The ODD Protocol for a *NetLogo* Model of

*C. Elegans* Population Dynamics

Wei Tao

1. **Purpose**

The model is designed to simulate and explore the behaviors and population dynamics of nematodes, in particular, *Caenorhabditis elegans*. The goal is to determine the relationship between genetically encoded life history traits of individuals and the emergent properties of population dynamics.

1. **Entities, state variables, and scales**

There are two entities in this model: worms (*C. elegans*) and square patches of liquid environment which make up a square grid flask of 57 \* 51 patches. Every patch has one state variable: concentration, which represents the concentration of bacteria (food) within said patch. Each patch’s concentration is represented using a grey scale of the black color, ranging from black (RGB: 255, 255, 255; no food) to white (RGB: 0, 0, 0; full food).

Worms are characterized by their life stage (larva, adult and worm-bag), energy reserve, size and number of progeny and age, which is in turn divided into three parts: age as a larva, age as an adult and age as a worm-bag. Worms also possess a generation number, which indicates which generation the specific individual belongs to (with the initial adults as the 0th generation).

Currently, the model remains generic. Therefore, the patch size and length of one time step in the simulation have not been specified. However, when real experimental scenarios are involved, the patch sizes and time steps will be assigned accordingly. Tentatively, a tick would represent 1/5 of a day. Patches are not differentiated and the concentration remains uniform over all patches. Overall, the patches make up a 5 ml flask.

1. **Process overview and scheduling**

The processes involved in this model include the following:

* *Move*: worm’s movement mechanism. On each time step, each adult and larva moves once. The order in which the agents execute this action is *NetLogo* default (random-order).
* *Reproduce*: worm’s reproduction process. On each time step, each adult-stage worm determines if it has enough energy for this action, and if yes, carries out this action once. The order in which the agents execute this action is *NetLogo* default (random-order).
* *Catch bacteria*: worm’s feeding mechanism. On each time step, each adult and larva feeds once. The order in which the agents execute this action is *NetLogo* default (random-order).
* *Grow:* worm’s stage transition. This action only applies to larvae. On each time step, larvae that have size larger than the size threshold perform this action. The order in which the agents execute this action is *NetLogo* default (random-order).
* *Death:* worm’s dying mechanism, determines if a particular worm is dying and performs the dying mechanism. On each time step, all adults and larvae carry out this method. The order in which the agents execute this action is *NetLogo* default (random-order).
* *Worm-bag:* transition of adult worm into worm-bag, determines if the stress an adult worm is under is going to cause said worm to bag and performs the transition. On each time step, all adult worms carry out this method. The order in which the agents execute this action is *NetLogo* default (random-order).
* *Explode:* worm-bag explosion mechanism, determines if a worm-bag has met the conditions to explode, and if yes, kills the worm-bag while hatching larvae. On each time step, all worm-bags carry out this method. The order in which the agents execute this action is *NetLogo* default (random-order).

1. **Design concepts**

* *Basic principles*: the model addresses the effects of ageing on population dynamics. Most current theories on ageing include ageing as a maladaptive trait while some consider it as an adaptive trait. Our hypothesis is that at a population level, ageing is an adaptive property. In the future, we will incorporate an aspect of competition into the model so that we can understand the relationship between ageing and fitness.
* *Emergence*: the emergent properties of this model include the population size, spatial distribution, age distribution and their change over time.
* *Adaptation*: the model will provide an approach for us to study *C. elegans*’ adaptation behaviors including feeding, growth and reproduction. The main goal is to study the adaptation in terms of population dynamics, i.e. population size, age distribution and individual life span.
* *Objectives*: the model aims to create an environment in which we can measure the competitive ability, which is in turn the measure of fitness.
* *Learning*: in the current model, the agents do not exhibit learning behaviors.
* *Prediction*: if ageing is adaptive, worms with a life span closest to the optimal life span (given the environment) will out-compete the ones with life spans either higher or lower than the optimal life span.
* *Sensing*: individual worm’s decisions on their growth, reproduction, and death are based on their age and energy reserve they possess at the given point.
* *Interaction*: there is one direct interaction between worms: parents generate progeny. There is one indirect interaction: competition for food.
* *Stochasticity*: the actions performed within each time step are executed in a pseudo-random order per *NetLogo* default. Each “ask” call generates a new random order in which all agents being “asked” will sequentially perform a set of actions.
* *Collectives*: all agents in this model are behaving individually, regardless of the presence of other agents. There are effectively no collectives in the model.
* *Observation*: the life history of every individual in the model, the size of the population.

1. **Initialization**

The environment of the flask is initialized when the “setup” method is called (or the “setup” button clicked). First of all, the environment is initialized using the set of input data (see 6.) read from “settings.txt” which are stored as global variables.

An output file is then opened to record certain events that would occur during the model’s simulation. Then all patches are initialized to have a white color and concentration set to user-defined value. The shapes of each breed of agents are selected from the library and set as default. A user-defined number of initial adult worms are generated and represented in the view. They each possess an energy reserve randomly selected between the user-defined minimum and maximum initial energy values. Their starting size is the minimal size of adult worms as per user definition. Their age and number of progeny are reset to 0. All worms have random x and y coordinates within the boundaries of the view field. Lastly, the feeding and predation timers are initialized.

