E.coli in Hostile Environments

New Mexico

Supercomputing Challenge

Final Report

April 4, 2007

Team 52

Los Alamos High School

Team Members

Stoyana Alexandrova

Iliana Alexandrova

Teacher

Mrs. Diane Medford

Project Mentors

Dr. Anny Usheva

Mr. Ludmil Aleksandrov

Table of Contents

Executive Summary

Our scientific problem is to explore the effects of hostile environments on the adaptation and evolution of bacteria *Escherichia coli*. Our hypothesis is that if a bacterial colony is placed in a special hostile environment, then due to:

- Natural selection,
- The expression of a new discovered gene which preserves the bacterium alive in hostile environments [2,3], and
- The recently discovered lack of functional immortality in *E.coli* leading to a complex mechanism of inheritance [4],

the colony will evolve, and after certain number of generations will acquire new characteristics.

The problem is important because of its connection with recent research showing that *E.coli* may be eventually used for cancer treatment and other medical purposes, if its usual characteristics can be changed [1].

We created an Evolutionary Agent-based computer model of a bacterial colony growing in special hostile environmental conditions. The code is written in ANSI C, and we used the MPI library to run it on a Linux cluster at Harvard Medical School. We modeled a colony growing up to 10^7 bacteria in a 10^4 by 10^4 lattice with initial random distribution of both the food quantity and the food density placed on the lattice. We modeled our agents (*E.coli* bacteria) to have the most necessary genetic traits, such as energy of the bacterium, speed of eating, number and length of flagella (defining their average speed), level of energy needed to divide, necessary conditions for expressing the preservation gene, etc. In the core of our simulation is the random-walk type motion of the bacteria, which is the general accepted way the bacteria moves. We used separate Gaussian distributions for each of the genetic characteristics of the bacteria. The average value and standard deviation of these distributions are inherited asymmetrically when the bacteria divides, according to the recently discovered complex mechanism of inheritance [4]. We incorporate into the model the new idea of “preservation” – an *E.coli* bacterium expresses a gene to accommodate itself to a starving environment. The

expression of this gene leads to a change in the cell's metabolism and as a result the bacterium preserves a lot of its energy, does not divide, and ultimately - survives the harsh conditions [2, 3]. The starvation time and the energy level required for the expression of this preservation gene depend on the genetic characteristics that are inherited by Gaussian distribution.

First, we established and verified our model by fitting all the parameters in order to mathematically repeat the experimental data by Jonathan E. Visick [5] for *E.coli*'s growth in a low food level environment. After that we started changing the gravity level. The gravity is physically affecting the bacteria by increasing their weight, which results in the bacteria's lower mobility, just like a higher pressure and a greater friction coefficient in the medium would slow down *E.coli*. We simulated hyper gravity (or high pressure/friction) assuming that this hostile condition will proportionally decrease the average speed of the bacteria and proportionally increase the average energy spent trough their movement.

We analyzed our data over all the 40 simulations we did (about ~500 CPU hours), some of them with different realizations of the initial conditions and different random numbers, to obtain statistically average values of the *E.coli*'s characteristics. Our results clearly indicate that in a hostile environment, the growth and the average inherited characteristics of *E.coli* changed drastically.

Most importantly, we showed the presence of a memory effect. After 100 generations, a colony grown in hyper gravity conditions will develop greater average number and length of flagella. This will make *E.coli* faster and more physically fit if put back into normal gravity conditions, which can be used in practice.

Introduction

Biologists today can genetically modify bacteria and other organisms to serve for beneficial medical purposes [1]. This can be done either by directly interfering with the genomes of the organisms (gene transfer through transformation, transduction, conjugation, plasmids, etc.) or by using the procedure we are introducing in our project – letting the organisms evolve under certain controlled conditions and simulating natural selection to choose the fittest organisms. We can control these organisms' characteristics and genes by changing the environmental conditions. In this way, a biologist working in the medical field can create useful specific agents that can perform a desired task; the agents can be used for tests and experiments for the benefit of, for example, immunological diseases. Although such research is not in the scope of our project, our project shows that this method does result in new characteristics. The reason we chose our topic of research is because we believe that this new method provides a very effective way which can be used to change the traits of biological agents (bacteria and other cells). These new agents can be used in the medical research and the betterment of human health. Also, this project is a continuation from a project we participated two years ago. We competed in the 2004-2005 NASA Hyper-G Competition, and submitted a proposal of *The Effect of Hypergravity on the Reproduction Rate of E.Coli K-12*. We proposed an experiment that would collect data of *E.coli* growth rate in a NASA centrifuge, and hypothesized that the gravity will slow down the bacteria's speed; thus, only the genes for faster speed will survive, and this will result in physically stronger bacteria that will reproduce massively if put back into normal gravity conditions. Some of the concepts explored in that project play major roles in this experiment.

