Escherichia coli simulation using the Dynamic Energy Budget theory applied on a multi-agent environment

Sara Magalhães da Cruz

Dissertação para obtenção do Grau de Mestre em Engenharia Informática e Computadores

Júri
Presidente: Professor João Marques da Silva
Orientador: Professor Pedro A. Santos
Co-Orientador: Professor Carlos Martinho
Vogais: Professor Tiago Domingos
        Dr. Gonçalo Marques

Outubro 2011
Agradecimentos

Ao Professor Pedro Santos pela constante disponibilidade e orientação cuidada ao longo de todo o meu percurso no último ano.

Ao Professor Carlos Martinho pela ajuda nas questões técnicas do trabalho e delineamento da sua estrutura e objectivos.

Ao professor Gonçalo Marques pelo esclarecimento imediato de toda e qualquer dúvida.

Aos Professores Tiago Domingos e Bas Kooijman por me terem introduzido e guiado nos meus primeiros passos na teoria DEB.

Aos meus colegas de curso, com quem tive a sorte de partilhar esta experiência.

À Adriana e ao Mário, por me animarem todos os dias e me darem em todas as alturas o seu apoio e compreensão, mesmo em assuntos por vezes para lá do seu interesse.

À Ritinha, uma mente muito para lá da sua idade, pelas longas conversas, conselhos sensatos e apoio incondicional.

À avó Lourdes, a minha alma gémea culinária, pela presença constante e reconfortante em todas as alturas da minha vida.

Aos meus pais, porque tudo o que sou é um reflexo daquilo que me deram, por me ajudarem a encontrar o meu caminho e me prenderem sempre às coisas que são verdadeiramente importantes na vida.

Ao Diogo, por não haver no mundo melhor colega de trabalho ou amigo mais leal. Pela generosidade, bondade, paciência e por ter moldado da melhor forma aquilo que foi a minha experiência na faculdade.

Este projecto foi parcialmente financiado pelo Programa Operacional Regional de Lisboa (QREN SI I&DT) na forma do projecto Life Engine.
Aos meus pais,

Carlos Manuel dos Santos Rodrigues da Cruz e

Custória Irente Vaz Magalhães,

Obrigada por tudo.
Abstract

*Escherichia Coli* is a Gram-negative, rod-shaped bacterium, and one of the oldest microorganisms that maintains a symbiotic relation with Men. It is very commonly used in research and, therefore, it has been long-time established as a reference in microbiology studies. In this work, the results of a multi-agents simulation of E-coli, using the Dynamic Energy Budget theory, are presented, with the goal of reproducing the life cycle behaviors and movement patterns that characterize this bacterium. The work also includes an implementation of a set of mutations based on the alteration of simulation variables and DEB parameters. The ION Framework was used as a development tool in order to generate the bacterial environment and the corresponding interaction protocols with the agents. The aim of this implementation is to test the possibility of using the DEB theory applied in a multi-agent scenario to obtain realistic results comparable with previously reported experimental investigation.
Keywords

*Escherichia Coli*, Multi-agent simulation, ION Framework, Dynamic Energy Budget theory, Bacterial life cycle, Bacterial chemotaxis, Emergent behavior, Mutations, Strains
# Index

1. **INTRODUCTION**  
   1.1. **Motivation**  
   1.2. **Problem**  
   1.3. **Objectives**  
   1.4. **Contribution**  
   1.5. **Document Organization**  

2. **THEORETICAL BACKGROUND**  
   2.1. *Escherichia Coli*  
   2.1.1. **Concluding Remarks**  

3. **STATE OF THE ART:**  
   3.1. **Bacterial Chemotaxis Simulations**  
   3.1.1. **AgentCell:**  
   3.1.2. **The Chemotactic Behavior of Computer-Based Surrogate Bacteria:**  
   3.2. **Bacterial Life Cycle Simulations:**  
   3.2.1. **A Simulation Model of Biofilms with Autonomous Cells:**  
   3.2.2. **BacSim:**  
   3.3. **Summary**  

4. **THEORETICAL MODEL**  
   4.1. **The Dynamic Energy Budget Theory**  
   4.1.1. **Standard DEB Model**  
   4.1.2. **Time-Dependent Differential Equations**  
   4.1.3. **V1-morphs**  
   4.1.4. **Calculating DEB Parameters for *E. Coli* in Life Engine**  

5. **IMPLEMENTATION**  
   5.1. **E. Coli**  
   5.1.1. **ION Framework**  
   5.1.2. **Solution Architecture**  
   5.1.3. **Environment**  
   5.1.4. **Life Cycle**  
   5.1.5. **Dilution**  
   5.1.6. **Chemotaxis**  
   5.1.7. **Strains and Mutations**  
   5.1.8. **Visualization**  
   5.1.9. **Customization**  
   5.2. **Life Engine**  
   5.2.1. **Architecture**  
   5.2.2. **DEBLib**  
   5.2.3. **Assimilation**  
   5.2.4. **Death**  
   5.2.5. **Reproduction**  
   5.2.6. **Shape Correction**
5.3. **INTEGRATION**

6. **RESULTS**

6.1. **EVALUATION STRATEGY**

6.2. **CHEMOTAXIS**

6.2.1. **Motor Rotations**

6.2.2. **Path**

6.2.3. **Emergent Behavior**

6.3. **LIFE-CYCLE**

6.3.1. **Individual Growth**

6.3.2. **Population Growth**

6.3.3. **Starvation**

6.3.4. **Constant Glucose Input**

6.4. **STRAINS AND MUTATIONS**

6.4.1. **Unlimited Growth**

6.4.2. **Individual Structural Growth**

6.4.3. **Starvation**

6.4.4. **Glucose-limited Growth**

7. **CONCLUSIONS**

8. **FUTURE WORK AND POSSIBLE APPLICATIONS**

8.1. **Future Work**

8.2. **Future Applications**

9. **REFERENCES**
# Notation and Symbols

<table>
<thead>
<tr>
<th>Variable</th>
<th>Symbol</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aspect ratio</td>
<td>$\delta$</td>
<td>-</td>
</tr>
<tr>
<td>Scaled reserve density: $[E]/[E_M]$</td>
<td>$e$</td>
<td>-</td>
</tr>
<tr>
<td>Non-allocated energy in reserve</td>
<td>$E$</td>
<td>J</td>
</tr>
<tr>
<td>Accumulated energy investment into maturation</td>
<td>$E_M$</td>
<td>J</td>
</tr>
<tr>
<td>Energy in reserve that is allocated to reproduction</td>
<td>$E_R$</td>
<td>J</td>
</tr>
<tr>
<td>Reserve density: $E/V$</td>
<td>$[E]$</td>
<td>J/µm$^3$</td>
</tr>
<tr>
<td>Volume specific costs of structure</td>
<td>$[E_G]$</td>
<td>J/µm$^3$</td>
</tr>
<tr>
<td>Maximum reserve density: ${p_{Am}}/v$</td>
<td>$[E_m]$</td>
<td>J/µm$^3$</td>
</tr>
<tr>
<td>Scaled functional response</td>
<td>$f$</td>
<td>-</td>
</tr>
<tr>
<td>Energy investment ratio</td>
<td>$g$</td>
<td>-</td>
</tr>
<tr>
<td>Specific death probability (hazard rate)</td>
<td>$h$</td>
<td>s$^{-1}$</td>
</tr>
<tr>
<td>Surface-area-specific max ingestion rate</td>
<td>${J_{XAm}}$</td>
<td>mg/(µm$^2$ s)</td>
</tr>
<tr>
<td>Fraction of mobilized reserve allocated to soma</td>
<td>$\kappa$</td>
<td>-</td>
</tr>
<tr>
<td>Specific-energy conductance: $\dot{v}V_d^{1/3}$</td>
<td>$\dot{k}_E$</td>
<td>s$^{-1}$</td>
</tr>
<tr>
<td>Maturity maintenance rate coefficient</td>
<td>$\dot{k}_J$</td>
<td>s$^{-1}$</td>
</tr>
<tr>
<td>Somatic maintenance rate coefficient</td>
<td>$\dot{k}_M$</td>
<td>s$^{-1}$</td>
</tr>
<tr>
<td>Structural length: $V^{1/3}$</td>
<td>$L$</td>
<td>µm</td>
</tr>
<tr>
<td>Real surface area</td>
<td>$M(V)$</td>
<td>-</td>
</tr>
<tr>
<td>Isomorphic surface area</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Assimilation flux</td>
<td>$\dot{p}_A$</td>
<td>J/s</td>
</tr>
<tr>
<td>Mobilization flux</td>
<td>$\dot{p}_C$</td>
<td>J/s</td>
</tr>
<tr>
<td>Growth flux</td>
<td>$\dot{p}_G$</td>
<td>J/s</td>
</tr>
<tr>
<td>Maturity maintenance flux</td>
<td>$\dot{p}_J$</td>
<td>J/s</td>
</tr>
<tr>
<td>Reproduction flux</td>
<td>$\dot{p}_R$</td>
<td>J/s</td>
</tr>
<tr>
<td>Somatic maintenance flux</td>
<td>$\dot{p}_S$</td>
<td>J/s</td>
</tr>
<tr>
<td>Ingestion flux</td>
<td>$\dot{p}_X$</td>
<td>J/s</td>
</tr>
<tr>
<td>Surface-area-specific maximum assimilation rate</td>
<td>${\dot{p}_{Am}}$</td>
<td>J/(µm$^2$ s)</td>
</tr>
<tr>
<td>Volume-specific maximum assimilation rate</td>
<td>$[\dot{p}_{Am}]$</td>
<td>J/(µm$^3$ s)</td>
</tr>
<tr>
<td>Specific volume-linked somatic maintenance rate: $\dot{p}_S/V$</td>
<td>$[\dot{p}_M]$</td>
<td>J/(µm$^3$ s)</td>
</tr>
<tr>
<td>Specific surface-area-linked somatic maintenance rate</td>
<td>${\dot{p}_T}$</td>
<td>J/(µm$^2$ s)</td>
</tr>
<tr>
<td>Age at division</td>
<td>$\tau$</td>
<td>min</td>
</tr>
<tr>
<td>Energy conductance (velocity): ${\dot{p}_{Am}}/[E_M]$</td>
<td>$\dot{v}$</td>
<td>µm/s</td>
</tr>
<tr>
<td>Structural body volume</td>
<td>$V$</td>
<td>µm$^3$</td>
</tr>
<tr>
<td>Meaning</td>
<td>Notation Rule</td>
<td></td>
</tr>
<tr>
<td>-------------------------------------------------------</td>
<td>---------------</td>
<td></td>
</tr>
<tr>
<td>Variable X at the moment of birth</td>
<td>$x_b$</td>
<td></td>
</tr>
<tr>
<td>Variable X at the moment of cell-division</td>
<td>$x_d$</td>
<td></td>
</tr>
<tr>
<td>Variable X at the moment of puberty</td>
<td>$x_p$</td>
<td></td>
</tr>
<tr>
<td>Physical value of variable X</td>
<td>$x_w$</td>
<td></td>
</tr>
<tr>
<td>Volume-dependent variable</td>
<td>$[x]$</td>
<td></td>
</tr>
<tr>
<td>Surface-area-dependent variable</td>
<td>${x}$</td>
<td></td>
</tr>
<tr>
<td>Time-dependent variable, rate (units per time)</td>
<td>$\dot{x}$</td>
<td></td>
</tr>
</tbody>
</table>
1. Introduction

1.1. Motivation

Systems’ computer modeling is a very important area of research nowadays. By translating knowledge obtained through experimentation so it can be processed by a computer, it is possible to watch a realistic simulation of the world it represents and easily alter its parameters dynamically [43].

This takes particular relevance in areas such as genetics and other biological research fields. By developing a coherent and realistic simulation, it is possible to study a given environment in a much more detailed way than with laboratorial experimentation. It is also possible to freeze the simulation, have it running at any desired speed, capture any information available and go back and forth in time to examine the cause of a given event in a maximum detail level. Furthermore, by applying an individual-based modeling approach to the simulation, it is also possible to study the life of a microorganism at an individual or even at a sub-cellular level. The ION framework, developed by the GAIIPS group in IST, is a tool that expedites the creation of virtual agents and worlds. It can be used for simulations such as the developed work, as well as a variety of other possible applications and, as such, it was tested and applied in this project.

The Dynamic Energy Budget theory [19] is a mathematical theory that aims to reproduce the life cycle of all living beings, from bacteria to plants and animals, at an individual level. Many multi-disciplinary teams around the world are, at the moment, investing work towards modeling tens or even hundreds of organisms using DEB theory, and have been able to do so quite successfully for many examples [7] [8] [12] [31] [33] [41].

The Escherichia Coli bacterium has been long-time established as a reference in microbiology studies, because of the simplicity and organization of its DNA. Hundreds of different E. Coli strains have been studied and sequenced. This takes particular relevance, since many Escherichia Coli strains are pathogenic and can be mortal for humans. Serious outbreaks have occurred throughout history. Most recently, the E. Coli O104:H4 [29] strain outbreak that started in Germany in May of 2011 contaminated more than 3000 people, causing the death of 50 of them. Therefore, it becomes relevant for academic purposes and useful for such public health problems to have an accurate model that allows the prediction of these bacteria’s behaviors.

