Creating Individual Based Models of the Plankton Ecosystem

Wes Hinsley[1], Tony Field[1], and John Woods[2]

[1] Imperial College Department of Computing, London, UK
w.hinsley@imperial.ac.uk
[2] Imperial College Department of Earth Science and Engineering, London, UK

Abstract. The Virtual Ecology Workbench (VEW) is a suite of utilities for creating, executing and analysing biological models of the ocean. At its core is a mathematical language allowing individual plankters to be modelled using equations from laboratory experiments. The language uses conventional mathematical assignments and seven plankton-specific functions. A model consists of a number of different plankton species, each with different behaviour. The compiler produces Java classes which when executed perform a timestep-based, agent-based simulation. Each agent is a Lagrangian Ensemble agent [13] representing a dynamic number of individuals, (a sub-population), that follow the same trajectory. The agents are simulated in a virtual water column that can be anchored, or can drift with ocean currents. This paper shows how the language allows biological oceanographers to create models without the need of conventional programming, the benefits of this approach and some examples of the type of scientific experiments made possible.

1 Introduction

There are two main methods for modelling plankton. Population-based modelling involves computing the size of a population of plankters from statistics and rules which apply to whole populations. Such models, first established by Lotka and Volterra [8], are computationally cheap and have thus been until recently the predominant method [1,4]. Alternatively individual-based modelling aims to describe the behaviour of an individual plankter and allow the properties of the population to emerge by integration. Popova et al used a population-based model to show the ocean ecosystem exhibits chaotic behaviour [11], whereas Woods et al with a comparable individual-based model produced stable results [14]. Lomnicki has argued that intra-population variation, unique to individual-based modelling, is the reason the two approaches may give different results. However since individual-based models are harder to code and computationally more expensive, population-based modelling has remained the preferred approach.

The Virtual Ecology Workbench (VEW) is designed to address the difficulties in creating individual-based models. The core of the VEW is the Planktonica language and the compiler which translates equations familiar to a biological oceanographer into Java, thus giving the oceanographer a familiar interface language with which to build models.

Y. Shi et al. (Eds.): ICCS 2007, Part I, LNCS 4487, pp. 111–118, 2007.

2 Creating Models

2.1 Functional Groups and States

A model consists of any number of functional groups which the user creates. A functional group is defined as a set of plankters that behave similarly. They may vary dynamically in terms of their position, volume, mass and any other user-defined property. They may also have parameters that are constant for all plankters of that type. The user creates a set of functions for each functional group. Each function describes some aspect of a plankter's behaviour in response to its biological properties and the properties of its local (ambient) environment. A function can have any number of rules which can be thought of as mathematical statements. These rules may be assignments to variables or calls to one of seven special-purpose functions, which may be executed conditionally or unconditionally.

Two important points must be noted. Firstly when writing functions and rules, the user is considering a single plankter and not a population, nor an agent, even though the simulation is agent-based. Secondly rules are executed each timestep and the user must ensure that the units of time in their rules are appropriate.

A functional group has at least one state, possibly many. These are usually used to represent stages in a life cycle. A plankter exists in one of its states at any time; in each state a subset of its functions are switched on and the remainder switched off.

2.2 The Water Column

The simulation is conducted in a virtual water column stratified into one-metre layers. The user can create any number of chemicals and for each, a concentration variable is automatically created in each layer. Chemicals can have pigmentation properties and when a plankter contains a quantity of such a pigment, the transfer of light and temperature through the water containing that plankter will be affected (called bio-optic feedback).

The physical properties of each layer are generated by a built in physics module, which calculates the irradiance (based on Morel [9]), temperature, salinity and density of water throughout the column. A separate layer structure offering higher resolution in the top metre is used, since this is where the irradiance and temperature change most rapidly with respect to depth.

The physics module also calculates the depth of the turbocline. Above this depth, the water can be assumed to be mixed in each half-hour timestep. The chemicals in solution above the turboclnie are therefore mixed each timestep. Below the turbocline, laminar flow may be assumed. Note that the plankters are not automatically mixed as part of the physics, to allow the user to make some plankters more buoyant than others.