The Scientific Problem

We are exploring the effects of “hostile” environments on *E.Coli*’s characteristics. We have taken a computational approach towards the problem and we are using agent-based modeling to mimic real-life laboratory conditions of an *E.coli* colony developing in a laboratory Petri-dish under controlled conditions. A hostile environment in our model is one with different than the normal gravity level and starving food amounts and distribution. More specifically, the hostile environment in our project mimics starvation conditions because both decreased speed and low food amounts starve the bacteria. Usually, a hostile environment inhibits the normal growth and development of organisms, forcing them to either die or accommodate to the new conditions. In our model, the different food and gravity levels both represent a hostile environment, because they both trigger the same effect in the agents by preventing them from getting enough energy. That is, by means of simulated natural selection, the bacteria in the modeled hostile environments either die, or they react to it by expressing a new gene that helps them preserve their energy and survive in the environment. This is in the core of our experiment – the bacteria are triggered by an environmental factor to express a new gene so that they can accommodate to the new environment and pass on their genes to the next generation. This concept is very important because it is in the core of biology, natural selection, and Charles Darwin’s theory of evolution. We based our presumption of this specific effect on *E.coli* on Jonathan E. Visick’s experimental results of *E.coli*’s normal growth under normal conditions but limited food supply and other research, explained later in the report. Given that bacteria are put in a hostile environment, they will develop differently than normally and some will be the fittest -

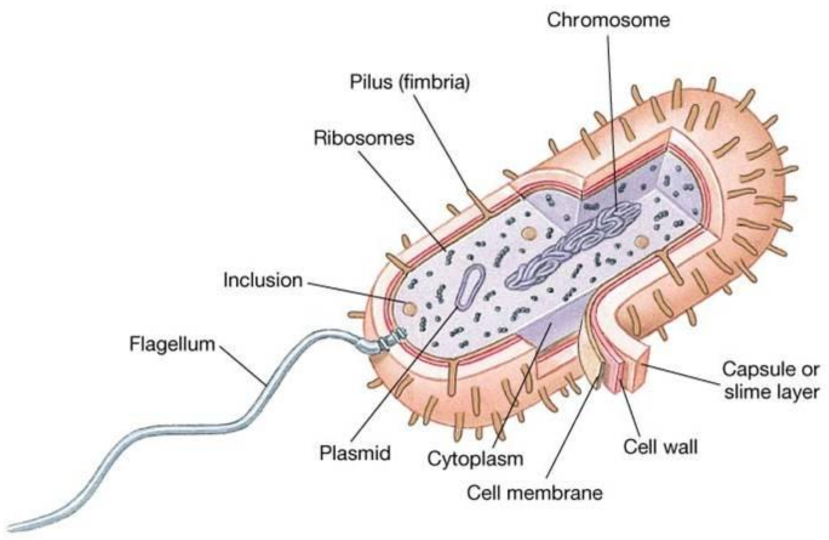
finding food for survival, dividing successfully, and being physically fit for the particular environment with the specific gravity level and food content. Bacteria that are able to survive in an environment with hyper gravity (which increases the hostility of the environment) will produce offspring with similar traits and the resulting colony will be potentially well-fit for surviving under normal conditions. This is how biologists can make physically strong bacteria that can survive under harsh conditions – by creating a sample that is physically fitter than the wild-type sample from the normal conditions. By letting the sample develop under a hostile environment and changing the bacteria's characteristics, biologist can aim to modify or even create specific desired traits in the agents, which can then perform concrete directed actions (i.e. living in a host human and releasing antibiotics).

Biology and *Escherichia coli*

General Background

Escherichia coli are the most widely used microorganisms for biological research experiments in microbiology. They represent a model organism, or a species that is widely used to explore potential causes and treatments for human diseases [14]. Ironically, when people hear about *E.coli*, they usually think of how dangerous it is and associate it with awful diseases and death. In reality, the bacterium has many useful purposes. For example, *E.coli* is considered the primary indicator of recent fecal pollution. Furthermore, it is also found in the healthy intestinal tracts of humans and other vertebrates. *E.coli* in fact crowds out disease-causing bacteria in the human body and produces Vitamin K. Even its abbreviation "*E.coli* ", corresponding to the Latin "of the colon", emphasizes the bacteria's benevolent roles. Only a mutant of *E. coli* (the strain *E.coli* O157:H7), found in undercooked beef is very dangerous, and can cause a severe disease that might become fatal in small children and elderly people [6]. Except those terrible incidents, in most other cases, *E. coli* is a friend of humans and can be used for beneficial medical purposes. The species is used extensively in recombinant DNA research because it has been genetically well characterized [7], and its members are commonly used as host cells for cloning segments of genomic DNA [8]. Geneticists study *E.coli* so intensively because of its small genome size, normal lack of pathogenicity, and ease of growth in the laboratory [9].