1.2. Problem

It is simple to see the usefulness of uniting such a wide-range mathematical model as the DEB theory with a multi-agent computer program. It would give scientists the power to quickly model potentially any living organism, and easily run realistic computer simulations that provide data with a detail
level that would be very hard to obtain with a regular laboratorial experiment.

The Life-Engine tool, developed by the Biodroid Company, seeks to take this approach towards creating a biological DEB engine that allows the addition of realistic biological creatures to video-games as well as way to model them for academic purposes. However, it is not yet complete. There are some functionalities that are not yet implemented, and the \textit{E. Coli} parameters were not yet calculated.

The present work represents a step towards the ambitious goal of creating a powerful biological simulation engine that allows the easy creation of DEB agents, their worlds and interactions between them.

\textbf{1.3. Objectives}

The main goal of this work is to develop an \textit{Escherichia Coli} simulation by using a multi-agent approach and applying the Dynamic Energy Budget theory. This implementation should be able to accurately imitate the chemotactic movements displayed by this bacterium as well as its life cycle and interactions with the environment it dwells in. It should also provide a set of mutated \textit{E. Coli} strains as well as an objectively defined process to include new mutations in the existing model. The obtained results should match the ones reported in experimental projects using real \textit{E. Coli} for the movement patterns [5] [22], DEB modeled life cycle [24] [28] and mutations [1] [20] [21].

\textbf{1.4. Contribution}

The work described in the present document combines results obtained in biological fields with a complex mathematical model and state of the art computer programming tools and methods. Therefore, it results in contributions to several fields of research:

\begin{itemize}
\item The resulting biological simulation engine will showcase the applicability of DEB theory to an agent-based computer model, as well as the directness of taking a non-standard DEB organism, such as the \textit{Escherichia Coli} bacterium, and use available literature results to introduce it in the program;
\item Several new sets of DEB parameters will result from the modeling work developed for the various strains of the \textit{E. Coli} bacterium;
\item The ION tool will be tested as to its potential in situations such as the proposed problem for this work;
\item The work will include a revision and integration of new functionalities in the Life Engine tool;
\end{itemize}
• It will also demonstrate how the afore-mentioned biological game engine can be applied towards the academic purpose of scientific investigation;

• Lastly, the project described in this document will result in a biological simulation application useful within itself. This tool will allow the fast extraction of data from bacterial simulations as well as the practical introduction of new strains and mutations, even ones that don’t exist, or haven’t been discovered yet.

1.5. Document Organization

This document is composed of 9 sections. The current one has provided the background motivation for the problem behind the development of this project, as well as the goals and contributions expected upon its completion. It is followed by a theoretical introduction, meant to transmit the theoretical background in which this thesis is based. The third part, the State of the Art, refers other models and simulations, which use the agent-based approach to generate the *E. Coli* bacterium's behaviors. Afterwards, the DEB theory is thoroughly defined, as the theoretical model applied in this work. This is followed by the definition of the structure of the implementation itself. This section includes the solution architecture as well as a description of the employed tools. The next section presents and explains the results obtained from various tests performed on the simulation. The conclusions drawn from the test results as well as from the development of the program are presented in the following section. Finally, the last chapter includes suggestions for possible future increments to the presented solution, both to extend already developed functionalities as well as to introduce new features. This chapter also includes a set of predictions involving future applications of the project. The references used throughout the thesis are gathered in the end of this document.
2. Theoretical Background

2.1. Escherichia Coli

*Without a shadow of a doubt, more is known about the* Escherichia Coli *bacterium than any other bacterial species.* [27]

*E. Coli* is one of the most widely studied bacteria, especially in the field of Genetics. Although the *E. Coli* DNA was not the first one to be sequenced, it is still one of the most well known organisms. This is mainly due to the simplicity of the organization of its DNA. In most bacteria, the information contained in the genetic code is somewhat scattered and unorganized. Oppositely, in the *E. Coli* genetic code, most of the RNA molecules contain information about only one gene. In fact, of all its 2584 predicted transcription units1, 70% code one sole gene. Furthermore, only 6% of its operons code four or more genes [27]. This makes *Escherichia Coli* an easy to use bacterium, and even a vehicle for the production of needed proteins. This is done by inserting the necessary chunk of DNA into the bacterial genetic code and have the *E. Coli* produce the desired protein. One of the most well known cases is the production of human insulin [34].

*E. Coli* bacteria are most well known for their pathogenic effects but most *E. Coli* strains are harmless. In fact, they inhabit all warm-blooded animals and have a regulatory effect in the digestive tube, generating vitamins such as the K vitamin and contributing for the maintenance of the anaerobic environment of the large intestine [27].

The bacterial life cycle is defined into three periods [36]:

B period: The time between birth and initiation of the DNA replication. In this phase, the bacterium merely absorbs and stores substrate.

C period: The time between the initiation of the DNA replication and its end. When this phase is over, the bacterium has its DNA entirely duplicated.

D period: The time between the end of DNA replication and cell division. The *E. Coli* suffers cell-wall constriction, generating two genetically identical bacteria.

![Escherichia Coli Bacteria](image)

**Figure 2.1.1** *Escherichia Coli* Bacteria

---

1 Transcription Unit: The stretch of DNA transcribed into one RNA molecule.
Koppes et al. [21] demonstrated that the switches between the periods were related to the length of the bacterium. This means that it is possible to calculate the duration of these periods by measuring the average time spent between two length thresholds in a population of bacteria. Figure 2.1.2 shows the progression through the cell-cycle periods for the *E. Coli* B/r A strain, and how they are related to the bacterial length.

![Figure 2.1.2 The progression through the cell-cycle periods. (data from Koppes et al. [21])](image)

All this intra-cellular activity is supported through the absorption of substrate such as glucose, fructose and other saccharides [27]. To maximize substrate intake as well as to move away from potentially harmful agents, the *E. Coli* displays a behavior named *chemotaxis* [18], as shown in Figure 2.1.3.

As all flagellated bacterium, *E. Coli* moves through rotations of its flagellar motors. These can be oriented in a clockwise (CW) or counter clockwise (CCW) manner. CCW rotations make the *E. Coli* travel forward, in a movement named *run*. With CW oriented flagella, the bacteria display random rotations, the *tumble* movement. When detecting an attractant, the time spent in the run mode is higher. However, when detecting a repellent, the number of tumble movements is higher, making the bacteria rotate to move away from it. The displacement of the E.Coli in a medium is characterized by long straight runs and occasional short tumbles [22].
Figure 2.1.3 Eschericia Coli navigating towards a capillary tube filled with glucose

The signal chain responsible for this reaction is triggered in the bacterial transmembral receptors and transmitted to the flagellar motors, causing the alternation between the two states [35]. Recent studies demonstrate that these receptors behave as if they possessed memory. This means that after a certain time detecting an attractant, the bacterium responds more strongly to it than in the beginning, when this attractant was first detected. This is thought to be due to the tendency that receptors have to cluster in the cell poles, however, the mechanism which leads to this effect is still obscure [37].

This means that, although the E. Coli does not detect the concentration gradient itself for the attractants in its environment, the aforementioned mechanisms make it possible for this bacterium to move according to it. The E.Coli adjusts the number of tumble movements to the local attractatant/repellent concentration, decreasing them when sensing a higher concentration of attractant. This will make the time spent in the run mode higher, and the effect displayed will be a swim towards where the attractant concentrations are higher. This will make it look as if the bacteria follow the concentration gradient of the attractant in the medium, although they do not explicitly orient towards it. The bacteria will also display a random, erratic behavior when in the presence of a repellent.

2.1.1. Concluding Remarks

The E. Coli agent developed for this project should display behaviors similar to the ones described above.

The bacteria should travel through long runs and short tumbles [22], and at a colony level, they should exhibit a clear displacement towards the attractant concentration gradient. The cells will also possess a simple kind of memory, that allows for the agents to mimic the crescent reaction to an attractant, when it has been detected for a given period of time [37].

The life cycle of the bacteria, modeled using the DEB theory, should also match results obtained through laboratorial experimentation at an individual level [24] as well as at a colony level [28] [42].

---

1 Image from:
http://users.rcn.com/jkimball.ma.ultranet/BiologyPages/T/Taxes.html
3. State of the art:

Early reported simulations of microbiologic life used a population-based model (PLM) to describe the colony's life cycle. Back then, a system's state would be represented as a set of variables and its evolution would be generated by the result of the corresponding differential equations [16].

However, biochemistry has evolved to the point of providing individual information about each bacterium in the colony. By allying this knowledge to the agent-based approach (also referred to as individual-based modeling - IBM) and high processing power of recent computers, it is possible to create simulations at a colony level as the emergent behavior of a high number of individually programmed agents. This is a relatively recent approach, which means that individual-based modeling of microbes, even such a well-studied bacterium as the *E. Coli*, is still in a very early stage.

*E. Coli* simulations tend to turn towards the simulation of one of two generic behaviors:

- Bacterial chemotaxis: the way bacteria travel towards attractants and away from repellents;
- Bacterial life cycle: Cell division, death, metabolism, maintenance, substrate ingestion, etc.

Two bacterial simulations stand out in each of these fields.

3.1. Bacterial Chemotaxis Simulations

3.1.1. AgentCell:

In this model [13], the world, the bacteria and their sub-components are represented as objects. The *world* object contains a set of *cell* (bacteria) objects, which have *motors*. The *motors* contain a given number of *flagella*, which define the *motion* objects, that is, the coordination and movement of the bacterial cell. Only the bacteria are implemented as autonomous agents. Their organelles are simple objects. The environment is defined as a biochemical network with linear gradient, using the *stochsim* package, which provides an interface for the bacteria.

The simulation steps are managed by a *scheduler*, which keeps a global clock and a sorted list of events, taking the necessary actions at constant time intervals.

The *motors* have two states: *clockwise - CW* (to generate the tumble motion) and *counter clockwise - CCW* (to generate the run motion). The switching between the two states occurs when the average concentration of the correspondent signalling chemic in the last 0.3 seconds reaches a pre-determined threshold, as long as the motor has spent a minimum amount of time in that same state.
Figure 3.1.1 Fraction of time spent in the CW and CCW states. a) real cell, b) simulated cell [13]

As Figure 3.1.1 shows, the model successfully recreated the time intervals spent in the run and tumble states registered for experiments with *E. Coli* bacteria.

The individual modeling of each of the cell's organelles is an efficient way to distribute the programmatic complexity of the agent, as well as to observe more realistic results, since it is more accurate than to have a single object to represent the behavior of the whole bacterium. In the current project, a similar approach was followed. The implemented simulation also has a three-dimensional space, like the one applied in this work. However, unlike this reported work which stores an ordered list of events to be processed in each simulation step, in the developed program, by using the ION framework, the simulation has order independence in the requests processing, as well as pseudo-simultaneous events.

### 3.1.2. The Chemotactic Behavior of Computer-Based Surrogate Bacteria:

The model described in [6] tries to recreate bacterial movement by simulating the protein chains that are in the origin of chemotactic reactions. Unlike the previously reviewed work, it is possible to take into account genetic mutations that interfere in the production of those proteins, therefore accounting for genetic heterogeneity in a bacterial colony.

This is achieved by individually modeling the 6 proteins responsible for transporting environment signals from the tansmembral receptors to the flagellar motors. To silence a gene, you simply need to eliminate the protein that it codes and watch the effects in the simulation.

There is one attractant in the environment and the user can define its concentration gradient before the simulation runs, or even in real time. The agents will then process the attractant, typically in 4 steps per second. This model also offers a graphical display so that the moving bacteria can be watched in real time.
Probabilistic models are used to define the switching between the run and tumble states as well as the tumble movement rotations. A random number generator defines the probability of tumbling, according to a given instantaneous concentration of attractant. The tumble rotations themselves are defined randomly, according to experimentally observed parameters [39]. This model was successful at recreating the movement patterns of Escherichia Coli bacteria observed in reported experiments as well as the typical time intervals they spend in each of the two states.

This is a very well thought-out work, and it was very successful in describing the bacterial movements. However, such level of detail results in a small number of simulated individuals due to the complexity of the algorithms. Since, for this thesis, it was important to obtain results at a population level,
this restriction would result in a difficulty to obtain meaningful data with the processing power of an average computer. Nevertheless, the user-defined concentration gradients as well as the visual interface were similarly implemented.

3.2. Bacterial life cycle simulations:

3.2.1. A simulation model of biofilms with autonomous cells:

This is a 2D model [38] that mainly describes a cell as the inside of a membrane, which is made of a set of monomers and the corresponding connections between them.

Cell shape is defined by such parameters as number of monomers in the cell membrane and osmotic pressure on the inside.

Cell state is defined, in each step, by the numbers of constituting monomers, its age (or life time) and its metabolic state (defined by its inner volume).

Growth occurs by adding a monomer to the cell membrane. This happens with a given probability that augments with cell perimeter itself, going down to zero when maximum cell size is reached.

![Figure 3.2.1 Image from [38]](image)

The main events in bacterial life cycle are then defined as:

- **Cell division:** The probability of occurring cell division becomes greater than zero when a certain age and cell perimeter are reached, becoming higher as the simulation goes on since that;
- **Cell death:** When a cell becomes of a certain age, death probability becomes greater than zero, getting higher as the cell gets older. When a minimum threshold of cell perimeter is reached, cell death occurs instantaneously.

The results obtained with this simulation are comparable to the ones reported by other authors working on simulating similar systems.

