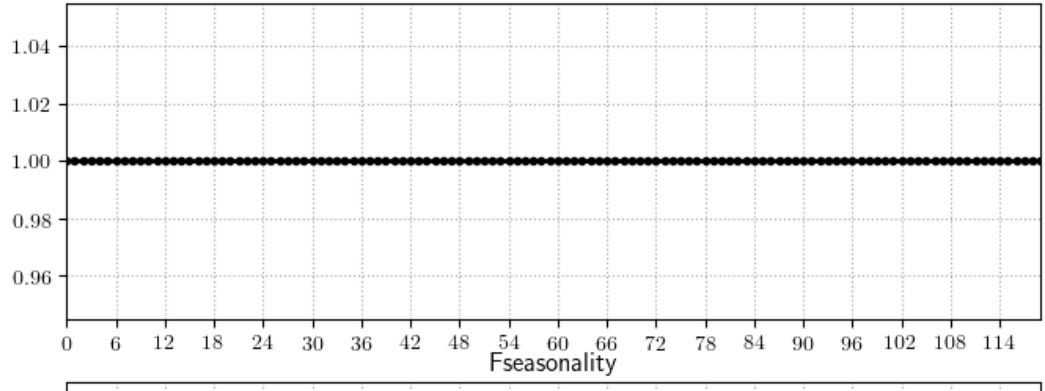
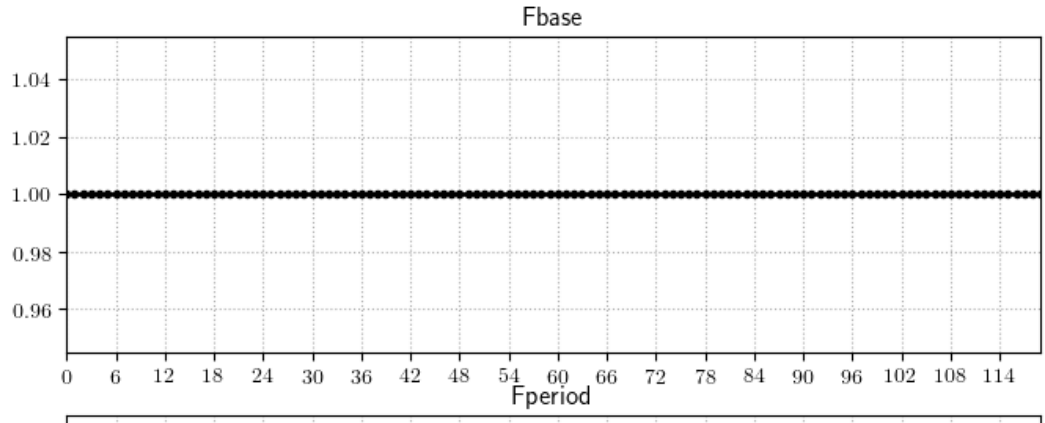
```
fisheries.rate.base.fsh0;1
```

```
fisheries.season.number.fsh0;1
```

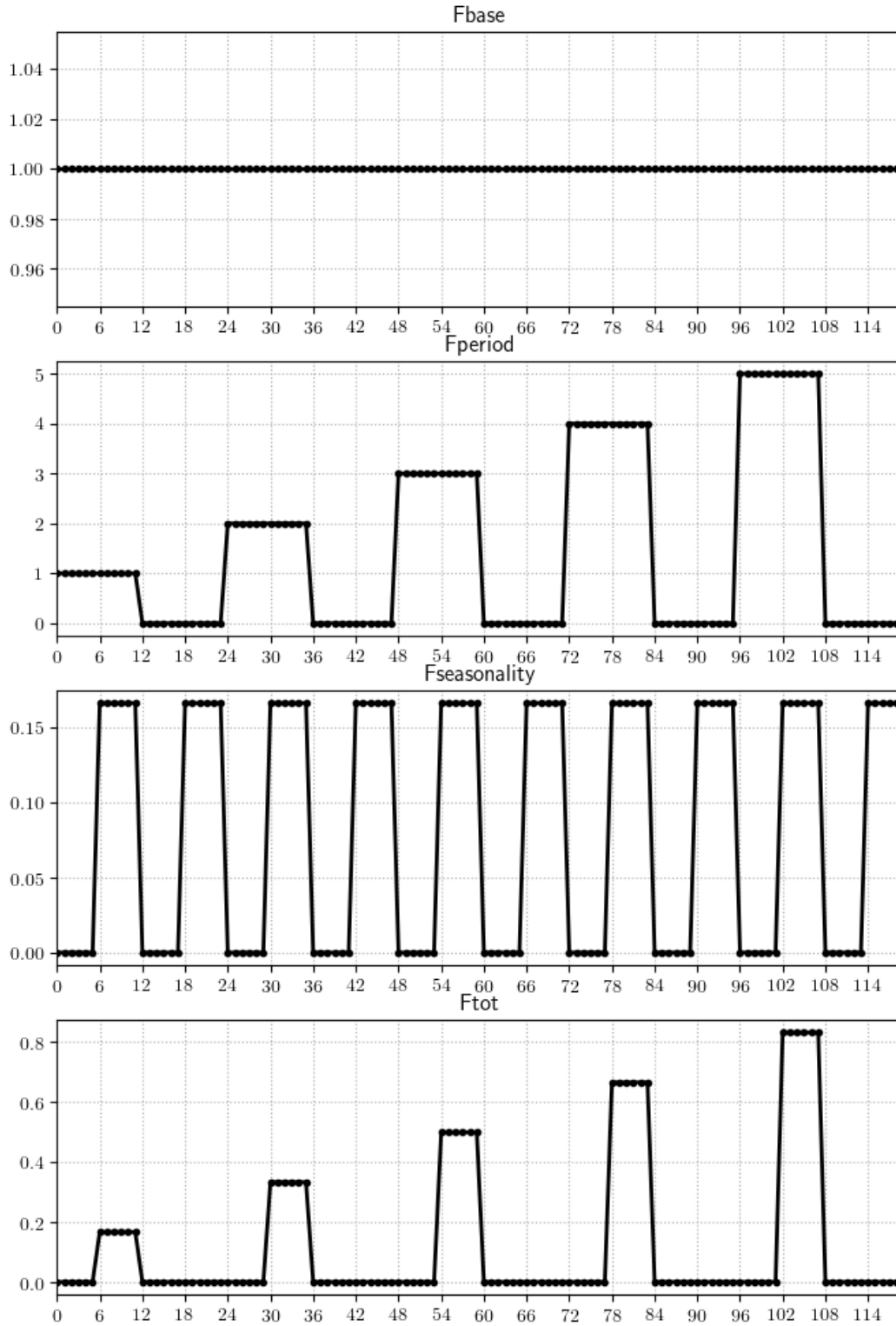
```
fisheries.rate.byperiod.fsh0;1
```

```
fisheries.season.start.fsh0;0
```