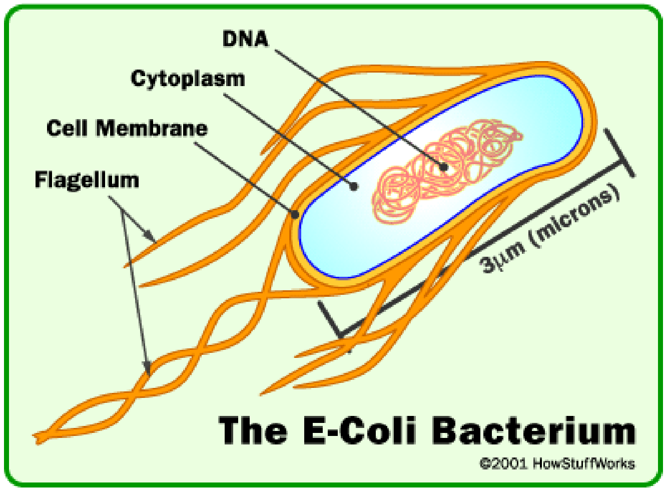
Structure



Picture 1.1 caption: **Picture 1.1** Prokaryotic cell (in courtesy of [38])

A gram-negative, facultatively anaerobic, nonspore-forming bacillus [10], *E.coli* is a prokaryotic member of Domain Bacteria, and more specifically, it is a member of the major bacterial group from phylum Proteobacteria [11]. The bacteria are found to live in a wide range of environments – from vertebrates and in the ground, in soil and in decaying plants, to the edges of hot springs [12]. The most favorable environmental temperature for the species is about 35 °C (95 °F).

As a facultative anaerobic organism, *E.coli* makes ATP (the molecule that is the cell's energy source) by aerobic respiration if oxygen is present but is also capable of switching to fermentation [13]. As a Gram-negative bacterium, *E.coli* does not retain a violet color when Gram stained. This indicates that *E.coli* can be potentially pathogenic, and can cause disease in a host organism. This pathogenic capability is usually associated with certain components of Gram-negative cell walls, in particular the lipopolysaccharide layer [15]. Furthermore, *E.coli* are rod-shaped, elongated, 1–3 µm in length and 0.1–0.5 µm in diameter [12]. They have a relatively simple cell structure that can vary, but has the general form of all prokaryotes (picture 1.1, 1.2). A typical bacterium has:



Picture 1.2 *E.coli* cell (in courtesy of [39])

- a jelly-like outer coating called a capsule,
- cytoskeleton,
- 5-20 flagella (important locomotion organelles),
- many pilli attached on the capsule,
- a bacterial chromosome located in a nucleoid (region where the cell's DNA is located with no membrane),
- ribosomes (synthesize proteins),
- a plasma membrane enclosing the cytoplasm,
- and a cell wall outside the plasma membrane. [16]

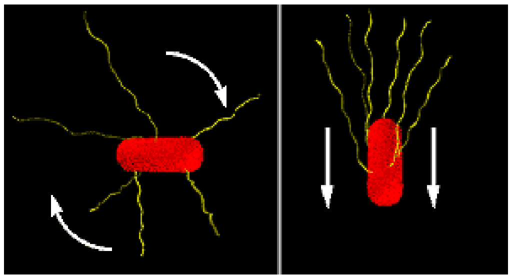
Metabolism

Research on *E.coli* indicates that the bacterium requires 224 metabolic reactions to support growth on a glucose-only medium, and 229 for an acetate-only medium [17]. If *E. coli* is fed glucose and lactose together, it will use the glucose first because it takes two less enzymes to use glucose than it does to use lactose [18]. Some commonly metabolized carbohydrates by *E.coli* are dulcitol, salicin, raffinose, and adonitol [19].

Furthermore, statistics [20] show that 1 glucose generates a total of 36-38 ATP in an *E.coli* cell, glycolysis yields about 6-8 ATP, oxidation of pyruvate yields about 6 ATP, Krebs cycle can yield up to 24 ATP, 72,289,000 ATP are needed to make 1 DNA , and 1500 ATP are needed to make 1 protein. One *E.coli* cell has about 55 billion ATP, and consumes about 1.4 billion glucose molecules.

Movement

An *E. coli* moves by propelling itself from place to place by rotating its flagella (picture 1.3). When moving forward, the flagella rotate counterclockwise and the bacteria swims. But when the flagella rotation suddenly changes to clockwise, the bacterium tumbles in place and is temporary incapable of going anywhere, after which it begins swimming in some random direction.



Swimming becomes more intense as the bacterium approaches a chemoattractant, or food. Direction change, on the other hand, is more frequent as the bacterium moves away from the chemoattractant.