This cell-wall-based approach will not be used in the current project, since the *E. Coli* is a very simple organism, and it doesn’t change shape that much throughout its life. However, it is worth mentioning to show one more
possible way in which to represent a microorganism’s body, and how it relates to the microorganical life cycle.

3.2.2. BacSim:

This model [23] seeks to represent bacterial behaviors such as substrate ingestion, metabolism, maintenance and death. Each cell is conceptually represented as a sphere and graphically drawn as a circle in a 2D grid. The spaced occupied by each cell becomes relevant in this model, as it is used to limit the number of cells in the simulation. Bacteria cannot overlap in this simulation’s bidimensional space, therefore, cell division itself is limited by each cell’s surrounding space and the maximum number of individuals is limited by the area defined for the grid.

Bacsim asks for the definition of 8 parameters that can be obtained experimentally, such as substrate uptake, anabolism, catabolism, cell volume at division and surface area. To get individual values for each agent, they used reference E. Coli values for these parameters and varied them randomly according to a Gaussian curve in order to account for the heterogeneity factor in the bacterial population.

Substrate is discreetly defined for each point of the grid. This makes it mandatory for the simulation to run at a minimum of 10,000 steps per simulation minute, to guaranty that between two steps a bacterium will not ingest more substrate units than the ones at its reach. This limitation can make it hard or even impossible to apply this model to the simulation of large-scale colonies.

As substrate is absorbed, it is converted into biomass and waste. Biomass itself also generates waste as maintenance operations occur. Maintenance needs are directly proportional to cell volume. When there’s shortage of substrate in the environment, cell size can decay, as there isn’t enough substrate to account for the maintenance operations. When a bacterium reaches a minimum threshold of biomass, cell death occurs. Its present biomass is then converted into substrate so that the surrounding individuals get to augment or, at least, maintain their volume.

The test results reported for this model match the experimental results obtained with real E. Coli colonies, making this a suitable model for the simulation of the reactions it tries to reproduce.

For the current project, a similar approach was followed to define the life cycle of the E. Coli. Experimental parameters were gathered and applied in a mathematical model, to generate the life cycle events that define the bacterial cell life. However, in the current project, the DEB theory was used, since it is a well-established model for the definition of the life-cycle of virtually any living-being.
3.3. Summary

Having in mind the previously reviewed works, the goal in the present project was to capture the most relevant features in each of them, having in mind the goal behavior of the simulated bacteria, as well as the restrictions of the world in which they were incorporated. Some new features were also added, in order to accommodate the specific needs of the current project. Table 1 shows a resume of the features of each model that are going to be incorporated, as well as the newly developed features.

<table>
<thead>
<tr>
<th>Project</th>
<th>Target Feature</th>
</tr>
</thead>
<tbody>
<tr>
<td>[13]</td>
<td>Create individual objects to define each cell organelle</td>
</tr>
<tr>
<td>[13]</td>
<td>Have a three-dimensional simulation space</td>
</tr>
<tr>
<td>[6]</td>
<td>Implement graphical interface</td>
</tr>
<tr>
<td>[6]</td>
<td>Allow user defined parameters for the concentration gradient of the attractants/repellents</td>
</tr>
<tr>
<td>[13], [6]</td>
<td>Display realistic chemotaxis movement</td>
</tr>
<tr>
<td>[23]</td>
<td>Implement accurate life cycle model</td>
</tr>
<tr>
<td>-</td>
<td>Use a well-studied, widely-tested, universal mathematical model to describe the bacterial life cycle (DEB [19])</td>
</tr>
<tr>
<td>-</td>
<td>Have order independence in the simulation’s request processing (ION Framework [40])</td>
</tr>
<tr>
<td>-</td>
<td>Introduce DEB modeled mutations</td>
</tr>
</tbody>
</table>

Table 1 Target features of each project
4. Theoretical Model

4.1. The Dynamic Energy Budget theory

To define the life-cycle variables, states and flows in a computer simulation, it is necessary to find a way to relate the information about all the entities present in environment that affect the behavior of the target organism. This can be done by analyzing experimental results and empirically defining the evolution of the simulation world, or by applying an already defined mathematical model.

The Dynamic Energy Budget (DEB) [19] theory aims to describe the life cycle of all living organisms, whether they are uni or multi-cellular, as the energy fluxes between them and the environment, as well as the internal energy fluxes between their sub-components.

The organism’s development state is defined by 3 main state variables:

*Reserve* ($E$) refers to the energy absorbed from the substrate present in the environment and stored to be used for metabolic purposes. The energy contained in this variable will be redirected to the other components in order to fulfill the organism’s needs.

*Structure* ($V$) is related to the physical size of the organism. Any increase in structure (growth) will later require energy to be mobilized from the reserve towards its maintenance.

*Maturity* ($E_{n}$) relates to the complexity (development level) of the organism. When a being is created, its maturity ($E_{n}$) has value 0 and it is designated as an *Embryo*. Birth occurs when maturity reaches the birth threshold ($E_{n}^{b}$), and is defined as the point where the organism starts feeding (assimilation). It becomes a *Juvenile*. When the puberty threshold is reached ($E_{n}^{p}$), the individual gets the ability to generate offspring, becoming an *Adult*. Until this moment, the energy invested towards maturity increase was devoted to maturation, to augment the individual’s complexity. DEB theory considers that when an individual becomes an adult, it has reached the maximum level of complexity, making it unnecessary to invest more energy towards maturation.

From puberty on, reserve energy will be applied to reproduction, so that an organism can generate offspring. The energy that was formerly mobilized to maturity will be “stored” in the *Reproduction Buffer* ($E_{r}$). Each time the mother lays an egg, $E_{r}$ will decrease. It is as if the mother releases energy into the world, in the form of an egg. A new egg is only made of reserve, it has no structure. Until birth, the energy will be converted to structure and the embryo will not feed, living of the reserve energy given by the mother. When an organism reaches the maturity level at birth, it becomes a juvenile. At the moment of birth, the reserve density [$E$] ($E/V$ – the relation between reserve
and structure) is the same as the mother’s when it laid the egg. This is called the *Maternal Effect.*

### 4.1.1. Standard DEB model

![Figure 4.1.1 DEB energy fluxes](#)

Figure 4.1.1 shows how energy flows through the DEB components of an organism, in the form of its various fluxes.

\( \dot{p}_x \) is the ingestion flux. It contains the energy that enters the organism from the environment as the result of ingestion and has the chemical composition of food.

After assimilation, the energy will be stored in the Reserve (E). This is flux \( \dot{p}_A \), the assimilation flux, which results from ingestion but already has the chemical composition of reserve.

From the reserve, energy will be mobilized to the other variables to fulfill the individual’s biological needs. This is defined as the mobilization flux \( \dot{p}_C \). One part of the reserve energy is applied to soma, and the other to maturity. The proportion in which reserve energy is applied to these two components is defined by \( \kappa \), a constant that takes values between 0 and 1. Therefore, a fraction \( \kappa \) of \( \dot{p}_C \) will be used for growth \( \dot{p}_G \) and somatic maintenance \( \dot{p}_S \), and...
while the rest of it goes to maturity increase, reproduction \( \dot{p}_r \) and maturity maintenance \( \dot{p}_j \). A higher maturity value will mean that the individual has greater complexity, and a higher structure value will mean an organism is bigger. DEB theory considers that one part of the somatic maintenance costs is proportional to volume and the other part is proportional to the surface area, and maturity maintenance is proportional to the organism’s complexity. Therefore, growth and maturation will imply greater maintenance costs from that moment on.

Sometimes, there isn’t enough food in the medium to support the organism’s activities. When there’s substrate shortage, growth continues for a short period of time, using the reserve. Afterwards, if there isn’t enough substrate available for somatic maintenance, the structure will diminish, causing the individual to “shrink”. When structure reaches a minimum threshold value, death occurs. If there isn’t enough available substrate to assure maturity maintenance, this will cause the value of this variable to decrease, “rejuvenating” the organism.

### 4.1.2. Time-dependent differential equations

Biological simulations are usually applied to reproduce the behavior of a given environment during a given time interval. The standard way to apply a mathematical model, such as DEB, to a computer simulation is to define a clear set of system variables, input and output parameters and how they evolve though time.

We have our three main state variables: Reserve, structure and maturity, with which we can define an individual's state and recreate the main events in its life cycle. These values can be differentially defined as functions of time, by using other time-dependent variables: the DEB energy fluxes, in the following way:

- **Reserve**: \( \frac{dE}{dt} = \dot{p}_A - \dot{p}_C \)

- **Structure**: \( \frac{dV}{dt} = \frac{\dot{p}_G}{E_G} \)

- **Maturity**: \( \frac{dE_H}{dt} = \dot{p}_R, \quad E_H < E_H^p \)

- **Reproduction Buffer**: \( \frac{dE_H}{dt} = \dot{p}_R, \quad E_H > E_H^p \)

At this point, all that there is left is to have the set of equations needed to create a DEB computer model is to define these fluxes as functions of parameters that can be obtained experimentally or by browsing the literature:
\[ \dot{p}_A = V^{2/3} \{ \dot{p}_{Am} \} f \]

\[ \dot{p}_C = E \frac{[E_G] + \dot{v}/L + \dot{p}_S/V}{\kappa E/V + [E_G]} \]

\[ \dot{p}_S = V[\dot{p}_M] + V^{2/3} \{ \dot{p}_T \} \]

\[ \dot{p}_G = \kappa \dot{p}_C - \dot{p}_S \]

\[ \dot{p}_J = \dot{k}_J E_H \]

\[ \dot{p}_R = (1 - \kappa) \dot{p}_C - \dot{p}_J, \text{ where,} \]

- \( \{ \dot{p}_{Am} \} \), in \( \text{J}/(\mu \text{m}^2 \ \text{s}) \) is the surface-area-dependent maximum assimilation rate, i.e., the maximum amount of energy a being with a given surface-area can assimilate in a given amount of time;

- \( f \) is the functional response. A being is not always able to assimilate according to its maximum assimilation rate, mainly because the amount of food present in the environment may not allow it. The functional response is multiplied by the maximum assimilation rate to specify how much of that energy can be assimilated according to environmental or other limitations;

- \([E_G] \), in \( \text{J}/\mu \text{m}^3 \) is the volume-specific costs of structure, which defines how much energy must be mobilized from the reserve in order to add a given amount of volume to the structure value;

- \( \dot{v}, \) in \( \mu \text{m}/\text{s} \) is called energy conductance and specifies how fast energy flows from reserve to the other variables;

- \( L \) (\( \mu \text{m} \)) is the structural length (\( V^{1/3} \));

- \( \{ \dot{p}_M \} \), in \( \text{J}/(\mu \text{m}^3 \text{s}) \) is the specific volume-linked somatic maintenance rate, which expresses the amount of energy that must be allocated from reserve in order to maintain a given amount of structure for a given time interval;

- \( \{ \dot{p}_T \}, \) in \( \text{J}/(\mu \text{m}^2 \ \text{s}) \) is the specific surface-area-linked somatic maintenance rate, which has the same meaning of \( \{ \dot{p}_M \} \), but refers to maintenance costs which relate to the surface-area of an individual, such as maintaining the body-temperature;

- \( \dot{k}_J (\text{S}^{-1}) \) is the maturity maintenance rate coefficient, which specifies how much of the energy that is applied towards maintenance is devoted specifically to maturity maintenance.

4.1.3. V1-morphs
For simple beings, such as prokaryotic cells, the cell-cycle stages can be directly related to the structure value. The *E. Coli* bacterium, for instance, reproduces through division when it has duplicated its entire DNA. The organism’s complexity doesn’t change throughout its life, and all the cell cycle events (birth, reproduction and death) result from changes in structure alone. Cell division, for example, occurs when the structure threshold at division is reached \( (V_d) \), producing two daughter cells with half the structure of the mother. To represent these kinds of beings using the DEB model, it is usually considered that they are always juveniles and that the constant \( \kappa \) is equal to 1, meaning that there is no energy invested into maturation or the reproduction buffer. This allows a simplification of the conceptual model, displayed in Figure 4.1.2.

![Figure 4.1.2 Energy fluxes in simplified DEB model for unicellular organisms](image)

Another concern one must have when modeling using the DEB theory is to take into account the shape of the organism. The standard DEB equations assume that the organism is an isomorph, which means that it grows proportionally in all three dimensions. Individuals with this kind of shape are considered \( V^{2/3} \)-morphs, since their volume is proportional to their surface area to the power \( 2/3 \). This relation is very important since it affects the substrate uptake process and maintenance. However, some beings such as certain bacteria grow in length alone. Such organisms are called \( V^1 \)-morphs, because their body volume is directly proportional to one dimension: length. There is one other shape possible: \( V^0 \)-morphs, that maintain their surface area throughout their

![Figure 4.1.3 E. Coli shape. [Image from: http://drolph.blogspot.com/2011/05/bacteria.html]](image)
entire life, which does not mean that there is no structure-wise growth. Structure increases throughout their lives, just as with other organisms, but it gets constricted, maintaining the same surface area from birth until death. Eschericia Coli is a rod. Its body grows in length alone, but it maintains two constant size caps in the extremities (Figure 4.1.3). As such, it is considered a mixture between a V0 and V1 morph.