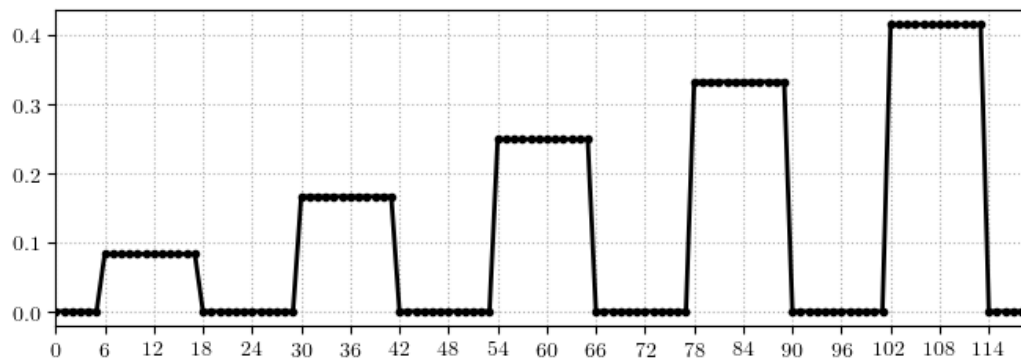
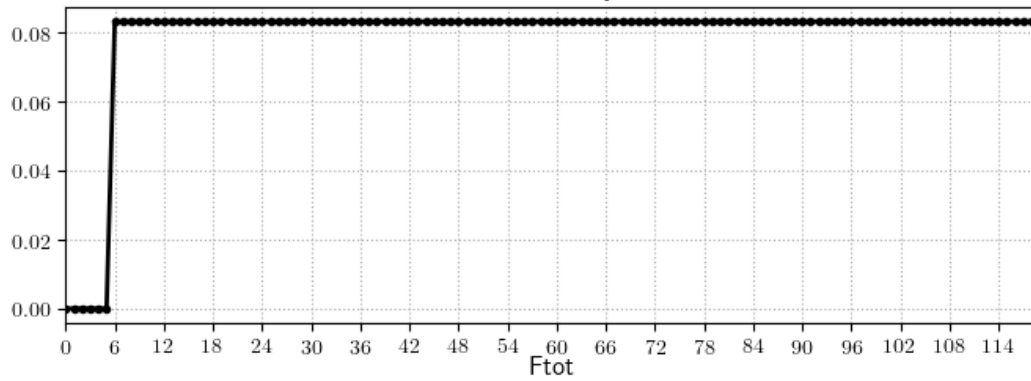
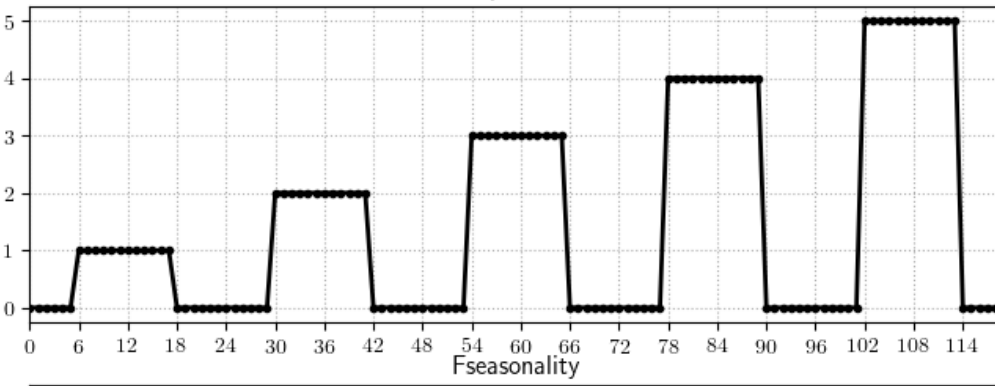
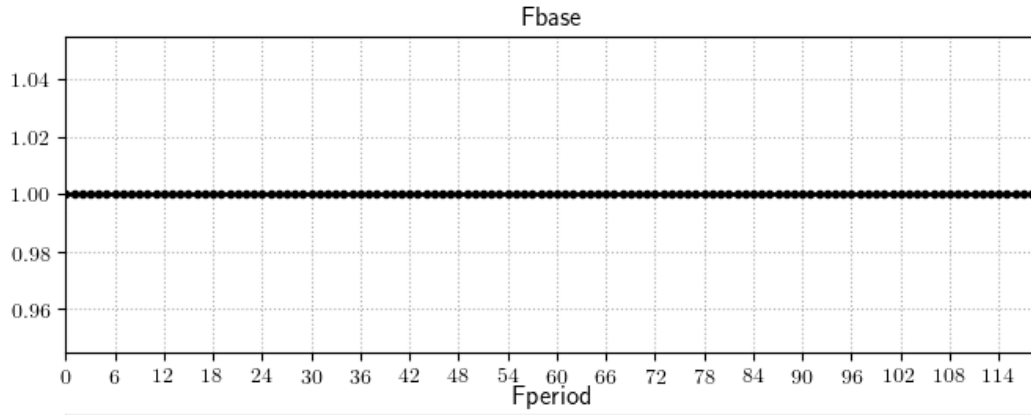
```
fisheries.seasonality.fsh0;0.04166;0.04166;0.04166;0.04166;0.04166;0.04166;0.04166;0.04166;0.04166;0.04166;0.04166;0.04166
```



