Environmental Modelling & Software 65 (2015) 67–78

Contents lists available at ScienceDirect

Environmental Modelling & Software

journal homepage: www.elsevier.com/locate/envsoft

Four-dimensional connectivity modelling with application to Australia's north and northwest marine environments

Johnathan T. Kool*, Scott L. Nichol

Geoscience Australia, GPO Box 378 Canberra ACT 2601 Australia

ARTICLE INFO

Article history:
Received 5 June 2014
Received in revised form
20 November 2014
Accepted 21 November 2014
Available online

Keywords:
Connectivity
Marine
Dispersal
Individual-based model
Seascape
Australia

ABSTRACT

A fully four-dimensional (3D × time) object-oriented biophysical dispersal model was developed to simulate the movement of marine larvae over semi-continuous surfaces. The model is capable of handling massive numbers of simulated larvae, can accommodate diverse life history patterns and distributions of characteristics, and saves point-level information to a relational database management system. The model was used to study Australia's northwest marine region, with attention given to connectivity patterns among Australia's north-western Commonwealth Marine Reserves (CMRs). Animations of larval movement near the Gascoyne canyon CMR, dispersal surfaces over depth and time for CMRs and Key Ecological Features in the northwest, as well as matrices of connectivity values among CMRs are shown. The matrices are further analysed to identify the sensitivity and elasticity of their values. The results generated by this model can aid in designing and managing marine protected area networks that incorporate extensive and complex benthic terrain (including the identification of marine 'corridors'), and for developing targeted field sampling strategies.

Software and/or data availability

The source code is available under BSD-3 licensing on Github at https://github.com/GeoscienceAustralia/Conn4D.

1. Introduction

Population connectivity research involves investigating the presence, strength and characteristics of spatial and temporal relationships between populations (Kool et al., 2013). These data can be used in many different ways: to identify source-sink relationships between populations (Cowen and Sponaugle, 2009); to detect critical pathways or keystone habitats (Treml et al., 2008); to find natural clusters or biogeographic regions (Jacobi et al., 2012; Kool et al., 2010, 2011); or to investigate the processes underlying population genetic structure (Foster et al., 2012; Kool et al., 2010, 2011), among others. This information can be of significant value for managers and decision-makers when designing reserve networks (Costello et al., 2010), evaluating the potential spread of invasive species (Johnston and Purkis, 2011), or when confronted with transboundary management issues (Worboys and Pulsford, 2011). However, empirically measuring connectivity is challenging; species such as fish or invertebrates typically release large numbers of tiny young over long time scales relative to the variability of the system, and the problem is compounded when taking into account seasonality, behaviour, or when studying connectivity over long periods of time (e.g. years or more). Researchers have therefore turned to numerical simulations, coupling models of advection patterns with organism behaviour to develop testable hypotheses, and to inform targeted sampling efforts.

A wide variety of studies have been performed and models developed to investigate marine dispersal. Examples include simulating the dispersal of coral reef fish (Andrello et al., 2013; Cowen et al., 2006; DeMartini et al., 2013; Paris et al., 2007), mussels off the coast of England (Gilg and Hilbish, 2003), and the cross-shelf transport of pelagic fish (Condie et al., 2011), as well as evaluating dispersal patterns for coral reef communities at a global scale (Wood et al., 2014). Full modelling frameworks have also been developed – for example, the Connectivity Modeling System (Paris et al., 2013), CONNIE (Condie et al., 2005), Ichthyop (Lett et al., 2008), LTRANS (North et al., 2011), MIKE (Hydraulics, 2005), and the dispersal routines of MGET (Roberts et al., 2010). However, much of this work has focused on shallow water species and effectively 2-dimensional environments (e.g. mussels, barnacles

---

* Corresponding author.
E-mail address: johnathan.kool@ga.gov.au (J.T. Kool).

http://dx.doi.org/10.1016/j.envsoft.2014.11.022

and coral reef communities) emanating from point sources or discrete patches, or at sub-regional scales. Yet, there are also many important benthic species such as crustaceans, molluscs, and echinoderms that occupy large swaths of complex semi-continuous benthic terrain (e.g. ridges, banks, shoals and canyons). Accommodating these kinds of life histories introduces additional complexities into the modelling framework, such as the need to explicitly address 3-dimensional movement and topographic interactions, as well as the need to generate, manage and analyse the massive numbers of particle locations associated with semi-continuous releases in three dimensions.

To address these challenges, we have developed a fully four-dimensional (3D × time) open-source, object-oriented biophysical dispersal model that is capable of volumetric releases and tracking simulated particles over extensive semi-continuous surfaces. This paper describes the model and presents results from processed output data for north western Australian Commonwealth waters. Additionally, we provide and describe methods for querying and analysing the data set. Although the model was developed to simulate the dispersal of marine larvae, the code can also be used to simulate other types of dispersal as well, such as petroleum spills or seeps (North et al., 2011; Paris et al., 2013), drifting debris, or wind-borne dispersal.

2. Methods

The model (Conn4D) was developed using the Java programming language (v. 1.7). Using Java as a development environment offers a number of benefits, including object-oriented classes and inheritance, concurrency through multithreading, platform independence (i.e. the executable will work on any operating system that supports a Java Virtual Machine), the availability of a wide variety of existing software libraries, as well as automated reclamation of memory resources from program objects no longer in use (garbage collection). The model simulates Lagrangian dispersal (Siegel et al., 2003) through the application of advection and diffusion, while also performing collision detection with the benthic boundary. Releases are performed in a semi-continuous manner, such that all areas at all depths are sampled according to a three dimensional stratified-random design.