Picture 1.3 *E.coli* moving (in courtesy of [46])

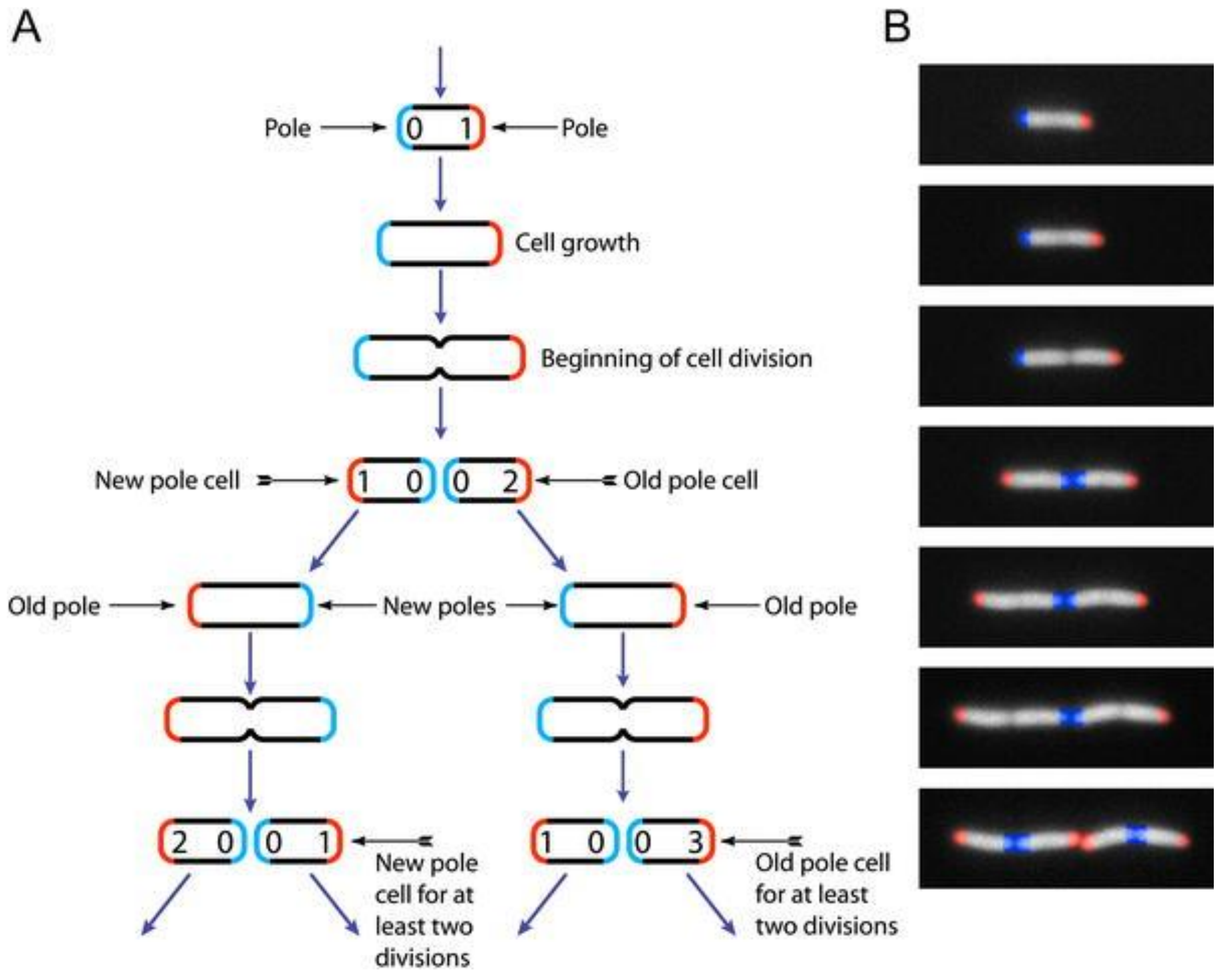
The complex combination of swimming and tumbling keeps the bacteria in areas of higher food concentrations. [21]

Statistics [20] shows that *E. coli* move at 50 um/sec, or at 18 x10-5 km/h. Since the flagella are important in movement, their size is also of importance. The average length of a flagellum is about 10-20 um or ~15,000 nm, and the average diameter of flagella 25 nm.

Reproduction

E.coli reproduces by division roughly every 20 minutes. An important aspect of *E. coli*'s reproduction is aging and symmetry. The primary results in aging have mostly come from organisms that share the traits of a visibly asymmetric division and an identifiable juvenile phase (characterized by the presence of a parent cell/organism that provides for a smaller offspring cell/organism). *E.coli* has been thought of to exhibit functional immortality because it was believed that the bacteria divides symmetrically and has no juvenile phase. Recently, however, Eric J. Stewart, Richard Madden, Gregory Paul, François Taddei [22], as well as Eric Stewart alone [23] proved that no life strategy is immune to the effects of aging because immortality is either too costly or is mechanistically impossible.