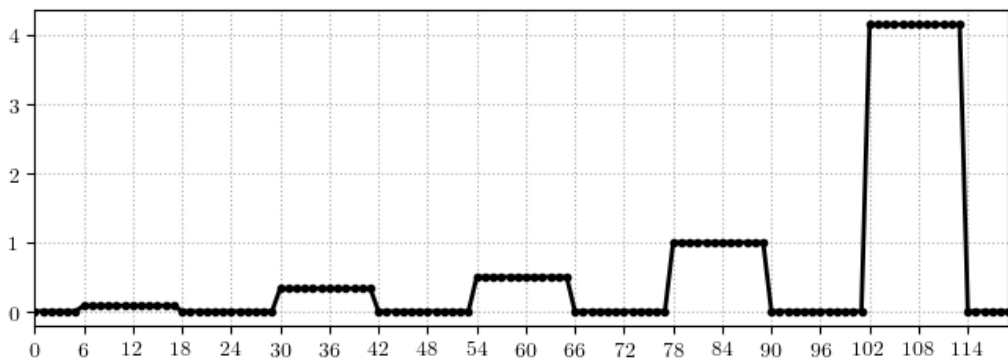
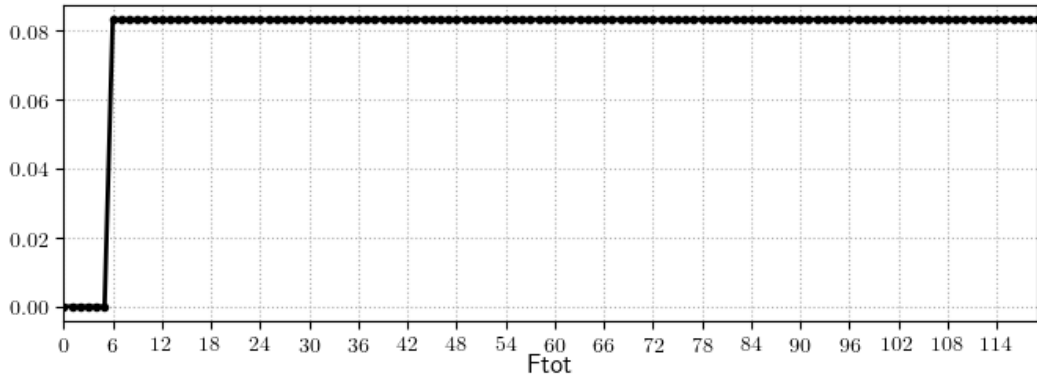
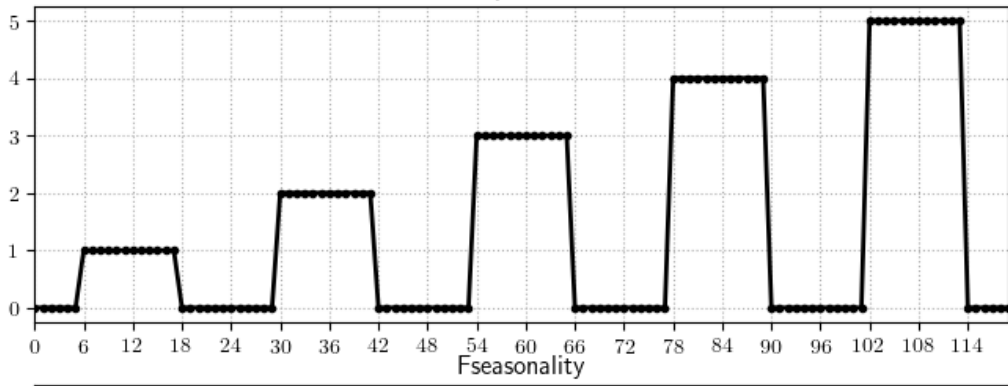
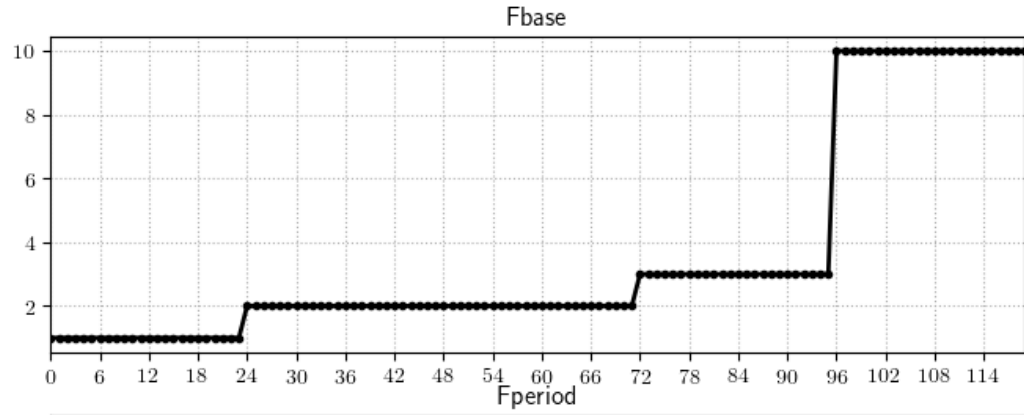
fisheries.rate.base.fsh0;1
fisheries.season.number.fsh0;2
fisheries.rate.byperiod.fsh0;1, 0, 2, 0, 3, 0, 4, 0, 5, 0
fisheries.season.start.fsh0;0
fisheries.seasonality.fsh0;0.0;0.0;0.0;0.0;0.0;0.0;0.1666;0.1666;0.1666;0.1666;0.1666;0.1666;