To deal with these possibilities, DEB theory defines the shape correction function $\mathcal{M}(V)$, which must be inserted in the equations that involve energy conductance $\dot{\nu}$ and the maximum assimilation rate $\{\hat{p}_{Am}\}$. For beings that are not isomorphs, $\dot{\nu}$ should be replaced by $\dot{\nu} \ast \mathcal{M}(V)$, and the same method should be applied to $\{\hat{p}_{Am}\}$. The shape correction function for rods is:

$$\mathcal{M}(V) = \frac{\delta}{\delta + 2} \left( \frac{V}{V_d} \right)^{-2/3} + \frac{2}{\delta + 2} \left( \frac{V}{V_d} \right)^{1/3}$$

The $\delta$ constant is called the shape coefficient and it determines the contributions of the V0 and V1-morph components. When $\delta \to \infty$, the being becomes close to a V0-morph. The *E. coli* has a shape coefficient of 0.26.

### 4.1.4. Calculating DEB parameters for *E. coli* in Life Engine

In Life-engine, the differential DEB equations are defined as function of the organism’s fluxes and certain other constants, as described before. To have the *E. coli* run using it, first it was important to find biological data and use it to generate the set of input variables requested. Initially, a set of parameters was gathered from literature. However, it was necessary to convert them to a group of numerical parameters usable as input in Life Engine’s DEB library. The current section displays the initial source DEB values and demonstrates how they were manipulated using DEB equations to generate the target input parameters to use on the DEB Lib.

Units used (adjusted to the scale of the simulation):
Length: µm, Volume: µm³, Energy: J; time: s

Source Parameters (From DEB book [19], chapter 9, page 358):
- $e_b = 1$
- $\dot{k}_g = 0.67$ h⁻¹
- $g = 0.86$
- $\dot{k}_M = 0.008$ h⁻¹
- $\dot{J}_{Am} = 0.65$ mg / (mm³h)

Data from *E. coli* literature for strain Br/A (mean values) [9]:
- Physical length at birth: $L_{b,w} = 1$ µm
- Physical length at division: $L_{d,w} = 2$ µm
- Physical volume at birth: $V_{b,w} = 0.6$ µm³
- Physical volume at division: $V_{d,w} = 1.2$ µm³
Target parameters (To run with Biodroid’s DEB library):
Allocation fraction to soma $\kappa = 1$
Surface-related somatic maintenance cost $\{\hat{\rho}_f\} = 0$
Maximum reserve density $[E_M]$
structureCost $[E_G]$
Volume-related somatic maintenance cost $[\hat{\rho}_M]$
Structure value at the moment of division $V_d$
Energy conductance $\psi$
Maximum assimilation rate (Relative to surface) $\{\hat{\rho}_{Am}\}$

The variables tagged with 0 were defined as the standard values for prokaryotes.

1. Calculating $[\hat{\rho}_{Am}]$

The maximum assimilation rate (relative to volume) is

$[J_{Am}] = 0.65 \text{ mg} / \text{(mm}^3\text{ h)} = 1.81 \times 10^{-16} \text{ mg} / \text{(um}^3\text{ s)}$

This is the amount of glucose an E. Coli with a certain volume consumes in a given time interval.

One glucose molecule yields 36 ATP. One mol of ATP generates 30,5 Kj.

Since glucose has a molar mass of 180,16 g/mol, we can conclude that glucose yields $6095 \text{ mg} / \text{mg}$.

Having the maximum amount of glucose an E. Coli can consume, and knowing how much energy a mg of glucose yields, it becomes simple to calculate the maximum amount of energy an E. Coli can assimilate in a given time interval, for a certain body volume:

$[\hat{\rho}_{Am}] = 1.81 \times 10^{-16} \times 6095 = 1.1 \times 10^{-12} \text{ / (um}^3\text{ s)}$

2. Calculating $[E_M]$

From the DEB definition of $\hat{k}_E$, we have:

$\hat{k}_E = \hat{\psi} \times V_d^{-\frac{1}{3}} \iff \hat{\psi} = \hat{k}_E \times V_d^{\frac{1}{3}}$

From which we can obtain the expression that gives us the energy conductance of the E. Coli $\hat{\psi}$.

It is also possible to convert from volume-related maximum assimilation rate to surface-related maximum assimilation rate:

$[\hat{\rho}_{Am}] \times V_d^{-\frac{1}{3}} \iff [\hat{\rho}_{Am}] = [\hat{\rho}_{Am}] \times V_d^{\frac{1}{3}}$

From there, we can obtain $[E_M]$ using the standard DEB expression that defines it:

$\frac{[E_M]}{\hat{\psi}} = \frac{[\hat{\rho}_{Am}] \times V_d^{\frac{1}{3}}}{\hat{k}_E \times V_d^{\frac{1}{3}}} \iff [E_M] = \frac{[\hat{\rho}_{Am}]}{\hat{k}_E}$
3. Calculating \([E_c]\)

\([E_c]\) can be directly obtained if we have the numerical value of the energy investment ratio \((g)\), in the following way:

\[
g = \frac{[E_G]}{\kappa * [E_M]} \Leftrightarrow [E_G] = g * \kappa * [E_M]
\]

4. Calculating \([\dot{p}_M]\)

Having calculated \([E_G]\) in the previous step, and knowing the numerical value of the somatic maintenance rate coefficient \((\dot{k}_M)\), the volume-linked somatic maintenance rate can be obtained using the standard DEB model expression:

\[
\dot{k}_M = \frac{[\dot{p}_M]}{[E_G]} \Leftrightarrow [\dot{p}_M] = \dot{k}_M * [E_G]
\]

5. Calculating \(V_d\)

It is known that for most bacteria, which reproduce through division, we have: \(V_d = 2V_b\).

Therefore, we can calculate the structure value at division \((V_d)\) if we have the structure value at birth \((V_b)\).

Knowing the physical volume at birth, it can be assumed that one part of that volume is occupied by reserve, while the rest is occupied by structure.

The reserve density \([E] = \frac{E}{V}\) gives us the amount of reserve for a given structure value. Knowing the scaled reserve density at birth \(e_b = \frac{[E_b]}{[E_m]} = 1\)

we can come to the conclusion that \(\frac{E_b}{V_b} = [E_m]\).

At this point we have a relation between reserve and structure at the moment of birth. However, \([E_m]\) has dimension energy per volume, and we are interested in a relation between volumes, corresponding to the reserve and structure, so that they can be related to the physical volume at birth.

It is possible to convert an amount of energy into a body volume if we have a factor that correlates them. Let’s call it \(F_{EV}\). It can be calculated from the reserve density \((d_e, \text{g/cm}^3)\), the chemical potential of the reserve \((\mu_E, \text{J/mol})\) and the molar weight of reserve \((w_E, \text{g/mol})\) as follows:

\[
F_{EV} = d_e * \frac{\mu_E}{w_E} (\text{J/\mu m}^3)
\]

The expression \(E_b = [E_M] * V_b\) gives us the reserve value at birth as an energy value. The volume of reserve at birth will then be \(E_{r,b} = \frac{[E_M] * V_b}{F_{EV}}\)
By defining the physical volume of the organism as the sum of the reserve volume and the structure volume $V_{w,b} = E_{v,b} + V_b$, we get

$$ V_b = V_{w,b} - E_{v,b} \iff V_b = V_{w,b} - \frac{[E_m]*V_b}{F_{EV}} \iff V_b = \frac{F_{EV} * V_{w,b}}{F_{EV} + [E_m]} $$

$$ V_d = 2 * \frac{F_{EV} * V_{w,b}}{F_{EV} + [E_m]} $$

6. Calculating $\dot{v}$

Now that we have calculated the structure value at division, it is possible to obtain the energy conductance value:

$$ \dot{k}_E = \dot{v} * V_d^{1/3} \iff \dot{v} = \dot{k}_E * V_d^{1/3} $$

7. Calculating $\{\dot{p}_{Am}\}$

The surface-specific maximum assimilation rate can also be computed now, resorting to the structure value at division:

$$ [\dot{p}_{Am}] = \{\dot{p}_{Am}\} * V_d^{-1/3} \iff \{\dot{p}_{Am}\} = [\dot{p}_{Am}] * V_d^{1/3} $$

At this point, all the expressions that were needed to generate the necessary parameters for life engine from literature values have been derived from the DEB equations.
Table 2 displays the set of source parameters and the numerical conversions applied in order to adjust them to the scale of the simulation.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Units</th>
<th>Value</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>$J_{X_{Am}}$</td>
<td>0.65</td>
<td>mg/(mm$^3$ h)</td>
<td>1.81E-16</td>
<td>g/(μm$^3$ s)</td>
</tr>
<tr>
<td>e</td>
<td>1</td>
<td>--</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\dot{k}_E$</td>
<td>0.67</td>
<td>h$^{-1}$</td>
<td>1.86E-04</td>
<td>s$^{-1}$</td>
</tr>
<tr>
<td>$\kappa$</td>
<td>1</td>
<td>--</td>
<td></td>
<td></td>
</tr>
<tr>
<td>g</td>
<td>0.86</td>
<td>--</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\dot{k}_M$</td>
<td>0.008</td>
<td>h$^{-1}$</td>
<td>2.22E-06</td>
<td>s$^{-1}$</td>
</tr>
<tr>
<td>$d_E$</td>
<td>0.1</td>
<td>g/cm$^3$</td>
<td>1E-13</td>
<td>g/μm$^3$</td>
</tr>
<tr>
<td>$\mu_E$</td>
<td>550000</td>
<td>J/mol</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$w_E$</td>
<td>27.8</td>
<td>g/C-mol</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$V_{w,b}$</td>
<td>0.6</td>
<td>μm$^3$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 2 Initial values and conversions

Table 3 resumes the set of DEB parameters used as input for the *E. Coli* bacterium in the DEB library.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>$[\dot{p}_{Am}]$</td>
<td>1.10E-12</td>
<td>J/(μm$^3$ s)</td>
</tr>
<tr>
<td>${\dot{p}_{Am}}$</td>
<td>7.37E-13</td>
<td>J/(μm$^2$ s)</td>
</tr>
<tr>
<td>$[E_M]$</td>
<td>5.91E-09</td>
<td>J/μm$^3$</td>
</tr>
<tr>
<td>$[E_G]$</td>
<td>5.08E-09</td>
<td>J/μm$^3$</td>
</tr>
<tr>
<td>$[\dot{p}_M]$</td>
<td>1.13E-14</td>
<td>J/(μm$^3$ s)</td>
</tr>
<tr>
<td>$\dot{v}$</td>
<td>1.25E-04</td>
<td>J/(μm$^3$ s)</td>
</tr>
<tr>
<td>$V_b$</td>
<td>0.15</td>
<td>μm$^3$</td>
</tr>
<tr>
<td>$V_d$</td>
<td>0.30</td>
<td>μm$^3$</td>
</tr>
</tbody>
</table>

Table 3 Output values generated
5. Implementation

The current chapter specifies the details of the implementation of the project, both the simulation engine developed for the *E. Coli* and the functionalities added to Biodroid's DEB Library to adjust it to the specific requisites of a bacterial simulation.

5.1. *E. Coli*

This section includes a description of the ION Framework, used to develop the *E. Coli* agents, as well as a detailed explanation of all the procedures followed in order to implement the simulation engine.

5.1.1. ION Framework

An agent-based simulation consists of a controlled interaction between autonomous agents and a virtual environment. Therefore, the simulation of the environment itself and its effects on the agents and vice-versa is a crucial part of this type of modeling.

The ION Framework [40], “A Framework for the Simulation of Worlds and Agents” offers a platform for the creation of virtual 3D worlds as well as mechanisms for integration with the agents’ perception-action systems. One key feature of this tool, that takes special relevance for the developed work, is that it offers order independence in the request processing. This means that, unlike most simulations where events are always handled in a sequential manner, ION is able to generate “simultaneous” events.

The main object used in ION is the *element*. The simulation state is represented by the set of all *elements*, which can be agents, physical objects, positions, *requests*, *events*, etc.

The *request* and *event elements* have a special function, as they are created to generate changes in the world. Each *element* has its own set of *requests*, *request handlers*, *events* and *event handlers*. This way, when an agent wants to perform an alteration to the world’s state, it generates an object of *request* type. Using the observer design pattern, the ION framework notifies every element that has registered a handler for that request, in the following update step. The *requests* are processed within the respective *elements* and then, if the changes registered are of interest to other classes, an event is raised. Afterwards, in the Process Events phase, every *element* which has registered a handler for a generated *event* is notified. If *events* are generated within *event handlers*, they are processed in the same step, if not, the simulation continues until the next update step is reached (see Figure 5.1.1).
This update flow makes it possible for developers to generate “simultaneous” events, since two requests that were placed at the same time and access the same state variables will generate a single event that combines the effects of both of them.

Figure 5.1.2 shows an example of how this technique could be used to generate “simultaneous” actions.

When concurrent requests collide, ION provides a set of tools for the developer to implement the necessary conflict resolution mechanisms. This is done by centralizing the conflicting requests in a single request handler, where the developer can create the mechanisms for conflict resolution according to the logic of the simulation. This way, ION allows total freedom on how to address these kinds of problems.
5.1.2. Solution Architecture

Figure 5.1.3 Solution architecture for the final E. Coli simulation version
The efficiency of the simulation can be compromised if the space occupied by the ION elements is greater than the RAM memory available. Because of this, the number of objects that implement the element class was kept at the minimum value possible.

The final simulation architecture, displayed in Figure 5.1.3, defines three main classes, which are ION elements:

**Program:** Controls the flow of the simulation. It calls the ION routines that trigger the elements’ initialization and updates, and provides the interface between the simulation logic and visualization components.