2.3 Variables and Constants

The user can create three types of variable. The first is a 'biological parameter' which is a constant for all plankters of a functional group. The second is a 'biological property' which defines a property of an individual at a given time. It can be assigned a new value in each timestep. Some biological properties are created automatically such as a plankter's depth and internal pool for each chemical the user has created. It also inherits a read-only 'incoming pool' for each chemical which provides the amount of chemical gained in the previous timestep by uptake from the environment, and ingestion of other plankters. Thirdly the user can for convenience create intermediate variables for breaking up a complex equation into parts, or for sharing an intermediate value between other rules or functions.

Other variables may be read by the user but not written. These include ambient physical properties (temperature and irradiance for example), ambient chemical concentrations for each chemical the user has created, and ambient biological properties - the local concentration of any functional group. Additionally, the timestep size in hours and the depth of the turbocline can be read. The latter is needed in order to write motion equations, since it is necessary to know whether the plankter is above or below the turbocline.

2.4 Special-Purpose Functions

Along with basic assignments, there are seven special-purpose functions defined below which facilitate interactions between the plankter and its environment.

Uptake(chemical, amount) and Release(chemical, amount). The uptake call is used when the plankter attempts to absorb chemical from its ambient environment. Requests are proportionally issued over each layer that the plankter visits on the journey between the current timestep and the next, scaled by how long it spends in each layer (or part thereof). In cases where the amount of chemical available is insufficient (since many plankters may try to uptake in the same layer), all the requests will be proportionally scaled down so that they sum to the available amount. The actual amount gained by an individual will be available in its 'chemical-gained' biological property for that chemical. The release call performs the opposite; the specified chemical is released into the environment proportionally in each layer the plankter moves through.

Divide(x) and Create(x, state, [assignments]). These two functions model cell division and reproduction respectively. The two are treated separately because cell division offers a useful optimisation, namely that having divided into two, the two parts will be identical. When the user writes divide(x), they are stating that the plankter should divide into 'x' indentical parts. Since cell divisions among diatoms are extremely common in Spring, creating new a plankter agent for each division would be costly on memory. Instead the sub-population size of the agent is multiplied by 'x'.

The create function by contrast creates a new agent with its own trajectory. The agent represents a number of plankters with the same biological properties, but they may differ from those of the parent. The number of offspring is defined by 'x'; this is the number of individuals that the new agent will represent. The state of the offspring is defined by 'state' and a list of assignments may be provided to initialise the biological properties of the offspring.

Change(state) and Pchange(p,state). Two methods of changing state are provided. The 'change' function causes an unconditional change from the current state to a newly specified one. When the user writes the 'pchange' function they are stating there is a probability 'p' that the plankter should change into its new state. Planktonica interprets this as splitting the agent into two, one representing the proportion 'p' that did change and the remainder are left in their original state.

Ingest(foods, rates, thresholds). The final function call is for plankters that perform ingestion. Any number of foods may be selected, where a food is defined by its functional group and stage. Each food may be ingested at a different 'rate', in individuals per second, provided that the concentration of prey is above a specified 'threshold'. More specifically, the type of food is specified by 'species' and stage; species will be defined in the following section.

Like the uptake call, requests are made in each layer that the plankter will pass through in the next timestep. If there is insufficient food of any of the types, the request for that food-type will be scaled down. The sub-population sizes of the prey that were ingested will be reduced accordingly and the chemicals in the pools of the ingested plankters will be transferred to the chemical-gain biological property of the predator. A further system variable is available called 'ingested', which returns the number of individuals of each food-type that were ingested.

3 Further Specification

3.1 Species

Having created the functional groups and chemicals the user creates at least one species of each functional group. A species is a parameterisation of a functional group in which a value is given for each of the biological parameters for that functional group. A common example is to create a number of size-based variations of a functional group so that the species all exhibit the same behaviour, but with different size-dependent rates. Having defined the species the user can now specify the foods, rates and thresholds for each of the ingestion calls they may have made in their model.

3.2 Scenario

The scenario defines the input data for the simulation. This includes the starting time and duration of the simulation and the track that the water column takes.

The column may be anchored at a location or it may drift in response to ocean currents, supplied by OCCAM [10]. Data is created for each timestep, providing the astronomical and climatological values required by the physics code. This data can be taken from Bunker [2] or the ERA40 data set [3]. The physical and chemical profiles of the water column also must be initialised. For some chemicals, data is available from the Levitus World Atlas [6].