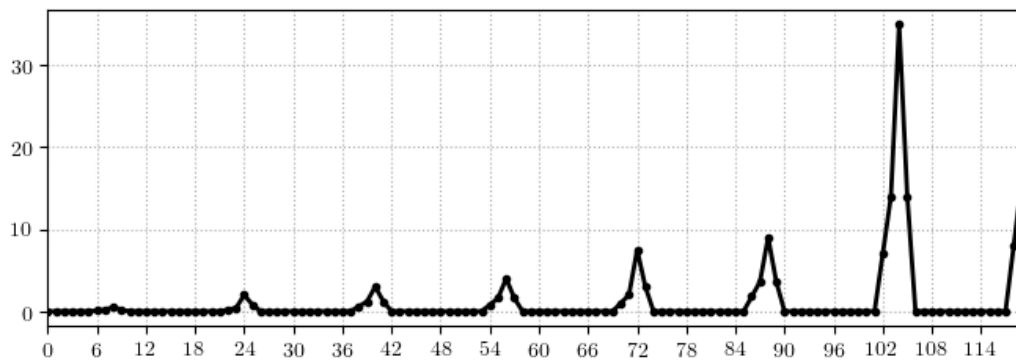
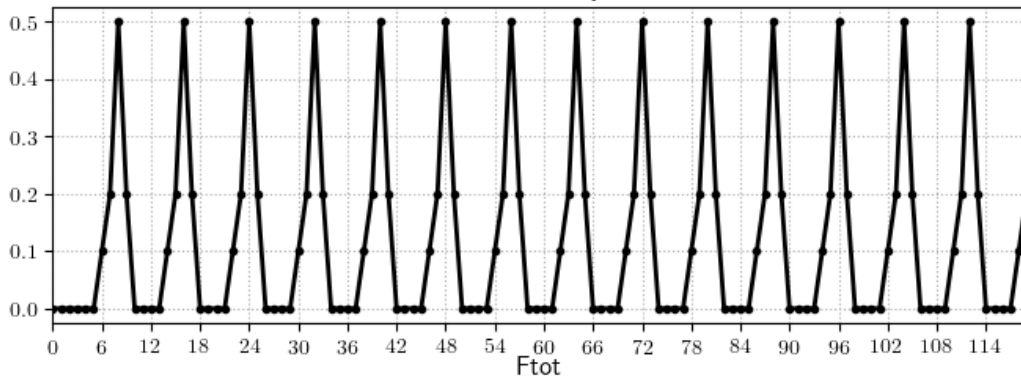
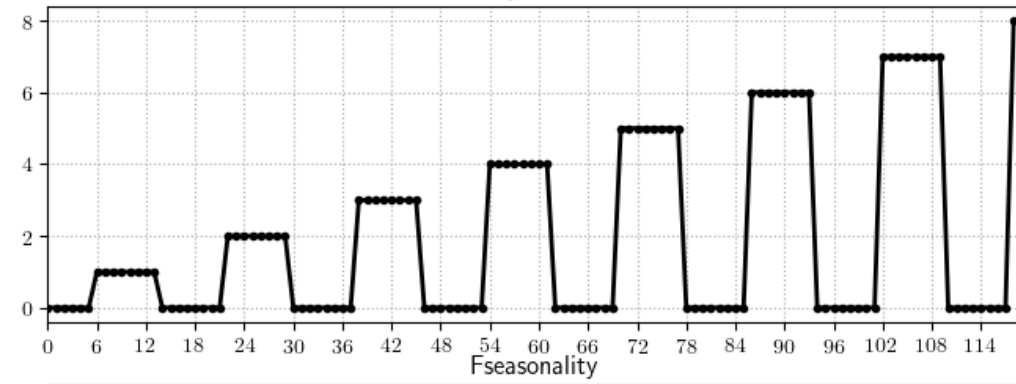
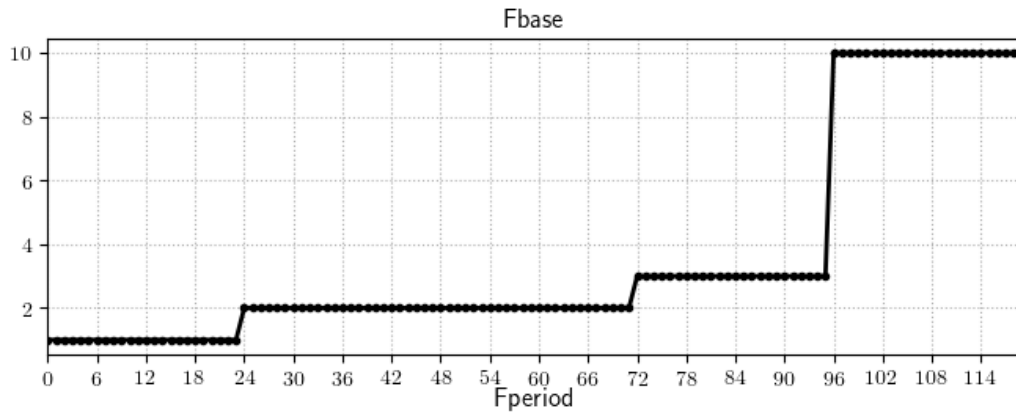
fisheries.rate.base.fsh0;1
fisheries.season.number.fsh0;2
fisheries.rate.byperiod.fsh0;0, 1, 0, 2, 0, 3, 0, 4, 0, 5, 0
fisheries.season.start.fsh0;0.25
fisheries.seasonality.fsh0;0.0833;0.0833;0.0833;0.0833;0.0833;0.0833;0.0833;0.0833;0.0833;



```
fisheries.rate.base.fsh0;1,2,3,10  
fisheries.rate.base.shift.fsh0;1, 3, 4  
fisheries.season.number.fsh0;2  
fisheries.rate.byperiod.fsh0;0, 1, 0, 2, 0, 3, 0, 4, 0, 5, 0  
fisheries.season.start.fsh0;0.25  
fisheries.seasonality.fsh0;0.0833;0.0833;0.0833;0.0833;0.0833;0.0833;0.0833;0.0833;0.0833;
```



```
fisheries.rate.base.fsh0;1,2,3,10
fisheries.rate.base.shift.fsh0;1, 3, 4
fisheries.season.number.fsh0;3
fisheries.rate.byperiod.fsh0;0, 1, 0, 2, 0, 3, 0, 4, 0, 5, 0, 6, 0, 7, 0, 8
fisheries.season.start.fsh0;0.25
fisheries.seasonality.fsh0;0.1,0.2,0.5,0.2,0,0,0,0
```