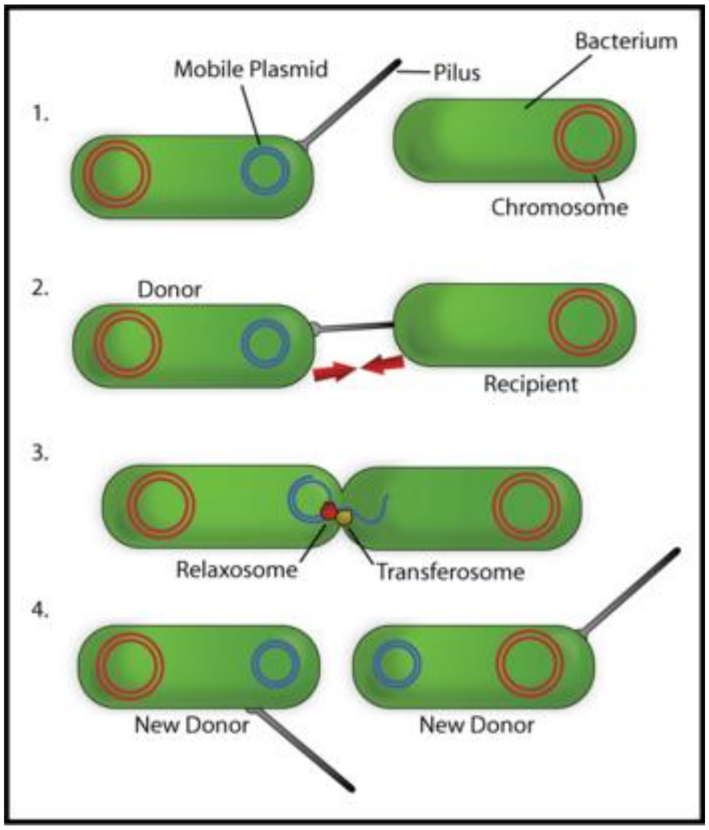
Bacteria usually reproduce by binary fission, which is a process in which the circular DNA molecule replicates, then the cell splits into two identical cells, and each contains an exact copy of the original cell's DNA. *E.coli* grows in the form of a rod, which reproduces by dividing in the middle. As research shows [22][23], each daughter cell inherits an old end or pole of a cell and a new pole of a cell, which is made during the division. The new and the old pole contain slightly different components, and even though they look the same, they are physiologically asymmetrical. At the next division, one cell inherits the old pole again, along with a new pole, while the other cell inherits a not-so-old pole and a new pole (picture 1.4)


A
B
Pole
0
1
Pole
Cell growth
Beginning of cell division
New pole cell
1
0
0
2
Old pole cell
Old pole
New poles
Old pole
2
0
0
1
New pole cell for at least two divisions
1
0
0
3
Old pole cell for at least two divisions

Picture 1.4 *E.coli* division (in courtesy of [40])

Thus, *E.coli*, having no juvenile phase and replicating asymmetrically, does undergo aging. Eric J. Stewart, Richard Madden, Gregory Paul, and François Taddei showed that if the "old" and "new" cells are separated, there are major differences between the two groups: the old pole is a significant marker for multiple phenotypes associated with aging, namely, decreased metabolic efficiency (reduced growth rate), reduced offspring biomass production, and an increased chance of death.

In addition, *E. coli* possess the ability to transfer DNA via bacterial conjugation, which is a process that allows a new mutation to spread through an existing population by the transfer of genetic information from a donor cell to a recipient [24]. In conjugation, a mating bridge is formed between the two new cells, and genetic material is exchanged between the cells. The transfer is usually a part from the bacterial chromosome of an Hfr donor to an F- recipient, or an F plasmid from an F+ donor to an F- recipient [25] (picture 1.5).

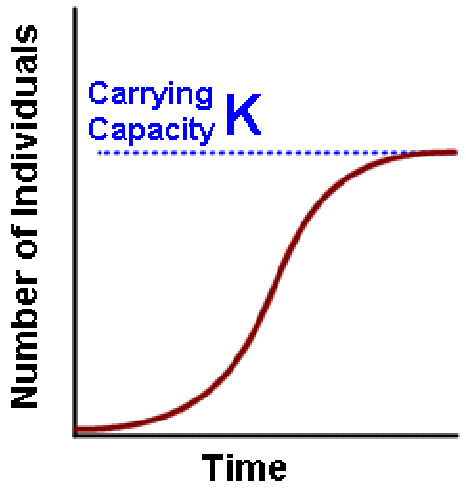


Picture. 1.5 Conjugation in *E.coli*

Schematic drawing of bacterial conjugation. 1- Donor cell produces pilus. 2- Pilus attaches to recipient cell, brings the two cells together. 3- The mobile plasmid is nicked and a single strand of DNA is then transferred to the recipent cell. 4- Both cells recircularize their plasmids, synthesize second strands, and reproduce pili; both cells are now viable donors (in courtesy of [41])