**Grid:** The class responsible for maintaining the state of the simulation. It holds reference to all the objects that constitute the environment as well as a 3D matrix that represents substrate availability in each cell of the simulation space.

**EColi:** Defines an *Escherichia Coli* agent.

The solution code was entirely developed in C#.

### 5.1.3. Environment

The simulation space is defined as a cube or parallelepiped with dimensions *X*, *Y*, *Z* μm. The environment was developed to incorporate three-dimensional periodic boundary conditions in Cartesian coordinates. This means that, when an individual trespasses the space boundary, it is placed in the opposite side of the simulation world. If we have a cubic world defined by coordinates that can take values from 0 to max, if the value of a coordinate drops below zero, such as *y* = −*a*, the agent will be placed in *y* = max − *a*. In the same way, if the value of a coordinate mounts to, for example, *y* = max + *b*, it will be placed in *y* = *b*.

The information about each cell is stored in the form of a 3D integer matrix. It was crucial for this work to store the minimum amount of information in each matrix entry, otherwise it would be impossible to simulate a world bigger that a few cubic microns with an average computer RAM. Since the world is discretized in cubic cells with side equal to 1 μm, to achieve this, the concentration matrix usually has volume equal to the world volume / 1000. This means that the grid is defined by cubic bins with side 10 μm, which contain a given amount of substrate. Furthermore, the amount of substrate is defined as an unsigned short integer. “Short” to save memory space, since it has half the size of regular integer (16 instead of 32 bits in C#), and “unsigned” because there can be no negative concentrations (the minimum amount of substrate is zero), and this allows the entire 16 bits to be applied to storing useful information, instead of keeping one bit to represent the number signal.
By applying these simple strategies it is possible to simulate a world with a volume of 1 ml in 2 Gb:

\[ 1 \text{ ml} = 1 \text{ cm}^3 = 1 \text{ cm} \times 1 \text{ cm} \times 1 \text{ cm} = 10000 \ \mu\text{m} \times 10000 \ \mu\text{m} \times 10000 \ \mu\text{m} \]

By using bins of \( 10 \times 10 \times 10 \ \mu\text{m} \), with each entry occupying 2 bytes (16 bits short integer), we get:

\[ \text{Memory Needed} = 1000 \times 1000 \times 1000 \times 2 \ \text{bytes} = 2 \ \text{Gbytes} \]

It is possible to occupy even less space by increasing the bin size, although it is important not to use a value that is much larger than the distance an \textit{E. Coli} travels in each simulation step, otherwise they won’t be able to accurately detect the concentration gradient and move according to it.

5.1.4. Life Cycle

The \textit{E. Coli} agent’s life cycle was defined by applying the DEB theory to this organism. To achieve this, the DEBlib part of the Life Engine (see Figure 5.2.1) was used as function library that interacts with the \textit{E. Coli} DEB component.

Each agent has a DEB object. This class provides an interface to the DEBLib and stores all the information returned from it that is relevant to the simulation. In each simulation step, the DEB variables are updated according to the substrate available in the environment. Afterwards, if any significant changes have occurred, the necessary requests and events are generated.

At the beginning of each simulation step (see Figure 5.1.4), the \textit{E. Coli} removes a part of the substrate available in the bin it is in. Afterwards, the DEB variables are updated by the DEBLib, to reproduce the effects of the assimilation of that given amount of substrate.

At this point, the DEBLib is pooled to ensure that the bacterium is still alive. If that is not the case, it will be removed from the simulation. To keep the simulation’s realism, it was important to create a mechanism to incorporate the necrofagy behavior [14]. The most accurate way to do this would be to keep the body of the dead bacterium in the simulation, and if another agent passed nearby it would be assimilated. The problem with this approach is that, to achieve this, in each step, every bacterium would have to see if one of all the other agents was dead and at its reach and, if that was the case, eat it.
To avoid the performance problems that would obviously arise from this method, at the moment of death the *E. Coli* structure is converted to substrate and added to the amount present in the Grid bin the agent died in. This way, the effect of including this behavior in the general performance of the simulation is irrelevant. A simple glucose spreading algorithm was also included, to reproduce the diffusion effect that happens when a soluble material, such as glucose, is placed in an aqueous medium. Furthermore, the distribution of glucose in the environment as a result of the death of an *E. Coli* allows the application of the
chemotaxis mechanisms to orient the other bacteria towards it. At the location of death, a higher concentration of glucose is placed. Then, 3 other glucose layers are introduced around it to create a concentration gradient that leads to the place of death. The sum of the glucose placed in each bin is equal to the total mass of substrate generated from the structure of the dead bacterium. This effect happens instantly after death, and only once. This means that the glucose will not suffer any more diffusion after this first moment.

If the individual is still alive, the visual layer of the program is notified, so that the size of the bacterium on screen will match its volume according to the DEB parameters, making the visual object grow or shrink.

Afterwards, the DEBLib will be pooled once more, to see if the individual is able to divide. If that is the case, a new bacterium will be added to the simulation, and both mother and child will have a size equal to half the size of the mother at the moment of division.

If no life-cycle events are triggered, the bacterium will move through the environment.

The last step will be to apply the hazard rate corresponding to the dilution factor of the laboratorial experiment that is being recreated.

5.1.5. Dilution

Many reported tests performed on the *E. Coli* bacteria take place in liquid media with suspended microorganisms. Frequently, to extract data from such experiments, a fraction of the fluid is removed continuously or at fixed time intervals. This changes the dynamic of the environment, and it would be inaccurate to carry out a simulation meant to reproduce those results without accounting for this dilution factor, or throughput rate.

DEB theory defines a constant named hazard rate (*h*), a percentage value that indicates the probability of an individual dying. It is usually applied to describe the effects of aging in cells or organisms. By increasing this value throughout the life of an individual, it is possible to simulate the increase in an organism's fragility as its cells get older, and model death by old age.

In the current scenario, it can also be set to a constant value to define the probability of a cell being removed because of the dilution factor.

For instance, in the experiment carried out by Dent et al.[10], we have a dilution factor of 0.064 h⁻¹, meaning that the probability of an individual being removed from the simulation is of 0.064%. To get the hazard rate, we just have to relate this value with the simulation step:

\[
h = \frac{0.064}{3600} \times \frac{\text{simulation \_ step}}{1000} \quad \text{(for a simulation step defined in milliseconds)}
\]
So, if the update function is called every 70 milliseconds, we have a hazard rate of 0.00012 %. In each simulation step, for each individual bacterium, a random number will be drawn, and the E. Coli will be removed from the simulation if the random number is smaller than the defined dilution rate.

It is also necessary to apply the dilution factor to the substrate. To get this effect, in each simulation step, a fixed portion (corresponding to the hazard rate) of the substrate present in each bin is subtracted from it.

5.1.6. Chemotaxis

In order to accurately reproduce the chemotactic behavior displayed by the E. Coli bacterium, the architecture of the system was developed to reflect the relation between the organelles responsible for this reaction in the bacterium itself.

Each bacterium has a set of about 10 sensors and an equal number of flagella. This is the average number of flagella for a B/rA E. Coli [9]. The sensors get information about the substrate present in the two poles of the E. Coli, to reproduce the naturally occurring receptor clusters. The difference between the substrate available at the moment and the substrate available in the previous step is transmitted to the corresponding flagellum, and then converted to a probability of tumbling.

In all the tests performed, the Grid bins had a size of 10 μm. Since an average size E. Coli has a length of 1 to 2 μm, at most the sensors would be able to get information from two different bins, if one half of the E. Coli was on the first bin and the tail on the second. However, this is not an implementation restriction. For the target results of the project, it was important to have a world with dimensions as broad as possible. This was done by using Grid bins with 1000 μm³. However, the bins can be as small as needed, for applications with different requisites, such as a detailed study of the chemotactic reactions of individual sensors and flagella.

In Figure 5.1.6 the bin size is 0.2 X 0.2 X 0.2 μm and the bacteria have 12 sensors. Both those parameters are easily adjustable and can be adapted to each situation.
There are four different tumble probabilities, according to the difference in substrate detected, obtained from experimentation of the simulation engine by comparing the values obtained with results from laboratorial work:

- Negative: 80 %. If the substrate detected in this step is less than in the previous one, tumble almost certainly occurs;
- Zero: 15 %, if there have been no changes in concentration since the previous step;
- Positive: 10 %, if the concentration gradient is positive, the probability of occurring a tumble should be very low;
- Positive with memory: 5 %, if the gradient is positive and the bacterium was already running in the previous step, then there is almost no possibility of occurring a tumble. This last rule models a naturally observed memory property displayed by the *E. Coli*, which seems to react more strongly to a present positive stimulus when the previous stimuli were also positive.

When the state of each flagellum is defined, the motor class gets the data from each one and converts it to movement and direction.

If the majority of the flagella are moving counterclockwise, the bacteria will exhibit a run movement, and move forward in space in the previous direction.

Otherwise, the bacteria rotate facing a new angle $\theta$, calculated in the following way:

$$\theta_i = \theta_{i-1} + \sigma \quad , \quad \sigma \sim \Gamma(\alpha, \beta, \gamma)$$

$\alpha = 4$, $\beta = 18.32$, $\gamma = 4.6$
A number $\sigma$ extracted from a gamma distribution, with shape parameter 4, scale parameter 18.32 and shape parameter 4.6 will be added to the previous angle $\theta$ (parameters obtained from the work of Emonet et al. [13]).

With this function, it is possible to obtain tumble rotations that closely resemble those exhibited by real bacteria. However, the movement displayed between tumbles, in the run mode, isn’t exactly a straight line (Figure 5.1.9). The path has small direction fluctuations. To model this behavior, the main orientation $\theta$ suffers small perturbations through the application of a Brownian diffusion method.

The Brownian diffusion consists of substituting the angle $\theta$ by a new value randomly extracted from a normal distribution with $\mu=0$.

Since the exactitude of the application of the Brownian diffusion is not very relevant, but the performance of the simulation is, in the end, a simplified approach was chosen, that still provides realistic results:

A value $\sigma$ is defined as $\sigma = \theta_{\text{max}} - \theta$, where $\theta_{\text{max}}$ corresponds to the maximum value to diverge from the main direction $\theta$. Afterwards, for a given number of iterations $n$, we define

$$
    x_n = \text{randomBetween} \left( \mu - \frac{\sigma}{n} \mu + \frac{\sigma}{n} \right) \quad \text{and} \quad \theta = \frac{\sum_{i=1}^{n} x_i}{n}
$$

### 5.1.7. Strains and Mutations

Since one of the main fields of research involving the *E. Coli* bacteria is related with the study of naturally occurring or man-induced mutations, it becomes interesting for a simulation to be able to capture, at least a part of these phenomena. Furthermore, it would be interesting to take this a little further and allow bacteria with different mutations to share a medium and watch as they directly compete in the same environment. To implement this, the functionalities of *Colony* and *Mutation* were added to the application.

The *Grid* object creates a set of colonies. Each *Colony* holds reference to a set of *EColi* and a set of *Mutations*, meaning that all those bacteria will manifest those mutations.

The *Mutation* class is a simple interface. It defines two methods: *init* and *update*. The mutation’s *init* method will be called in each bacterium, after it is initialized in the regular way, to introduce definitive changes in variables...
such as the DEB parameters. The update function is called for each bacterium in every update cycle, to introduce modifications that will affect the following step.

To define a different strain as the result of a mutation, all one needs to do is create the corresponding class and have it implement the *Mutation* object.

The strains were defined by changing the standard DEB values obtained for the *E. Coli*. Since the life-cycle of viable strains of *E. Coli* is very similar, the main values that alter between them are the size of the cells (physical volume at division – $V_{w,d}$) and the reproduction rate (division time - $\tau$). In terms of DEB theory, that means change the structure value at division and birth and the metabolic rate. This last one can be achieved by multiplying all time dependant parameters by a constant value, the metabolic factor, which will make the life cycle steps occur faster or slower. It will affect the bacteria in the same way as if they were placed in higher or lower temperatures. Table 5.1.1 displays the source values and the changes made to accommodate them through the DEB parameters.
The *Viewer* class uses Windows Forms to create the structures necessary for the other classes to display information on the screen, for example, text labels and a graphical interface.

The *Drawer* class is created by the Program object and accesses Grid information to draw it on screen, using the functionalities defined by the *Viewer*. It is responsible for displaying the substrate concentration in the background (usually in shades of blue) and the bacteria in foreground.

The *EColi* class contains an array with the set of positions that it occupies at a given moment. It is calculated by using its length and direction, and updated each step to account for the movement. The Drawer gets those points and colors them on screen according to the color of the colony.

![Visualization](image)

*Figure 5.1.9 Visualization (the rasterization method was developed specifically for this work and follows an approach similar to the Bresenham algorithm)*

In Figure 5.1.9, at the left is an example of the output produced by this class. The squares displayed represent pixels which, in this case, correspond to an area of 0.2 µm X 0.2 µm. The shades of blue in the background represent the Grid bins, with stronger colors to match stronger concentrations. There are also three different-shaped colonies, colored in light blue, red and yellow.

### 5.1.9 Customization

To accelerate the development process of the presented solution, a set of customization tools were created to allow fast changes in the program without the need to recompile.