8.5.4.2.2 Size selectivities

Parameter	Description
fisheries.selectivity.tiny.fsh#	Selectivities values below which selectivity is forced to 0 (ϵ)
fisheries.selectivity.type.file.fsh#	File containing the selectivity types
fisheries.selectivity.type.shift.fsh#	Array containing the selectivity periods
fisheries.selectivity.type.fsh#	Selectivity types (one value per shift period). Must be 0, 1 or 2
fisheries.selectivity.a50.file.fsh#	File containing the age selectivities.
fisheries.selectivity.a50.shift.fsh#	Array containing the A_{50} shift periods
fisheries.selectivity.a50.fsh#	Age selectivity (one value per shift period). If set, assumes that fishery selectivity is age-based
fisheries.selectivity.l50.file.fsh#	File containing the L_{50} .
fisheries.selectivity.l50.shift.fsh#	Array containing the L_{50} shift periods
fisheries.selectivity.l50.fsh#	L_{50} (one value per shift period).
fisheries.selectivity.l75.file.fsh#	File containing the L_{75}
fisheries.selectivity.l75.shift.fsh#	Array containing the L_{75} shift periods
fisheries.selectivity.l75.fsh#	L_{75} (one value per shift period).

Note that type, a50, l50 and l75 are parameterized in the same way. If the .file parameter is defined, then it is used. If it is not set, then values are defined by using the other two parameters. The shift array contains thresholds, where the values are to change.

The selectivity type must contain 0 (knife-edge), 1 (sigmoid), 2 (Gaussian) or 3 (log-normal).

If one of the a50 parameter, it is assumed that age selectivity is used.

Warning

Only knife-edge selectivity can be used with age.

Note

If only knife-edge selectivity is used, then the l75 parameters are not used.

8.5.4.2.2.1 Knife-edge selectivity

Knife-edge selectivity is computed as follows:

$$S(L) = 1 \text{ if } L \geq L_{50}$$

8.5.4.2.2.2 Sigmoid selectivity

Sigmoid selectivity is computed as follows:

$$S(L) = \frac{1}{1 + \exp^{S_1 - S_2 L}}$$

$$S_1 = \frac{L_{50} \times \ln 3}{L_{75} - L_{50}}$$

$$S_2 = \frac{S_1}{L_{50}}$$

8.5.4.2.2.3 Gaussian selectivity

Gaussian selectivity is computed as follows:

$$S(L) = \frac{F(L)}{F(L_{50})}$$

$$F(L) = \exp\left(-\frac{L - L_{50}}{2\sigma^2}\right)$$

$$\sigma = \frac{L_{75} - L_{50}}{q_{75}}$$

with q_{75} is the inverse cumulative standard normal distribution for the 75th percentile.


8.5.4.2.3 Catchability

Fishery catchabilities are parameterized in a similar way as predation accessibility matrix.

Parameter	Description
fisheries.catchability.file	Name of the catchability file
fisheries.catchability.file.cat#	Name of the catchability file
fisheries.catchability.years.cat#	List of years when the catchability should be used.
fisheries.catchability.initialYear.cat#	First year when the catchability matrix should be used (if year list not provided)
fisheries.catchability.finalYear.cat#	Last year when the catchability matrix should be used (if year list not provided)
fisheries.catchability.steps.cat#	List of steps within a year when the catchability should be used.