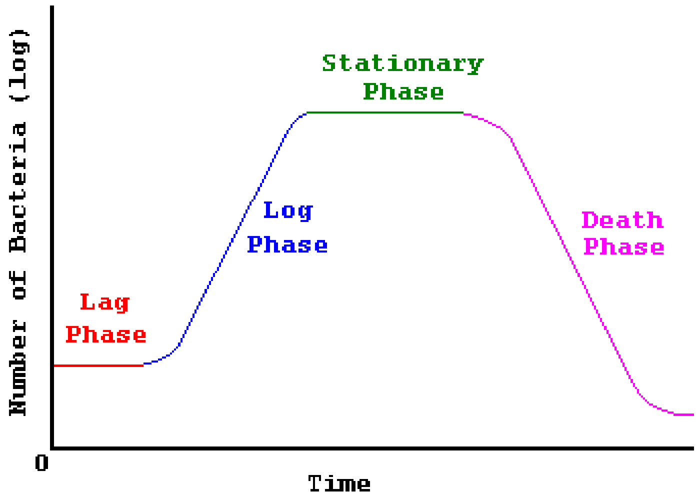
Furthermore, *E.coli*'s growth is often graphed. In a perfect world, *E.coli* would obey the logistic growth model (often formed by laboratory populations), an S-shaped curve that describes the bacteria's growth in a constant environment which always has a constant amount of resources, capable of supporting a definite number of organisms (picture 1.6)

Initially, the bacteria have enough resources to grow exponentially, but the population growth slows as the population approaches the carrying capacity (the maximum population size that can be supported) of the environment.



Picture 1.6 Logistic Growth curve (in courtesy of [42])

In reality, however, resources are neither unlimited, nor constant. Thus, a more realistic growth curve of *E.coli* is considered to be the bell-shaped one (picture 1.7). As in the logistic growth model, after a short lag phase (a period of slow microbial growth that occurs following inoculation of the culture media [26]), the bacteria grow exponentially, then they reach the maximum of the environments' resources, and they start decaying, because the resources get consumed and there are not enough nutrients for organisms to grow (death phase).



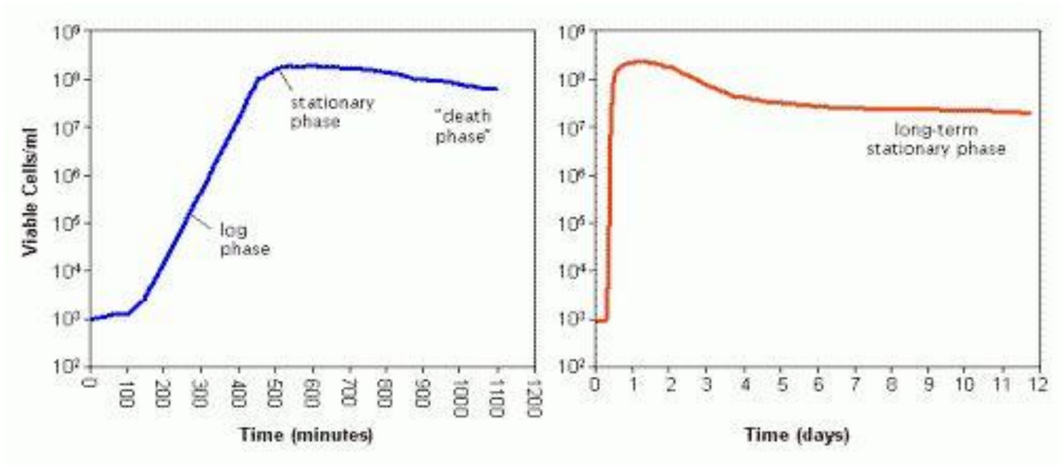
Picture 1.7 Bacteria Growth curve (in courtesy of [43])

Hostile Environment

E.coli live in a variety of environments. Since all resources on our planet are scarce, especially on low-distributed areas, the bacteria have to react to the different factors accordingly. Studies have shown that evolution of microbial diversity occurs during prolonged starvation [27] and *Escherichia coli* expresses new genes in different hostile environments (such as the expression of RpoS regulon which is crucial for survival in liquid cultures during stationary phase [28]). The stationary phase in the growth curve of a colony growing in other than standard conditions represents the bacterial growth in a hostile environment. Thus, when looking at the growth of bacteria in the stationary phase, one is observing the effects of the hostile environment on the bacteria's growth.