1. **Input data**

* *initial-bacteria-concentration:* the starting concentration of food. This variable currently has an arbitrary unit.
* *bacteria-replenish-interval:* the number of ticks in between two consecutive feeding actions. This variable currently has an arbitrary unit.
* *bacteria-replenish-concentration:* the amount of concentration raised globally due to each feeding action. This variable currently has an arbitrary unit.
* *predation-interval:* the number of ticks in between two consecutive predation actions. This variable currently has an arbitrary unit.
* *predation-percentage:* the percentage of *C. elegans* out of the total number present in the system that are removed due to each predation action. This variable’s unit is percent.
* *initial-number-worms:* the number of *C. elegans* at model initialization. The unit of this variable is number worms.
* *worms-gain-from-food:* the amount of energy a *C. elegans* gains from ingesting one concentration unit of food. This variable currently has an arbitrary unit.
* *reproduce-energy-cost:* the amount of energy it requires for an adult *C. elegans* to reproduce. This variable currently has an arbitrary unit.
* *worms-max-progeny:* the maximum number of larvae an adult *C. elegans* is able to produce within each tick. The unit of this variable is number worms.
* *adult-movement-energy-cost:* the amount of energy it requires for an adult *C. elegans* to move per tick. This can be perceived as a maintenance cost that is inevitable to every living *C. elegans*. This variable currently has an arbitrary unit.
* *larvae-movement-energy-cost:* the amount of energy it requires for an larva *C. elegans* to move per tick. This can be perceived as a maintenance cost that is inevitable to every living *C. elegans*. This variable currently has an arbitrary unit.
* *larvae-initial-energy:* the amount of energy a larva possesses when it is born. This variable currently has an arbitrary unit.
* *worm-bag-dormancy:* the number of ticks a *C. elegans* stays in the worm-bag stage before it “explodes” and dies. The unit is number ticks.
* *worm-bag-energy-threshold:* the energy threshold for an adult *C. elegans* to transform into its worm-bag stage, i.e. the adult *C. elegans* transforms into a worm-bag if its energy reserve falls below this value. This variable currently has an arbitrary unit.
* *worm-bag-max-progeny:* the maximum number of larvae a worm-bag is able to produce upon “explosion”.
* *larvae-max-size:* the maximum size a larva can reach. Upon reaching this size, the larva transforms into an adult. This variable currently has an arbitrary unit.
* *adult-max-size:* the maximum size an adult (and any *C. elegans* in this model) is able to reach. Any *C. elegans* that has reached this size will stop its growth.
* *size-energy-multiplier:* the amount of energy that is equivalent to one size of *C. elegans*, i.e. said amount of energy is required for a *C. elegans* to increase its size by one. This variable currently has an arbitrary unit.
* *initial-worm-energy-min:* the lower bound of initial energy the adult *C. elegans* possess upon initialization of model. This variable currently has an arbitrary unit.
* *initial-worm-energy-max:* the upper bound of initial energy the adult *C. elegans* possess upon initialization of model. This variable currently has an arbitrary unit.

All input data listed above are inputted into “settings.txt” in the same listed order, with a space (ASCII = 32) following each variable.

1. **Submodels**

* *Move*: the worm moves in a random direction for 0.5 patch length.
* *Reproduce*: the worm produces a random number (between 0 and a user-defined max number) of larvae, each of which possess a user-defined initial energy reserve (taken from the mother worm’s energy reserve), a generation number increased by 1 compared to its parent, and initialized with its other state variables as well. Upon birth, each larva moves at a random direction for 0.5 patch length so that they do not cluster the display.
* *Catch bacteria*: the worm ingests an amount of bacteria from the patch where it is located. The amount of bacteria it ingests depends on the size of the worm and the concentration of bacteria in the patch. The amount of bacteria ingested times a multiplier dictates the amount of size growth added to the worm.
* *Grow:* when a larva reaches a user-defined critical size threshold, it transforms its stage into an adult. This is a one-way process and no adults can transform back into larvae.
* *Death:* a worm dies under two possible circumstances: 1. Its energy reserve below or equal to zero; 2. A randomly generated number out of 100 falls below the value of the function of dying possibility due to ageing (see below). If either condition is met, the worm is killed.
* *Ageing function:* generates a number from 0 – 100 indicating the percent susceptibility of a particular worm to die due to ageing. The function implemented in this model is function value = 100 \* {1 – exp [-A \* exp (B \* age)]} (Dr. Mitteldorf).
* *Worm-bag:* if a worm’s energy reserve falls below a user-defined threshold, the method kills the adult worm and hatches a worm-bag with the same set of parameters as the adult worm (effectively replacing it).
* *Explode:* if an agent has been in the worm-bag state for longer than a user-defined length of time, the method hatches a randomly generated number of larvae (out of 10), each of which has an energy reserve equal to the worm-bag’s energy reserve divided by the number of larvae produced and a generation number increased by 1 compared to its parent. Each larva then moves 0.5 patch length in a random direction to avoid clustering. Their age and size are initialized as other new-born larvae.
* *Predation:* generates a random number from 0 – 100. If this number falls below the used-inputted predation percentage, the agent performing this action dies (predated).
* *Replenish bacteria:* the user-inputted bacteria replenish concentration is added to the current food concentration of all patches in the environment.

1. **Data Output**

* *Netlogo interface:* the *NetLogo* user interface has been designed to provide a graphic illustration of the behaviors of the agents in the model. On the left side of the interface, a continuous plot shows time as the x-axis and the average concentration of food levels across all patches, the number of adult worms, the number of larvae divided by 10 (in order to maintain readability) and the number of worm-bags as the y-axis. Two data monitors display the overall average food concentration across all patches and the total number of worms in the current system. On the right-hand side, a graphic illustration of the model is shown to provide intuitive and visual representation of the model’s dynamics.
* *File output:* the model generates an output file “log\_\*.txt” (\* is the next integer number that has not been used) in the same directory as the *NetLogo* program. Upon the death of each agent, a line of its life history information is printed to this file in the order of: agent name (number), generation, number of ticks in larva stage, number of ticks in adult stage, number of ticks in worm-bag stage, current energy level, number of progeny produced, and cause of death.

*04/03/2014*