Fishery catchabilities should be provided as a CSV file, with fisheries as column (predators) and species (background and focal) as rows (preys). If the first parameter (`fisheries.catchability.file`) is found, then this catchability matrix will be used over the entire simulation.

If this parameter is not found, Osmose will assume that catchability matrixes may vary over time. It will therefore look for all the `fisheries.catchability.file.cat#` parameters. Each catchability matrix should be associated with time-indications, which specifies on which year (interannual variability) and which time-steps (seasonal variability) this catchability matrix should be used.

 **Warning**

Note that the # here is not related to the one of fisheries.

8.5.4.2.4 Discards

There is also the possibility to define fisheries discards. It is defined in the same way as catchabilities (cf above for a detailed description of the parameters).

Parameters	Description
<code>fisheries.discards.file</code>	Name of the catchability file
<code>fisheries.discards.file.dis#</code>	Name of the catchability file
<code>fisheries.discards.initialYear.dis#</code>	First year when the catchability matrix should be used
<code>fisheries.discards.finalYear.dis#</code>	Last year when the catchability matrix should be used
<code>fisheries.discards.years.dis#</code>	List of years when the catchability should be used.
<code>fisheries.discards.steps.dis#</code>	List of steps within a year when the catchability should be used.

Fishery discards should be provided as a CSV file, with fisheries as column (predators) and species (background and focal) as rows (preys).

 **Note**

It is possible to mimic the by-species implementation with the by-gear one, by providing a diagonal catchability matrix and knife-edge selectivities

8.6 Growth

Individuals of a given school are assumed to grow in size and weight at a given time only when the amount of food they ingested fulfill maintenance requirements, i.e., only when their predation efficiency at that time is greater than the predation efficiency ensuring body maintenance of school.

$$G(s, a) = M_{\Delta}(s, a) \times \frac{S_R(s, a) - C_{S_R}(s)}{1 - C_{S_R}(s)} \text{ if } S_R(s, a) \geq C_{S_R}$$

$$G(s, a) = 0 \text{ if } S_R(s, a) < C_{S_R}$$

with C_{S_R} is the critical predation efficiency and S_R the predation success rate of the school.

$M_{\Delta}(s, a) = \lambda \times \Delta L(a)$ is the maximum growth rate at age a and for species s .

$\Delta L(a) = L(a+1) - L(a)$ is the mean length increase determined from a growth function (Von Bertalanffy or Gompertz growth function), while λ is a factor that allows to control the maximum length at a given age.

Table 8.23: Growth parameters

Parameter	Description
growth.java.classname.sp#	Class name of the age to length conversion
predation. efficiency.critical.sp#	Critical predation success (C_{S_R})
species.delta.lmax.factor.sp#	λ (default = 2)

8.6.1 Von Bertalanffy growth

When the `growth.java.classname.sp#` is equal to `fr.ird.osmose.process.growth.VonBertalanffyGrowth.java`, a von Bertalanffy growth function is used. **It is the default one.**

$$L(a) = L_{egg} \text{ if } a = 0$$

$$L(a) = L_{egg} + (L_{thres} - L_{egg}) \times \left(\frac{a}{a_{thres}} \right) \text{ if } a > 0 \ \& \ a < a_{thres}$$

$$L(a) = L_{\infty} \times (1 - \exp^{-K(\text{age} - t_0)}) \text{ else}$$

with

$$L_{thres} = \min [L_{egg}, L_{\infty} \times (1 - \exp^{-K(a_{thres} - t_0)})]$$

A Von Bertalanffy model is used to calculate mean length increase above a threshold age a_{thres} determined for each HTL group from the literature. Below a_{thres} , a simple linear model is used. The rationale behind this is that Von Bertalanffy parameters are usually estimated from data excluding

young of the year or including only very few of them. Assuming a linear growth between age 0 and a_{thres} ensures a more realistic calculation of mean length increases for early ages of HTL groups (Travers et al. (2009)).