The main method of the *Conn4D* class (italics indicate program objects) is the principal entry point into the model (Fig. 1). The *Conn4D* class reads in parameter data provided in input files to set up a *ReleaseFactory* instance, which is used to generate individual *Release* instances. The *Release* class is the primary operational unit of the model, and contains the variables and behaviour associated with the path travelled by an individual generic *Particle* object over its lifespan. The *ReleaseFactory* also generates individual *Release* instances as threads, allowing the program to take

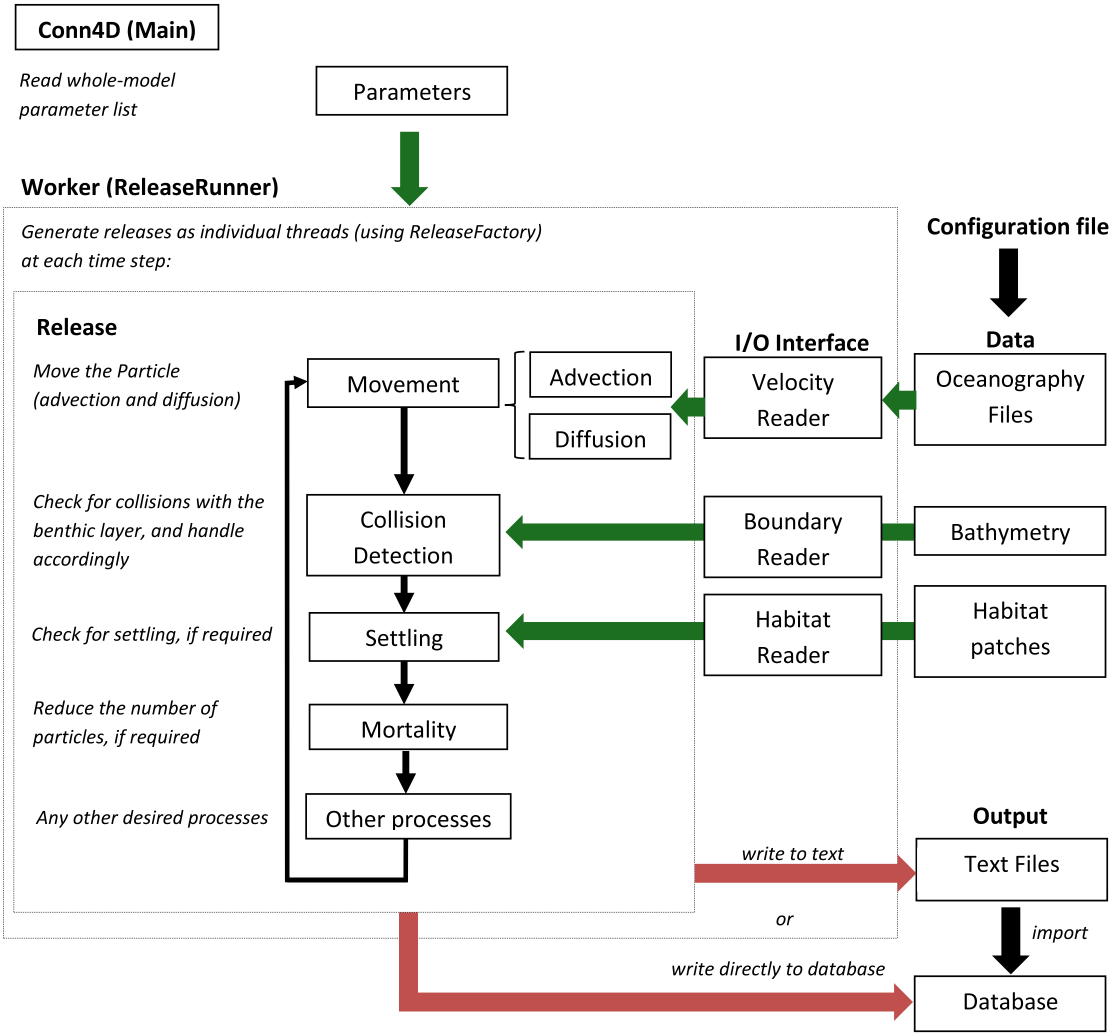

Fig. 1. Model objects and linkage flow diagram for Conn4D, showing the processing flow of the dispersal code. The logic follows a master-worker/map-reduce pattern, with the main *Conn4D* class sending a list of parameters to *ReleaseRunner* instances, which use the *ReleaseFactory* class to generate individual Release threads. Each *Release* applies *Movement*, *Collision Detection*, *Settlement* behaviour and *Mortality* to *Particles* corresponding to the *Release*. Other processes can be added into the release procedure if required, and the various steps can be re-ordered as well. Data files (e.g. oceanography, bathymetry, habitat) are linked to the objects through their corresponding *Reader* classes. All of the classes shown here are interfaces, meaning that different implementations can be used without needing to modify the structure of the underlying code. Output can be written to text files and then imported into a database, or written directly into the database.