The main way to do this is by altering an xml file, which is loaded by a singleton class called Input. Since it is a singleton, the variables will be loaded one time upon creation, and every other class in the program will be able to access them. This configuration file defines a great number of important simulation parameters. It also enables and configures such functionalities as:

- Time control: It is possible to define the factor that relates real time with simulation time, and have the it run faster or slower than real time. This takes particular relevance since most starvation experiences last for several days. Using this functionality, they can be reproduced in minutes.
• Space control: Allows the definition of the world dimensions, Grid size and the relation between the world dimensions and how the simulation appears on screen.

• Environment variables: It is used to control parameters related with the recreation of the experimental environment, such as *E. Coli* and substrate concentration as well as the dilution rate.

• Colonies and mutations: The xml file contains a listing of all the colonies. Each colony can manifest zero, one or more mutations.

• Life-cycle functionalities: Using this property it is possible to turn on or off the functionalities that depend on the DEB theory, such as assimilation, reproduction, death and necrofagy.

• Chemotaxis parameters: It is possible to set the probability values for the chemotactic response according to the substrate concentration gradient, and provide different values for each colony.

• Graphing: Is used to set several properties of the generated output data used to plot the results.

It is also possible to draw a picture and use it to define the distribution of substrate in the medium. It should have a black background, and stronger shades of blue to define stronger substrate concentrations, just as it is later displayed by the drawer. This makes it very practical to define new concentration gradients and test them using the simulation.

A problem with this approach is that images are 2D and the simulation world is 3D. To keep the image functionality and apply it to the virtual three-dimensional world, in each layer of the 3D world, corresponding to a value of the z-coordinate, the substrate is distributed as defined by the picture. This propagates the 2D image in 3D, as shown in Figure 5.1.10, which means that, at the simulation start, the concentration is constant along the z-axis. However, after the simulation starts, the cubic bins are independently consumed by the bacteria, and this condition will not be maintained. In this example, as the image defines a circular concentration distribution, the 3D corresponding distribution will resemble a cylinder. The image approach was chosen instead of allowing the use of 3D concentration models, because this way it is simpler for a user with no training in the use of 3D modeling tools to easily introduce his own concentration gradients.
Figure 5.1.10 Image to concentration gradient implementation
5.2. Life Engine

The Life Engine, is a game engine developed in C++ by the Biodroid company\(^1\).

---

\(^1\) http://www.biodroid-entertainment.com/
The goal was to produce a tool that allows the fast creation of living beings and integration in a game environment. It applies the DEB theory and will include a database with a great number of commonly found organisms, so that it will be easy and quick to have them interacting in a game world or even as a scientific simulation. It will also provide tools for a user to design new organisms, through the application of new DEB parameters. In the end, the engine should present great flexibility, but ensure the accuracy of the simulations, since they will run according to DEB rules.

5.2.1. Architecture

Since the Life Engine is presently being developed, it is constantly changing, and for this application, an earlier version was applied to ensure the program’s stability. At the time this work will be submitted, the whole architecture will probably be entirely different.

The Life Engine architecture defined two different components. A simulation engine, which is called Life in the latest versions, and a function library, called DEBlib.

The simulation engine specific for the E. Coli simulation had already been developed in C# by the author, as described before. This being the case, only the DEBlib part of the engine was used. The population management provided by Life was not applied in this specific scenario.

5.2.2. DEBlib

The main class defined by DEBlib is the Organism. As the name suggests, is used to define a DEB organism. The EColi extends the Organism and defines an E. Coli individual. The DEB parameters specific for this bacterium are defined in the EColiDefine class. This is a singleton object, this way there will only be a set of DEB parameters for every EColi instance created.

The Species class defines the flux functions described in chapter 3 – time dependent equations for \( \dot{p}_A, \dot{p}_C, \dot{p}_S, \dot{p}_G, \dot{p}_I, \dot{p}_R \). It contains one or more structures and reserves, and zero or more maturities, reproductions and products (i.e. something that results from metabolic activities, such as heat, carbon dioxide of feces).

The differential equations themselves, for reserve, structure and maturity dynamics are defined in the corresponding object. Each of those classes contains a definition for the right-hand side of the time-dependent differential equation that describes its evolution. To make the time advance discretely in the simulation, with time intervals equal to the simulation step, the Species class contains a Solver, which implements 5 differential equation solving methods: Euler, RK2, RK4, AK2 and AK3. For each of the DEB state variables, there is also a history buffer with size 16. It is used to store past values returned from the differential equations, to aid in future calculations. The Euler method is the most commonly used solving method for the DEB
equations, since its error doesn’t affect the results in a relevant way and it weighs very little in the program performance.

5.2.3. Assimilation

In the version of DEBLib made available for this simulation it was assumed that the functional response \( f \) had always value 1.0. This means that the organism would always be assimilating according to the maximum assimilation rate \( \{P_{\text{amm}}\} \), which is only true for environments where the amount of substrate available is sufficient to ensure that level of food for every individual, no matter where it is. However, to maintain the accuracy of the simulation the bacteria should respond to the food level of the bin they were in, and it should be possible to carry out starvation experiments (no substrate). This was achieved though the manipulation of the DEBLib functional response variable.

Each cubic bin has a certain amount of substrate – mg of glucose. The functional response was calculated as the fraction of the maximum amount of food that an \( E.\ Coli \) can eat in a time step. We have:

\[
f = \frac{\text{availableSubstrate}}{\text{max Substrate}} = \frac{\text{availableSubstrate}}{j_{XAm} \times V \times \text{simulationStep}}
\]

where \( j_{XAm} \left( \frac{mg}{\mu m^3s} \right) \) is the maximum amount of glucose an \( E.\ Coli \) can consume, according to its structure value and the time interval corresponding to the simulation step. \( f \) may take values between 0 (no substrate) and 1 (when the amount of substrate is equal or greater than \( \text{maxSubstrate} \)).

5.2.4. Death

In the Life Engine version described before, the death event wasn’t yet defined. In the case of the \( E.\ Coli \) bacterium it is very simple to implement using the DEB parameters. When the bacterium reaches a structure value that is below a certain threshold, it dies.

In literature there is no reported work that provides this threshold value directly. Even measurements performed in starved \( E.\ Coli \) were hard to find. The value used is, therefore, a cross between morphological data gathered about starved \( E.\ Coli \) and fine adjustments to make the population curve match more closely the ones obtain with experimental procedures.
It has been observed that starved *E. Coli* bacteria tend to acquire an almost spherical look, as shown in Figure 5.2.3 [25]. In terms of quantifiable values, this would mean that their length would be close to their width. Since the average birth length of a *B/rA E. Coli* is 1 μm, and the width throughout its lifetime is 0.8 μm [9], this would mean that the structure death threshold would be around 0.8 times the structure value at birth.

To include this functionality in the Life Engine, another class was created and named *DEBDeathManager*. This object acts as a function library that receives an object of the *Organism* type and returns true or false, if the individual is dead or alive, respectively.

It is crucial to distinguish between the natural occurring death of the *E. Coli* cell, described in this section, from the dilution effect described in section 5.1.5. Here, the bacteria are removed from the simulation as a result of natural adverse conditions, in this case, starvation. These mechanisms apply to any simulation that involves the *Escherichia Coli* bacterium. However, the dilution factor tries to account for an effect that happens only in certain experiments, in which a quantity of medium is removed from the environment to gather data at different times in the course of the experimentation. This mechanism should be turned on or off, according to the laboratorial work that is being recreated.

5.2.5. Reproduction

Like death, the reproduction functionality was also not yet implemented in Life Engine. For the *E. Coli* bacterium, it happens much like death: when an individual hits a structure value that is two times what it had at birth, it will divide into two daughter cells. The *DEBReproductionManager* is implemented like the death manager. When the reproduction threshold is reached for an organism, the manager will respond with true when pooled about it. The population management is performed by the *EColi C#* application, which will add a new agent to the simulation.

5.2.6. Shape Correction

As mentioned before, the standard DEB equations only model the life cycle of isomorphic creatures. It was necessary to make the equations in the DEBLib more flexible about this aspect. To do this, the notion of shape correction was introduced.

In the *OrganismDefine* file, the shape property was added, which can take one of four values: isomorph, V0-morph, V1-morph or mixture. For the mixture case, the aspect ratio parameter (δ) must also be defined.

The shape correction function, \( M(V) \), is calculated when the organism is initialized, and applied as a multiplicative factor in all equations that include the maximum assimilation rate, \( \tilde{p}_{\text{am}} \), and energy conductance, \( \dot{v} \), parameters.
5.3. Integration

The C# simulation engine for the *E. Coli* and C++ DEB function library have been described in the two previous sub-chapters. All that there is left is to explain is how they were integrated to work together for the final version of the simulation.

The C# language provides a set of tools called InteropServices which allows the integration of C# applications with libraries developed in other languages. The most straightforward way to use this functionality is to have a dynamic library (.dll) developed in C++ and import functions defined in its header files (.h). Another project was therefore developed in C++, to generate the dynamic library. This project, called DEBInterface, consisted of one header and one source file. The header file defines a set of functions that allows the C# code to create and update DEBLib organisms, and pool it to inquire about life-cycle events such as reproduction and death. The source file makes the connection between the DEBInterface functions and the DEBLib functionalities. By doing this, and labelling the DEBInterface functions with the “extern” tag, the DLL can be imported from within the C# code, and the DEB functions will be available in the *E. Coli* simulation.

![Diagram](image)

**Figure 5.3.1 DEBInterface integration with the E. Coli C# simulation and the C++ DEB library**
6. Results

6.1. Evaluation strategy

The main evaluation method used to validate the implementation developed for this application will be to compare results obtained through carrying out simulations with reported experimental results. The set of parameters applied specifically to reproduce a certain experimental environment is thoroughly defined for each simulation.

The application is divided into three features, which will be evaluated individually:

- **Chemotaxis:** The standard *E. Coli* strain (Br/A) will be placed in an environment which matches the experimental work of Korobkova et. al [22], to examine the time spent in the tumble and run modes individually. The paths displayed by swimming bacteria will be compared with the experimental data obtained by Berg and Brown [5] to visually validate the chemotactic reactions. Finally, the emergent behavior of a bacterial colony will be displayed, to analyze its reaction to a medium with a simple gradient that should produce a gathering of bacterial cells in a central point of the environment.

- **Life-cycle:** The life cycle of the *Escherichia Coli* will be evaluated at an individual and population level. At an individual level, the growth curve of a single bacterial cell will be examined and compared to a real one [24]. At a population level, two different cases will be analyzed. The evolution of the bacterial colony will be compared to the curves obtained by Martinez-salas et. al [28] for the standard bacterial strain, in a medium with substrate abundance. A starvation experiment will also be carried out to evaluate the colony response to a medium without substrate, and the results obtained will be compared to experimental work of Farida K. Vasi and Richard E. Lenski [42].

- **Strains and Mutations:** The growth rates and sizes of the several *E. Coli* mutated strains will be cross-matched to values obtained in experimental reports [21] [20] [1].

6.2. Chemotaxis

The bacterial response to attractants (chemotaxis) was evaluated in three ways: the distribution of the time intervals spent in each state, the paths defined by the bacterial movements and the emergent behavior that arises from the final implementation as a response to an attractant gradient. All the results presented in this section were obtained with the same set of chemotaxis parameters:
<table>
<thead>
<tr>
<th>Substrate difference</th>
<th>Previous &gt; Current</th>
<th>Previous = Current</th>
<th>Previous &lt; Current</th>
<th>Previous &lt; Current and is running</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tumble Probability</td>
<td>80 %</td>
<td>15 %</td>
<td>10 %</td>
<td>5 %</td>
</tr>
</tbody>
</table>

Table 6.2.1 Tumble probabilities according to the detected differences in substrate concentration

Simulation step = 50 ms (time interval between simulation updates)

Sensor step = 100 ms (time interval between sensor updates, which include the analysis of the substrate concentration and adequate chemotactic response)

6.2.1. Motor rotations

The bacteria were placed in a medium without attractant, to reproduce the conditions described for the experiment. Results were obtained after a period of 150 minutes.

The flagellar motors can turn clockwise (CW), which produces a tumble, or counterclockwise (CCW), which produces a run. When switches between the rotation modes occur, the time spent in the previous mode is calculated, and 1 is added to the frequency registered for that time interval, in the corresponding mode. The charts in Figure 6.2.1 display the distribution of the relative frequencies of the time intervals spent in each of the flagellar motor orientations.

![Figure 6.2.1 Fraction of time intervals spent in CW and CCW mode. Real, from the work of Korobkova et al. [22] (left) and Simulated (right)](image)

The results obtained with the simulated chemotaxis closely resemble those displayed by real bacteria. The approach followed to reproduce the movement patterns was very simple, since it didn’t directly implement the protein chain responsible for the chemotactic responsible. The charts represent the result of the final implementation, after fine-tuning the tumble probabilities for each gradient detected, and the sensing intervals of the *E. Coli* bacteria.
6.2.2. Path

The images in Table 6.2.2 represent the 2D projections of the paths followed by one simulated bacteria as well as a traced sequence of loci obtained experimentally with real *E. Coli* bacteria.

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Real</td>
<td><img src="image1" alt="Path 1" /></td>
<td><img src="image2" alt="Path 2" /></td>
<td><img src="image3" alt="Path 3" /></td>
</tr>
<tr>
<td>Simulated with the developed simulation engine</td>
<td><img src="image4" alt="Simulated Path 1" /></td>
<td><img src="image5" alt="Simulated Path 2" /></td>
<td><img src="image6" alt="Simulated Path 3" /></td>
</tr>
</tbody>
</table>

Table 6.2.2 Paths of Real and simulated bacteria. Experimental data from Berg and Brown [5]

Visually, the rotation angles of the tumble movements seem to accurately reproduce those displayed by real bacteria. The run perturbations implemented through the Brownian diffusion method also provided very realistic results. The time spent in both run and tumble mode tested in the previous section, besides making sense numerically, also provided paths that match the ones observed in nature.