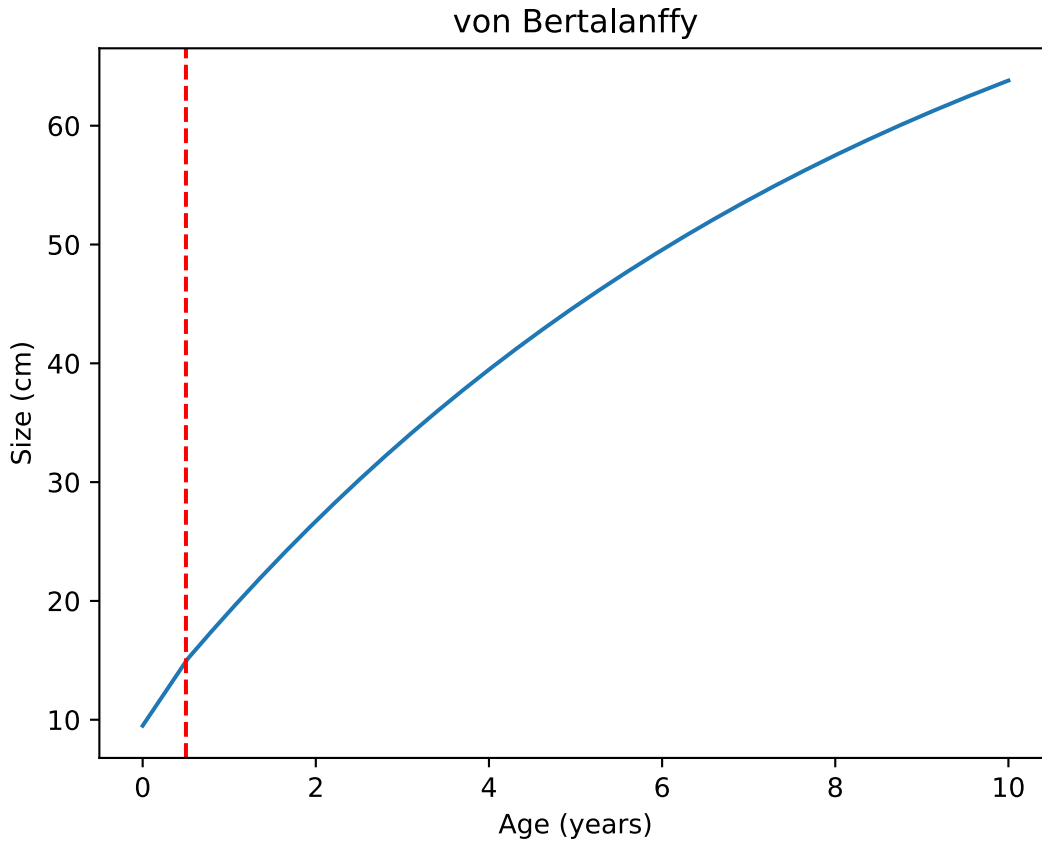


Figure 8.7: Von Bertalanffy growth curve

Parameter	Description
species.linf.sp#	L_{inf} (cm)
species.k.sp#	K
species.t0.sp#	t_0
species.vonbertalanffy.threshold.age.sp#	a_{thres} (years, default=1 year)

8.6.2 Gompertz growth

When the `growth.java.classname.sp#` is equal to `fr.ird.osmose.process.growth.GompertzGrowth.java`, a Gompertz growth function is used.

$$L(a) = L_{egg} \text{ if } a = 0$$

$$L(a) = L_{start} \times \exp^{K_e \times a} \text{ if } a > 0 \text{ \& } a < a_{exp}$$

$$L(a) = L_{exp} + (L_{gom} - L_{exp}) \frac{a - a_{exp}}{a_{gom} - a_{exp}} \text{ if } a > a_{exp} \text{ \& } a < a_{gom}$$

$$L(a) = L_{inf} \times \exp^{-exp^{-K_g(a-t_g)}} \text{ else}$$

with

$$L_{exp} = L_{start} \times \exp^{K_e \times a_{exp}}$$

$$L_{gom} = L_{inf} \times \exp^{-exp^{-K_g(a_{gom}-t_g)}}$$

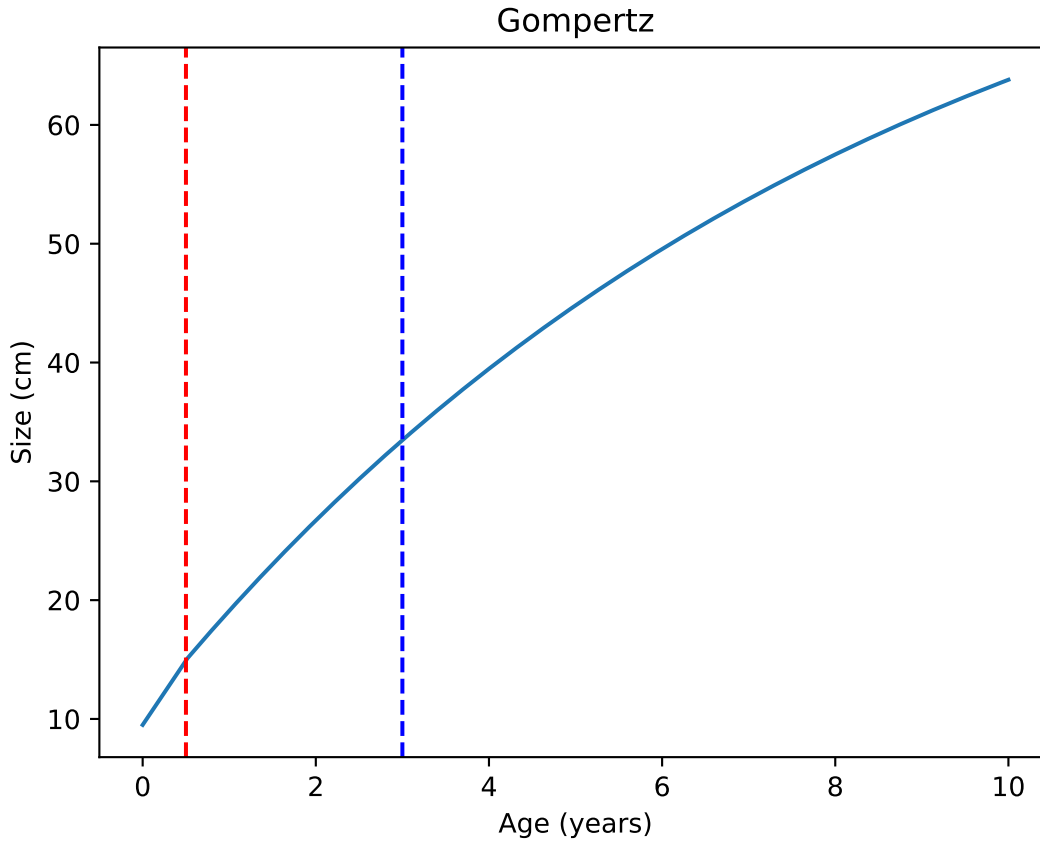


Figure 8.8: Gompertz growth curve

Parameter	Description
growth.exponential.lstart.sp#	L_{start} (cm)
growth.exponential.ke.sp#	K_e
growth.gompertz.linf.sp#	L_{inf} (cm)
growth.gompertz.kg.sp#	K_g
growth.gompertz.tg.sp#	t_g (years)
growth.exponential.thr.age.sp#	a_{exp} (years)
growth.gompertz.thr.age.sp#	a_{gom} (years)

8.7 Reproduction

The system starts from a pristine state, with no schools in the domain. For a few years (user-defined), Osmose will release some eggs for every species. The eggs enter the different steps of the life cycle,

and once the fish reach sexual maturity, the reproduction process takes over. Osmose stops the seeding, unless the spawning stock biomass gets depleted. In that case Osmose resumes the seeding by releasing some eggs until there are again mature individuals in the system for carrying on the reproduction process. Osmose completely ceases the seeding when the simulation reaches the maximal number of years for seeding (user-defined).