3.3 Agent Initialisation and Management

Initial distributions of plankters are then configured; any number of sets of plankters may be distributed at a given density with a given set of initial values for their biological properties and an initial state. Agent management rules can then be specified which allow the user to control the compromise between demographic noise and computational cost. Two rule types are available. The first allows splitting of the largest agents (that is, agents with the largest sub-population size), should the number of agents of a given species and stage fall below a given limit. The second allows merging of the smallest agents if the number of agents exceeds a given limit. Both of these can be specified to apply to each one-metre layer of the column, or to the column as a whole.

3.4 Logging Options

The output of the simulation can include water column properties such as the total number of plankters of each species and stage, or field data can be recorded layer-by-layer for all or part of the column, such as concentrations of plankters, concentrations of chemical (in solution or particulate form), and physical properties such as temperature and irradiance. Audit trails, which are values of the biological properties of individual plankters and their ambient environmental properties may be plotted. These are unique to individual-based modelling, and not easily observable in nature. Additionally, demographic statistics can be plotted which show how many plankters changed state due to a particular function and the location of those plankters in the column. Finally, for debugging purposes the local variables the user created can be logged.

4 Compilation and Execution

A single XML file stores the model and all the associated specification information. The compiler then produces Java classes from the specification file, for each functional group and chemical. These are then compiled by the standard Java compiler along with a set of kernel classes which drive the simulation. The climate data specified in the scenario is extracted from the available datasets and written to binary files. All the files are then packaged into a single executable JAR file.

Execution can be done silently, or alternatively a utility called ‘LiveSim’ is provided for running the simulation step by step and inspecting every property

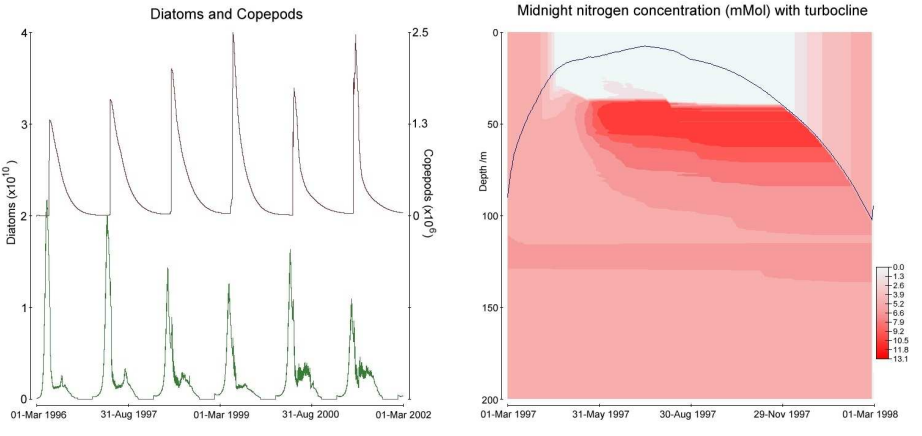
of the system visually. This is useful for debugging models and as an interactive teaching tool to explain the behaviour of plankters. The VEW contains a documentation tool which writes all the information necessary to recreate the model in an indexed HTML format.

5 Applications and Example Results

The modelling language was designed with reference to the WB model [13], an individual based containing Nitrogen, Phytoplankton, Zooplankton and Detritus (NPZD). The VEW has been recently used to develop the Lagrangian Ensemble Recruitment Model (LERM), which includes diatoms, copepods and squid larvae in an environment with nitrogen and silicate [12].

Below are some of the typical plots available using the VEW's own Analyser package, taken from a simulation of the WB model [13]. Figure 1(a) shows the yearly cycle of diatoms and copepods. The diatoms bloom in Spring until they run out of nutrient. The copepods then grow by feeding on the diatoms until they can reproduce. The parent copepods then die of old age and the next generation will have to wait until the following year until there is enough food for them to grow to reproductive size.

Figure 1(b) is an example plot of field data. It shows the nitrogen concentration varying over depth and time, with the turbocline marked. Through Summer nitrogen near the surface is depleted by the diatoms, which are nitrogen-limited. In Autumn, the turbocline descends and nitrogen from deeper water is mixed above the turbocline. Copepods excrete nitrogen and dead diatoms, dead copepods and copepod pellets remineralise nitrogen.