6.2.3. Emergent behavior

The final goal of the implemented chemotactic reactions was to have the bacterial cells respond to a glucose concentration gradient in the simulation environment. The ultimate test for the bacterial movement would be to obtain an emergent reaction to a simple circular gradient that guided the agents to its center, in the form of a bulk of bacteria visibly moving away from the periphery of the simulation world, and converging in the center, where the substrate concentration is higher.

Figure 6.2.2 displays a circular gradient, where the bluer areas represent a higher concentration of substrate, and the black zones are areas with no glucose.

The simulation was set in a space with $1.6 \times 1.6 \times 1.6$ mm ($4.1$ mm$^3$), and it lasted for 250 s.

There were 1994 individuals, moving at an average speed of $20 \mu$m/s.
Figure 6.2.2 *E. Coli* response to a circular gradient
6.3. Life-cycle

The life-cycle implementation of the *E. Coli* simulation, applying DEB theory, will be divided into three stages: individual and population growth with unlimited substrate, and population growth in starvation.

The tests were carried out using the Br/A (standard) strain of the *E. Coli*, with a doubling time (τ) of approximately 100 min, fed on glucose. The simulation space had dimensions 1.6 * 1.6 * 1.6 mm, and all the simulations started with around 2000 individuals. The chemotaxis movement parameters applied were the same as defined in the previous section.

6.3.1. Individual growth

According to the DEB theory, the isomorphic creatures usually have von Bertalanffy growth curves (Figure 6.3.1). The *E. Coli*, however, does not display this kind of growth, since it is a mixture between a V1 and a V0 morph, with shape coefficient δ=0.26. Therefore, it is important to test if the growth curve resulting from the insertion of the shape correction function $M(V)$ displays realist growth results.

Figure 6.3.2 displays *E. Coli* growth curves obtained from experimental results [24] and from the simulation. The simulated bacterial growth in volume appears as an almost linear function of time, verifying the correction of the implementation and of the shape correction function, when applied to an individual cell. The slight growth acceleration could be adjusted by slightly modifying the shape coefficient (δ) or it might be due to an imprecision in the DEB equations. However, the result came quite close to reality.

![Graph showing von Bertalanffy growth curves](http://www.fao.org/docrep/W5448E/W5448E04.htm)

*Figure 6.3.1* Growth curves based on von Bertalanffy growth. Image from [http://www.fao.org/docrep/W5448E/W5448E04.htm](http://www.fao.org/docrep/W5448E/W5448E04.htm)

![Graphs showing E. Coli individual growth](image2)

*Figure 6.3.2* *E. Coli* individual growth, real (left) [24] and simulated (right - The line represented in this chart is a linear regression obtained from the displayed dots. The dots themselves correspond to data directly extracted through experimentation.)
6.3.2. Population growth

To test the evolution of the *E. Coli* population through time, the experience from the work of Martínez-Salas et. al. [28] was recreated using the simulation program. In order to do this, the bacterial colony had to be synchronized by size, as it happens in the corresponding laboratorial experience. However, to create a bacterial colony in which all the bacteria were initialized with the same size was not enough. An approach as simple proved unrealistic, since the separation of bacterial cells according to their dimensions is not an absolutely precise method. Using this approach, the number of individuals in a population doubled almost instantly.

To eliminate this error, the cells were synchronized by size, but a small error coefficient was added. The initial structure value of all the bacteria in the population was defined as:

\[ V_i = \frac{V_d}{2} + RandomBetween\left(0, \frac{V_d}{10}\right) \]

![Graph of E. Coli Br/A growth](image)

*Figure 6.3.3 Simulation of E. Coli Br/A growth with unlimited substrate (t=100 min) using the developed simulation engine*

![Graph of E. Coli Br/A growth](image)

*Figure 6.3.4 E. Coli Br/A growth, from the laboratorial work of Martínez-Salas et. al. [28] (t=45 min)*
The population growth curves for both tests exhibits periods with no division, characterized by straight flat lines in the chart, followed by multiplication periods, where the relative cell number appears to increase linearly as a function of time (this effect happens because of the logarithmic scale applied).

The growth curve displayed by the simulated bacterial colony is very similar to the real population in shape. However, the scale is different from the data gathered in the experimental work. This is to be expected, since the bacteria used for each of the tests had different doubling times ($\tau = 45$ min for the real *E. Coli* and $\tau=100$ min for the simulation strain). The results can be considered to be positive despite this fact, since the cycles are equivalent when matched to the doubling times for each of the colonies. The first doubling step ends at time $t = \tau$ for both colonies, and the second starts at approximately $t = 1.9 \tau$, which means that the results obtained are verified by the experimental data.

### 6.3.3. Starvation

Since the model created responds to the amount of food present in the environment through the application of the DEB theory, it is possible to carry out starvation tests using the proposed solution. These were achieved by placing the standard bacterial strain in an environment with no substrate.

Figure 6.3.5 displays the simulated population curve obtained for the Br/A colony without the presence of glucose in the environment.

![Figure 6.3.5 Population curve in starvation](image)

The chart displays a short growth and multiplication period in the first 10 hours, in which the reserve is applied towards structural increase, allowing the division of the bacteria whose structure threshold for duplication is crossed.

This is followed by a stagnation interval, represented by a straight line. At this point, continuous shrinking occurs. This means that the structure value will decrease without crossing the structure death threshold. The overall
structure of the population becomes smaller, but no deaths can be observed. The work of Barcina et. al. [3] states that continuous shrinking can be observed in bacteria placed in starvation conditions, which corroborates the results obtained through the application of the DEB theory.

Afterwards, the cell number begins to decrease, as the structure threshold for death starts to be crossed. The population can survive for about 5 days, completely disappearing after this time. The small fluctuations in the downward curve might result from the implemented necrofagy mechanism, which keeps some of the bacteria alive feeding on dead colony members throughout this period of time.

![Population curve in starvation, with several E. Coli strains, from the work of Farida K. Vasi and Richard E. Lenski](image)

*Figure 6.3.6 Population curve in starvation, with several E. Coli strains, from the work of Farida K. Vasi and Richard E. Lenski [42]*

When compared to a population curve in starvation from experimental data, it is possible to notice some differences between what happens in nature and the results of the simulation. The real populations last longer, they display no stagnation period and the general shape of the curves is different. This might be due to several factors.

First and foremost, the bacterial strains used in the simulation and in the laboratory are quite different. The simulation applies a standard Br/A E. Coli strain, while the strains used in the experiment are mutants selected from previous tests, which gives them a greater endurance to adverse conditions.

The chart in Figure 6.3.6 also displays evident cell number increase in certain points after the descending period. This is probably a cause of the naturally-occurring mutations that take place in the E. Coli bacterium when it is subjected to non-optimal conditions. Although the developed simulation allows the creation of mutations that could possibly reproduce the naturally-occurring ones, it does not allow a colony to change in the course of a single experiment. It is not possible with the implemented solution to have a colony adapt to its environment in the course of a test.
Another known fact is that living beings tend to decrease their metabolic rates when subjected to substrate shortage. The DEB theory does not yet predict this effect in its equations. The rate at which reserve is utilized does not depend on substrate availability. This factor has a tendency to cause DEB populations to extinguish faster than real ones [32].

Lastly, the application does not allow the simulation of numbers of individuals comparable to a laboratorial sample. In this example, we have a sample of 1 ml (1 cm$^3$), and it is hard to have a simulation sample bigger that 1 mm$^3$, without having performance issues manifesting in a standard machine.

In general, the starvation mechanisms that are now implemented do not suffice to reproduce what happens in reality, requiring further work towards implementing new features and refining those that are already working.

### 6.3.4. Constant glucose input

The final standard *E. Coli* scenario was the constant substrate input test. In a laboratory, a certain number of *E. Coli* were placed in a medium with a small quantity of glucose and, at constant time intervals, a certain quantity of glucose was added to the environment. This allows the bacteria to multiply, by not indefinitely, since the population growth is limited by the amount of substrate in the environment.

![Graph showing population growth with constant glucose input](image)

**Figure 6.3.7** Simulated population growth with constant glucose input. The environment was initially set with 450 B/rA *E. Coli* individuals and 1.28 picograms of glucose, in a simulation space of 0.4 X 0.4 X 0.4 mm. The test lasted for 50 days, and an amount of 1,28 pg of glucose was added each 60 minutes. The dots represent numerical data directly extracted from the simulation and the lines are 6th degree polynomial regressions obtained from those point sets.
Figure 6.3.7 displays the behavior of a simulated bacterial colony in the afore-described conditions, using the developed simulation engine.

The graph shows sinusoidal fluctuations of the glucose amount and *E. Coli* population, with a difference of approximately T/2 between the two curves. This means that, when the *E. Coli* population increases, the amount of glucose decreases. This is to be expected, since a higher number of individuals feeding makes the substrate amount in the environment decrease. At a certain point, the substrate present in the environment is no longer sufficient to feed all the individuals in the simulation, and the population size begins to fall, decreasing the speed at which the glucose is consumed. This makes the substrate concentration rise again. These shifts in population and substrate happen periodically, with a tendency towards stabilization.

These results show that, although the starvation implementation is not perfect, the population displays realistic and plausible results in substrate-limited-growth conditions.

### 6.4. Strains and Mutations

After having the standard *E. Coli* model fully functional (B/rA), the process applied to generate the parameters for the rest of the bacterial strains was very direct.

Firstly, the length at division (Lₐ) was defined from literature data. From its value, the structure value at division was computed using a proportionality constant with the standard strain at the power 3, as follows:

\[ V_{d_{-A}} = \left( \frac{L_{d_{-A}}}{L_{d_{-S}}} \right)^3 \times V_{d_{-S}} \]

where A is the strain we want to model and S is the standard strain. This approach can be followed as long as the initial reserve density [E] for all bacterial strains is equal to the B/rA reserve density at birth [E₀], guarantying that the reason between structure and reserve remains coherent.

From there, the metabolic factor is applied, to generate realistic mean division time (τ) values. This is done by sequentially trying out metabolic rates, and stopping at the one that provides most realistic results.

This approach was followed to define the set of DEB parameters, usable with DEBlib for 3 more slow growing *E. Coli* strains, and 2 fast growing ones. Each of the strains is coded as a C# object that implements the mutation class.

Table 6.4.1 displays the results obtained for all the strains simulated, as well as the set of DEB parameters and literature data used for comparison. The average and standard deviation for the simulation values were generated by a population of 10 tests, where the target strain was placed alone in the
simulation environment. For each of those tests, the simulation ran for an amount of time that allowed, at least, 5 division cycles. The bacteria were fed on glucose with no limit in terms of substrate.

As explained before, the metabolic rate affects all the parameters that are time dependent. In DEB notation, those which are dotted: \( \{ \dot{p}_T \}, \{ \dot{p}_M \}, \dot{v} \) and \( \{ \dot{p}_{Am} \} \).

<table>
<thead>
<tr>
<th>Strains</th>
<th>Slow Growing</th>
<th>Fast Growing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data</td>
<td>B/rA</td>
<td>B/rK</td>
</tr>
<tr>
<td></td>
<td>Simulated Average</td>
<td>106,3</td>
</tr>
<tr>
<td></td>
<td>Standard Deviation</td>
<td>15,35</td>
</tr>
<tr>
<td></td>
<td>Simulated</td>
<td>2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DEB</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>( \kappa )</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>{ \dot{p}_T }</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>( [\dot{p}_M] )</td>
<td>1,13E-14</td>
<td>2,26E-14</td>
<td>1,53E-14</td>
<td>5,31E-15</td>
<td>4,27E-14</td>
<td>5,54E-14</td>
</tr>
<tr>
<td>( [E_G] )</td>
<td>5,08E-09</td>
<td>5,08E-09</td>
<td>5,08E-09</td>
<td>5,08E-09</td>
<td>5,08E-09</td>
<td>5,08E-09</td>
</tr>
<tr>
<td>( \sigma )</td>
<td>0,30</td>
<td>1,75</td>
<td>0,52</td>
<td>0,40</td>
<td>0,30</td>
<td>0,30</td>
</tr>
<tr>
<td>{ \dot{p}_{Am} }</td>
<td>1,25E-04</td>
<td>2,50E-04</td>
<td>1,69E-04</td>
<td>5,88E-05</td>
<td>4,73E-04</td>
<td>6,13E-04</td>
</tr>
<tr>
<td>( [E_M] )</td>
<td>5,91E-09</td>
<td>5,91E-09</td>
<td>5,91E-09</td>
<td>5,91E-09</td>
<td>5,91E-09</td>
<td>5,91E-09</td>
</tr>
</tbody>
</table>

| Metabolic Factor | 1 | 2 | 1,35 | 0,47 | 3,78 | 4,9 |

Table 6.4.1 Strain modulation results

* No data was found about the length at division for the fast growing E. Coli strains, so it was defined as the standard E. Coli value
The results obtained for the length at division ($L_d$) value are completely accurate. This was to be expected, since this value is directly mapped to a DEB parameter, the structure threshold at division ($V_a$).