For a given species:

$$B_{mat} = \sum_{mature\ sch.} B$$

$$N_{eggs} = FRAC_{fem} \times \alpha \times season \times B_{mat} \text{ if } SSB > 0$$

$$N_{eggs} = FRAC_{fem} \times \alpha \times season \times B_{seeding} \text{ if } SSB = 0$$

By following this approach:

- No assumption is made about the structure of the populations but it emerges from individual interactions
- it reduces computing requirements for the spin-up as the first years are the fastest to run
- it reduces the number of time steps for the spin-up
- it minimizes the amplitude of population oscillations

Finally, the seeding biomass is then used to add new schools to the system, depending on the value of N_{eggs} .

If $N_{eggs} < N_s$:

$$N_{sch} = 1$$

$$A_{sch} = N_{eggs}$$

else:

$$N_{sch} = N_s$$

$$A_{sch} = \frac{N_{eggs}}{N_s}$$

Table 8.26: Reproduction paramters

Parameter	Description
simulation.nschool.sp#	Number of schools of species # to create during reproduction (N_s)
simulation.nschool	Number of schools to create during reproduction (N_s). Used if no species specific value provided
population.seedling.biomass.sp#	Seedling biomass ($B_{seedling}$, tons)
reproduction.season.file.sp#	File providing the seedling distribution within a year
species.sexratio.sp#	Fraction of females ($FRAC_{fem}$)
species.relativefecundity.sp#	Number of eggs per gram of mature female (relative fecundity)
population.seedling.year.start	Number of years when the artificial seeding is activated

References

- Fu, Caihong, R Ian Perry, Yunne-Jai Shin, Jake Schweigert, and Huizhu Liu. 2013. “An Ecosystem Modelling Framework for Incorporating Climate Regime Shifts into Fisheries Management.” *Progress in Oceanography* 115: 53–64. <https://doi.org/10.1016/j.pocean.2013.03.003>.
- Grüss, Arnaud, Michael J Schirripa, David Chagaris, Michael Drexler, James Simons, Philippe Verley, Yunne-Jai Shin, Mandy Karnauskas, Ricardo Oliveros-Ramos, and Cameron H Ainsworth. 2015. “Evaluation of the Trophic Structure of the West Florida Shelf in the 2000s Using the Ecosystem Model OSMOSE.” *Journal of Marine Systems* 144: 30–47. <https://doi.org/10.1016/j.jmarsys.2014.11.004>.
- Halouani, Ghassen, Frida Ben Rais Lasram, Yunne-Jai Shin, Laure Velez, Philippe Verley, Tarek Hattab, Ricardo Oliveros-Ramos, et al. 2016. “Modelling Food Web Structure Using an End-to-End Approach in the Coastal Ecosystem of the Gulf of Gabes (Tunisia).” *Ecological Modelling* 339 (Supplement C): 45–57. <https://doi.org/10.1016/j.ecolmodel.2016.08.008>.
- Oliveros Ramos, Ricardo. 2014. “End-to-End Modelling for an Ecosystem Approach to Fisheries in the Northern Humboldt Current Ecosystem.” PhD thesis, University of Montpellier, France.
- Scheffer, M., J. M. Baveco, D. L. DeAngelis, K. A. Rose, and E. H. van Nes. 1995. “Super-Individuals a Simple Solution for Modelling Large Populations on an Individual Basis.” *Ecological Modelling* 80 (2): 161–70. [https://doi.org/https://doi.org/10.1016/0304-3800\(94\)00055-M](https://doi.org/https://doi.org/10.1016/0304-3800(94)00055-M).
- Shannon, Lynne J, Coleen L Moloney, Astrid Jarre, and John G Field. 2003. “Trophic Flows in the Southern Benguela During the 1980s and 1990s.” *Journal of Marine Systems* 39 (1): 83–116. [https://doi.org/https://doi.org/10.1016/S0924-7963\(02\)00250-6](https://doi.org/https://doi.org/10.1016/S0924-7963(02)00250-6).
- Travers, M., Y.-J. Shin, S. Jennings, E. Machu, J. A. Huggett, J. G. Field, and P. M. Cury. 2009. “Two-Way Coupling Versus One-Way Forcing of Plankton and Fish Models to Predict Ecosystem Changes in the Benguela.” *Ecological Modelling* 220 (“21): 3089–99. <https://doi.org/10.1016/j.ecolmodel.2009.08.016>.
- Travers-Trolet, M, Y-J Shin, and JG Field. 2014. “An End-to-End Coupled Model ROMS-N2P2Z2D2-OSMOSE of the Southern Benguela Foodweb: Parameterisation, Calibration and Pattern-Oriented Validation.” *African Journal of Marine Science* 36 (1): 11–29. <https://doi.org/10.2989/1814232X.2014.883326>.