(a) **(b)**

Fig. 1. (a): Diatom and Copepod yearly cycles. (b): Nitrogen throughout the column at midnight with turbocline.

Figure 2 shows the unique characteristic of individual-based modelling. A single diatom's energy pool is plotted along with its depth. The more intense the diatom's ambient irradiance, the higher its energy gain by photosynthesis will be, whereas in relatively dark water, or at night, respiration losses outweigh energy gains. Note also the effect of the turbocline on the depth of the plankter; below the turbocline it sinks, whereas above the turbocline it is randomly displaced.

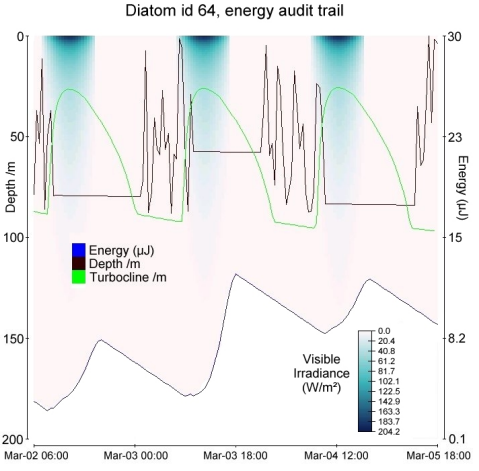


Fig. 2. Energy of an individual diatom

6 Conclusion

The Virtual Ecology Workbench with the Planktonica language at its core offers the best known method for creating individual-based plankton models. The equations for the WB model above can be shown succinctly in a few pages of equations, whereas conventional methods were only able to produce raw computing source code. The latter gave rise to inherent problems of maintainability, requiring a mediator between the biological modeller and the simulation code, as few biological modellers are skilled programmers.

While it is necessary for the modeller to familiarise themselves with the special function calls and to take the necessary care in ensuring their equations are correct and input correctly to the system, the amount of pure computing skill required has been significantly reduced.

While individual-based models offer advantages over population-based models [7,14], the difficulty in constructing such simulations has limited research into the precise nature of the them. The VEW now offers a convenient method for building these simulations, and investigating more thoroughly the issues regarding these two approaches to modelling plankton.

References

1. Anderson, T.R., Pondaven, P.: Non-redfield carbon and nitrogen cycling in the Sargasso Sea: pelagic imbalances and export flux. Deep-Sea Research I **50** (2003) 573–591
2. The Bunker Climate Atlas of the North Atlantic Ocean. http://dss.ucar.edu/datasets/ds209.2/
3. The ERA-40 Re-analysis Dataset. http://www.ecmwf.int/research/era/
4. Fasham M.J.R., Ducklow H.W., McKelvie S.M.: A nitrogen-based model of plankton dynamics in the oceanic mixed layer. Journal of Marine Research **48** (1990) 591–639
5. Hinsley W.: Planktonica: A System for Doing Biological Oceanography by Computer. PhD Thesis, Imperial College Department of Computing. (2005)
6. The Levitus World Ocean Atlas. http://www.cdc.noaa.gov/cdc/data.nodc.woa98.html
7. Lomnicki, A.: Individual-based models and the individual-based approach to population ecology. Ecological Modelling **115** (1999) 191–198
8. Lotka, A.J.: Elements of physical biology. Baltimore: Williams & Wilkins co. (1925)
9. Morel, A.: Optical modelling of the upper ocean in relation to its biogenous matter content (case 1 water). Journal of Geophysical Research **93(c9)**, 10749
10. The OCCAM global ocean model. http://www.noc.soton.ac.uk/JRD/OCCAM/
11. Popova E.E., Fasham M.J.R., Osipov A.V.O., Ryabchenko V.A.: Chaotic behaviour of an ocean ecosystem model under seasonal external forcing. Journal of Plankton Research **19:10** (1997) 1495–1515
12. Sinerchia M.: Testing Fisheries Recruitment Models. Ph.D in preparation, Imperial College Department of Earth Sciences and Engineering. (2006)
13. Woods, J.D.: The Lagrangian Ensemble metamodel for simulation plankton ecosystems. Progress in Oceanography **78** (2005) 84–159
14. Woods, J.D., Perilli, A., Barkmann, W.: Stability and predictability of a virtual plankton ecosystem created with an individual-based model. Progress in Oceanography **67** (2005) 43-83