E.coli has seven RNA polymerase σ-factors (one of which is the RpoS), which compete for association with the core polymerase subunit, each coordinating the transcription of an important set of genes, which allows control of adaptation to different physiological conditions [29]. It is these genes that are expressed in stressful environments to help the bacteria survive in the new stressful conditions. More than a hundred genes are reported to belong to the RpoS regulon whose initiation results in physiological and morphological changes in the bacteria and increase the bacteria's resistance to various stresses such as heat shock, oxidative shock, cold shock, acid shock, etc. [30] Several studies show that the fitness of *rpoS* mutants depends on environmental conditions [31, 32]. Claude Saint-Ruf, François Taddei, and Ivan Matic's research on *Escherichia coli*'s survival in hostile environments gives significant results supporting the idea of new gene expression in hostile conditions. In nature, *E.coli* is found in structured environments, such as micro colonies, aggregates, and biofilms, and this group's study tested how a structured environment influences the survival of an *E. coli rpoS* mutant to determine whether the selection of *rpoS* alleles can occur in such environments. Apart from their results indicating the importance of *rpoS* gene, their general conclusion that new genes are in fact expressed to accommodate the organisms to the new environment is much more significant. Their results show an increase in the bacteria's expression of *RpoS*-independent genes involved in stress resistance and DNA repair, as well as a global increase of the expression of genes coding for ribosomal proteins. Despite indicating an increase in transcription, protein, and DNA synthesis in aging *rpoS* colonies, the results also showed that RpoS-independent stress responses may be sufficient to ensure the survival of an *rpoS* population in the aging colonies. Furthermore, their results show that the new gene expressions facilitate access to resources because *rpoS* bacteria use the available nutrients more efficiently as there is an increase in their transcriptional modifications of energy-producing pathways. A compensatory transcription in *rpoS* colonies was suggested by their results because there was an increase of the expression of genes involved in energy preservation in *rpoS* colonies. In other words, when *E.coli* are put into a hostile environment where there isn't enough food available, the bacteria react by preserving their energy better through new gene expression. The group's results conclusively showed that unless the bacteria are provided with a sufficient environmental niche, new genes will be expressed to compensate for the hostile environment and help the bacteria accommodate to it. Jonathan E. Visick performed experiments of *E.coli* growth and his results also indicated that when food is limited, the bacteria reduce their metabolic rate and activate a variety of genes to enable them to survive nutrient limitation and cope with stresses such as heat and oxidation that they might encounter before the return of nutrients [5].

Soon after they reach the environment's carrying capacity, the bacteria begin to decrease, marking what was once referred to as "death phase". More recently, however, Jonathan E. Visick's experiments renamed that state as "long-term stationary phase". His predictions based on his results are that the cell number will eventually stabilize again and remain stable indefinitely (picture 1.8). As the graph shows, however, the number of cells changes a little, as some cells die, others divide and the better equipped mutants for survival are selected, and there is an overall slight decrease of growth. Although the slope of the long-term stationary phase has a very small numerical value, it is still present. Visick notes that during this phase, the bacteria's metabolism is very limited, and cells can survive only if they can maintain their macromolecules, and proteins, in a functional state. Oxidative damage and other environmental stresses are major causes of cell death, and stress protection is critical, which is why the genes for preservation of energy and protection play a crucial role in the bacteria's growth.



Picture 1.8 Jonathan E. Visick's experimental results of E.coli growth (in courtesy of [5])

In another study [33], *E.coli* was again tested under nutrient deprivation. The group notes that prolonged starvation is a condition in which microbes (such as *E.coli*) undergo rapid evolution by natural selection. Mutants with an increased fitness, termed the growth advantage in stationary phase (GASP) phenotype, grow and displace their wild-type parents as the majority. Interestingly, GASP mutation, which is continuous through the stationary period, has been identified as an allele of *rpoS*. This study's results indicated that the bacteria were physically more fit in the hostile environment, since they could win in the competition with their wild-type parents. This means that the bacteria that were able to survive by expressing new genes to accommodate to the hostile environment could pass on their genes to the next generation, resulting in a final mutant colony. All of these results indicate the importance of new gene expression under stressful conditions and their physiological change of the fitness of the organisms. Additionally, *E.coli* has been studied in hostile environments with different gravity levels. Studies show that unicellular organisms proliferate faster when cultured under microgravity in orbit, and slower when cultured under hypergravity [49]. Another study found that pressures ranging from 100 to 500 atm retard the growth and reproduction of *Escherichia coli* in nutrient medium [51]. Cultures of suspended bacteria exhibit increased growth in the spaceflight environment [34]. The cytoskeleton plays a significant role when cells are under higher gravity levels. It is a dynamic structure that maintains the shape of the cell, protects the cell, enables cell motion (via flagella and cilia), and plays important roles in both intra-cellular transport and cellular division [35]. The cytoskeleton feels the effects of gravity and reacts to it in unexpected ways [36]. Donald Ingber's research indicates that tugging on integrins (proteins attached to the cytoskeleton) not only cause the cytoskeleton to stiffen, but it also activates certain genes to generate RNA and proteins, which in turn signal the cell to take action. Thus, tickling the cytoskeleton can make cells switch between different genetic programs. This research has already led to a prospective cancer treatment based on changes in cell shape, and it could provide new treatments for osteoporosis, cardiac disease, lung problems and developmental abnormalities, notes Ingber.

Natural Selection

Picture 2.0
Charles Darwin
(in courtesy of [45])

The term “natural selection” was introduced by Charles Darwin (picture 2.0) in his book *The Origin of Species*. Natural selection explains why organisms match their environments so well. It is the process by which individual organisms with favorable traits are more likely to survive and reproduce than those with unfavorable traits. Among the genetic variability, the genotypes associated with the favored traits will increase in frequency in the next generation. After enough time, the process results in adaptations.