As for the mean division time, by manipulating the metabolic factor, it was possible to obtain values that came very close to those displayed by real bacteria. The accuracy of the results is verified by the small variations obtained for every $\tau$ value, when compared to the average.

After modelling and verifying the correctness of the 6 $E. Coli$ strains, some tests were carried out to see how the different strains reacted when placed in the same environment.

### 6.4.1. Unlimited growth

Firstly, a simple simulation was ran, with 4 $E. Coli$ strains with different doubling times and no substrate limit. The simulation started with 64 individuals for each colony and ran for about 9 hours.

![Graph](image)

**Figure 6.4.1 Simulation with no substrate limit with $E. Coli$ B/rA ($\tau = 109$ min), B/rK ($\tau = 100$ min), B/rF26 ($\tau = 240$ min) and MC1000 ($\tau = 21$ min)**

As it was to be expected, every colony displayed exponential growth, with much faster cell number increase for those that had smaller doubling times, with the exception of the B/rK strain, which in the end displayed a slightly smaller growth than the B/rA colony. However, this is just a result of the randomness of the simulation, since the difference is not significant, and the B/rK strain even exhibited greater cell number increase around minute 60.
6.4.2. Individual structural growth

In the same test conditions, another simulation was ran to compare the structure evolution at an individual level between different strains.

![Graph showing structure evolution through time for different strains](image)

Figure 6.4.2 Structure evolution through time for strains B/rA (Ld = 2 μm, τ=109 min), B/rK (Ld = 3.6 μm, τ=100 min) and MC1000 (Ld = 2 μm, τ=21 min)

Figure 6.4.2 displays the results of individual growth for 3 bacterial strains, with different doubling times and structure division thresholds. The peaks in the three lines represent the points where the bacterium divided. At those points, a daughter cell was created with half the structure and reserve of the mother. As a result, the mother also becomes half its size at division. The B strains have approximately the same division time, so they have the same number of peaks. However, strain B/rK is larger and has greater size fluctuations. The MC1000 strain has around 5 times the number of peaks of the slow growing bacteria, since its doubling time is around 5 times smaller.

6.4.3. Starvation

Afterwards, the reaction of 4 different bacterial strains to starvation was tested. 121 cells of each strain were placed in an environment without substrate for 11 days.

As it had happened in the starvation test with the B/rA strain, the chart displays a clear population growth during the first half of the first day, followed by stagnation, for every strain. The B/rK colony, however, having a length at division almost 2 times larger than the rest of the bacterial colonies, did not increase the structure value enough for a single individual to reproduce, stepping directly into the stagnation phase.
Figure 6.4.3 Starvation test with 4 bacterial strains

After the stagnation period, all strains go into a descending phase. The CM1000 colony is the first to be extinct, because it needs to keep a high metabolic rate in order to divide almost 5 times as fast as the B/rA and B/rK strains. The B/rK disappears in second because, although its division time is the same as the B/rA strain, it needs to maintain a higher metabolic rate in order to get to almost the double of the size of the B/rA strain in the same time interval. This means that the reserve is consumed in a higher rate than the rest of the bacteria, as well as the structure when the reserve runs out. This mechanism also explains why the B/rF26 E. Coli is the last to disappear, since it has the average size of an E. Coli and a division time that is more than the double of the rest of the slow growing cells.

6.4.4. Glucose-limited growth

As it was said before, one of the main advantages of using biological simulations is to obtain a broad range of result data faster and with fewer costs than through laboratorial experimentation. The simulation performed in this section is an example of how these properties can be applied.

The most direct way to generate competition between different bacterial strains is to place them in substrate-limited conditions. Figure 6.4.4 displays the behavior of 4 different E. Coli strains, one fast-growing population and three slow-growing, subjected to similar conditions to the ones described in section 6.3.4.
As it had happened with the B/rA strain, the substrate concentration displays an approximately sinusoidal behavior.

The fast-growing MC1000 strain had a clear advantage over the slow-growing ones. In the initial growth phase, these individuals multiplied much faster than it happened with the rest of the strains, spreading through the simulation space as they grew, consuming almost all the substrate available. After that point, the B/rA and B/rF26 populations stabilized. The MC1000 continued to display the sinusoidal behavior, inverse to the glucose concentration, as it had happened before with the isolated B/rA strain. The B/rK *E. Coli* has the double of the size of the other strains, and is also slow-growing. It had no chance to get enough substrate to allow any population increase, and became extinct in the first population-descendant period.

After synchronizing completely with the glucose, around the 700th simulation hour, the MC1000 strain showed a tendency towards stabilizing, as did the glucose amount, displaying smaller and smaller cell number amplitudes.

To carry out such an experimental work, to obtain similar results, it would be necessary to set up an experiment with constant glucose input, 4 different strains and find a way to measure the dimensions of the population of each strain throughout the entire experiment, that would last for 55 days.

By using the proposed simulation solution, all that was needed was to define the quantity of each strain present in the environment, the rate and amount of glucose input, and the displayed data was generated. Furthermore, these results were obtained after running the simulation for only 30 minutes.
7. Conclusions

The main goal of the developed application was to implement a biological simulation that accurately reproduced the behavior of the *Escherichia Coli* bacterium in a multi-agent environment.

The DEB theory was chosen to define the bacterial life-cycle. This was very useful for the project, since the implementation laid upon a stable background theory, with well-defined methods for recreating the desired behaviors. It is also useful for the DEB community to count on a working computer model that uses the theory and helps to ensure its accuracy and applicability in different conditions. To expedite the development process, instead of writing all the equations from the beginning, Biodroid’s Life Engine was improved and applied to the project. Although almost all the metabolic DEB equations were already written in the DEBLib component of the engine, it still lacked in the using of the state variables to trigger life-cycle events such as birth, division and death. This part of the functionality was added by the author in the course of this work.

In parallel with this part of the work, the basis for the simulation itself was developed in C#. In the end, it incorporated a fully configurable 3D multi-agent environment with a visual interface. The events defined by the DEB state variables thresholds were triggered using the ION framework, as well as the basis for the communication between objects that share the virtual world. The C++ DEBLib functionalities were integrated into the C# code, creating a simulation engine that has all the features of DEB theory connected to a virtual world with a visualization layer.

The simulation was tested in two phases, to evaluate two different features.

In a first phase, the chemotactic reactions were evaluated by comparing the results from the simulation with laboratorial experiments. The time intervals spent in each of the rotation modes were very similar to the ones reported in [22]. The paths described by the moving individuals closely resembled those defined by real cells [5]. The emergent behavior of the bacterial colony reproduced the displacement of *E. Coli* bacteria navigating towards a concentration gradient, without adding rules more complex than simply detecting concentration differences in the environment. A more complex algorithm would be unrealistic having in mind the simplicity of the object of study. The approach followed was very simple, when compared to the related work, where the whole protein chain responsible for the chemotactic reactions was virtually recreated. However, the results presented were just as accurate in the tests performed. The high level of abstraction means a greater effort towards parameter estimation, since the tumble probabilities are hard to directly extract from an experiment. However, in terms of performance, the simplicity of the solution takes great relevance.
In the second phase, the life-cycle of the bacteria was tested and compared to results from the literature. The bacterial growth was very similar to the laboratorial results, at an individual [24] and population [28] levels. This validates the applicability of DEB theory to similar situations and confirms the accuracy of the implemented model. The starvation tests did not come out as positive, since the population curves extracted were very different from the ones obtained in [42]. Although the genetic profile of the individuals is quite different, this is not enough to account for the disparity of results. Further work is needed to perfect this situation, in terms of adding auto-mutation functionalities and changing the DEB metabolism rules in substrate shortage. Despite this fact, the results of the substrate-limited growth simulation were very coherent.

Using the implemented solution, it was also possible to develop a set of bacterial strains, both slow and fast growing. The numeric results obtained from a relevant number of tests came very close to the results gathered from literature [21] [20] [1]. From there, 4 different simulations were carried out, to showcase the different ways to use the solution.

Generally speaking, the solution presented in this work adequately answered the problem, which was a coherent, accurate biological simulation. Tools were also created to allow the extensibility of the model to incorporate other functionalities, such as different strains and mutations. All the relevant simulation variables are exposed to the user, so that a great number of tests can be performed without the need to recompile the code. Lastly, the solution is connected to a biological simulation engine, and the respective theory, which are in constant evolution. This means that the developed tool, with minor adjustments, has the potential to be as up to date in any time the future, as it is in the current moment.
8. Future work and possible applications

8.1. Future work

The implementation presented in this document was built in a way that allows future extensions in terms of content and functionality.

It could be interesting to implement the life cycle of other organisms and add them to the simulation. This would involve implementing a new class for the agent, based in the *E. Coli* one, and add the corresponding set of DEB variables in the DEBLib. This feature could be enhanced by the introduction of interaction between those organisms, such as trophic relations. As an example, the bitrophic chain experiment reported by Dent et al. [10] would be relatively straightforward to reproduce using this computer model. The work of Marco Rodrigues [32] could serve as a base to extract the movement of the *Dictyostelium Discoideum* amoeba, as well as its DEB parameters and the emergent cooperation behaviors displayed in substrate shortage situations.

Having this example in mind, it would also be useful to implement the concept of trophic relations in DEBLib. This would mean adding methods for an organism to consume another, and gain energy according to the prey’s reserve and structure.

The DEBLib could also be extended to include reproduction buffer handling rules. This would allow the modeling of organisms that don’t reproduce by division. To introduce this functionality, the energy should be allocated from the reserve to the reproduction buffer once the being is fully matured. After the reproduction buffer is full, the embryo stage would be fairly simple to reproduce. The standard DEB rules should suffice, as long as the functional response is kept at 0, meaning that the organism doesn’t assimilate, living only of the reserve available within the egg. The fetus case would be more complicated, since it would require a permanent relation between child and mother, which transfers part of its reserve continuously to the fetus, until birth.

Although the changes made to the DEBlib in the course of this project introduced death by starvation, the notion of time accumulated damage is not yet included, therefore it is not possible to reproduce death by old-age or as a consequence of an event that crippled the organism, but not in such a way that it dies instantly. It would be essential to implement this functionality in order to simulate the life cycle of complex organisms.

It is known that besides navigating towards attractants, the bacterial chemotaxis allows organisms to move away from repellents such as toxic substances. In terms of movement, the introduction of toxic elements is very simple. It would simply be necessary to represent them as negative concentrations. Since the bacteria respond to differences in concentration,
not to the concentration values themselves, the result would be as realistic as an environment with nothing but attractant. This functionality was not included in the project because it made no sense to display a repellent in the environment, if the toxicity effects upon the bacterial life cycle were not implemented. In this way, a possible extension to the current implementation would be to introduce the toxicity effects of selected compounds upon the DEB variables [2].

The final solution includes a set of mutations that can be introduced individually or simultaneously in bacterial colonies. However, in nature, mutations occur spontaneously in the E. Coli. After developing a larger set of mutations, it would add realism to the simulation if they occurred spontaneously. It would be particularly interesting to include stress-resultant mutations and have them appear spontaneously when the bacteria were submitted to the corresponding stress conditions, such as temperature, starvation, pH, etc. Furthermore, it has been experimentally shown that the E. Coli bacteria can incorporate mutations from dead individuals. After death, the cellular wall is decomposed, and the DNA lies in suspension in the environment. The live members of the colony can eat the DNA, and absorb the energy of the sugars it contains (necrofagy) [14], or incorporate the material into their genetic code, assuming whichever mutations the dead bacterium had suffered. This is called natural competence [4] and takes a very important role in stress conditions such as the ones that have been previously described.

Finally, although the simulation environment is 3D in logic, it is displayed in 2D. If the display mode were changed to look like a three-dimensional space, it would add visual realism to the simulation and allow an easier understanding of what is happening in the world.

8.2. Future applications

The Escherichia Coli bacterium is an organism that has always lived in a symbiosis with the human beings. However, many E. Coli strains are pathogenic. Having in mind this strict relation between this bacteria and humans, it becomes all the more important to know as much as we can about it.

The developed simulation is meant to be used as a tool to model virtual environments populated by E. Coli bacteria, and allow a faster, more practical and cheaper first approach towards the investigation of its behaviors in several conditions. The strains which are already modeled are usable, and serve as an example of how other mutations can be introduced in the simulation by manipulating DEB parameters.

In an academic environment, the model can also be used to help in the familiarization with DEB theory. The E. Coli bacterium is very simple, and well known in most biology fields. The simulation engine can be used as a tool that allows the fast adjustment and manipulation of DEB parameters, to
provide a better grasp of the parameters themselves and what is involved in modeling using the DEB theory.

As far as DEB theory itself goes, the model can be applied as a practical way to test the use of DEB parameters to reproduce mutations in *E. Coli* bacteria, or even other microorganisms, with a few adjustments.

Lastly, the model can be used as a tool for anyone who has some programming experience to easily simulate a great number of organisms. The use of the DEBLib allows the practical creation of DEB creatures, and the infrastructures already developed for the project make it simple to introduce them as agents in the virtual environment. From there, there is no limit to where the work started here might be taken.
9. References


34. Schmcale, H., Synthesis of Human Insulin in Bacteria – An example for applied recombinant DNA-technology, VCH Publishers INC., 3 (1985) 214-221


37. Sourjik, V., Receptor clustering and signal processing in *E. coli* chemotaxis, TRENDS in Microbiology 12 (2004) 569-576