Darwin's definition of the term was "principle by which each slight variation of a trait, if useful, is preserved." The concept is powerful because individuals best adapted to their environments are more likely to survive and reproduce. As long as there is some variation between them, there will be an inevitable selection of individuals with the most advantageous variations. For Darwin, natural selection was synonymous with evolution by natural selection.
Although the process of selection seems to be a simple concept, where fitness differences between phenotypes compete, the interplay of the actual selection mechanism with the underlying genetics makes the theory powerful.

If a trait is heritable, natural selection will select the frequencies of the better fitted alleles. Selection can be divided into three classes, on the basis of their effect on the allele frequencies:

- Positive or directional selection occurs when a certain allele has a greater fitness than others, resulting in an increase in frequency of that allele until it is fixed and the entire population expresses the fitter phenotype.
- In purifying or stabilizing selection genetic diversity decreases as the population stabilizes on a particular trait value (unit of selection includes genes, cells, individuals, etc.)
- In balancing selection alleles are maintained at intermediate frequencies in a population.

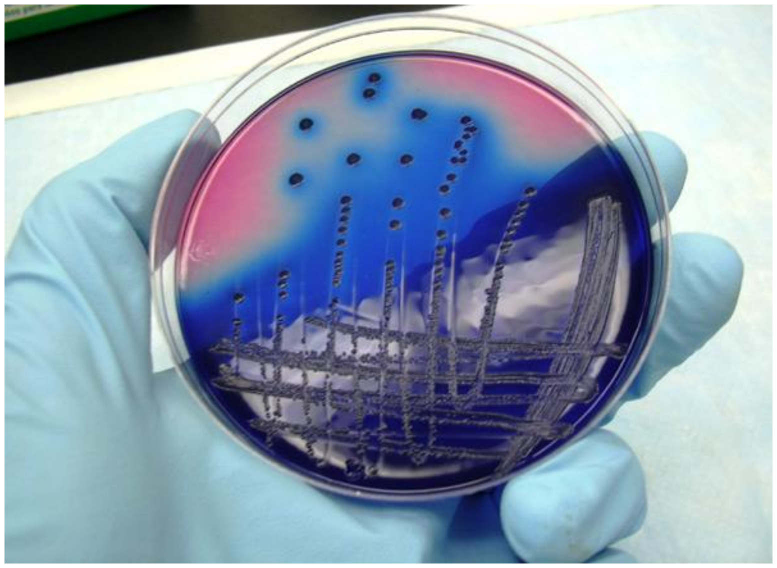
In our project, the type of selection that occurs is positive or directional selection. In general, natural selection acts on the phenotype of organisms. The phenotype results from the individual's genetic make-up, the environment, and the interactions between genes and between genes and the environment.

Fitness is a very important concept in this theory. Natural selection acts on individuals, but its average effect on all individuals with a particular genotype is the fitness of that genotype. Fitness is measured as the proportion of progeny that survives, multiplied by the average fecundity (the ability to reproduce), and it is equivalent to the reproductive success of a genotype. A fitness value of greater than one indicates that the frequency of that genotype in the population increases, while a value of less than one indicates that it decreases. The relative fitness of a genotype is estimated as the proportion of the fitness of a reference genotype. Related to relative fitness is the selection coefficient, which is the difference between the relative fitness of two genotypes. The larger the selection coefficient, the stronger natural selection will act against the genotype with the lowest fitness.

The concept of natural selection is crucial in today’s biology, and it is very important in our project, because it shapes evolution.

The Petri dish

A Petri-dish is a shallow glass or plastic cylindrical dish that biologists use to culture microbes (picture 1.9). When well covered, a Petri-dish keeps the conditions constant. Biologists usually culture the colony in the Petri-dish in a sterile medium - either liquid (broth) or in the form of nutrient agar plates. In our model, we create a virtual Petri-dish where the agents are sheltered. We can control the medium's concentration and distribution, as well as the initial number of cultured bacteria. All other conditions are kept constant.



Picture 1.9 Petri-dish

(in courtesy of [44])

Evolutionary Agent-Based Model

The definition of the Agent-Based model technique is “a specific individual based computational model for computer simulation extensively related to the theme in complex systems, emergence, Monte Carlo Method, computational sociology, multi agent systems, and evolutionary programming” [37]. Agent based modeling can result in complex and interesting behavior through the use of simple rules followed by agents. The three main ideas that are central to all agent based models are:

1. social agents – objects that interact with each other
2. emergence – the development of complex organized systems
3. complexity – exhibit of variation without randomness

The models consist of dynamically interacting agents, or objects, that function by following simple rules performed at given situations. The systems within which they interact can create complexity as the one we see in the real world. All agents

- have specific characteristics
- they are purposeful
- they are situated in space and time
- they reside in networks and on lattice like neighborhoods.

The agents’ responsive and purposeful behavior is encoded in an algorithmic form in the computer program. The modeling process is best described as inductive because the modeler makes the assumptions most relevant to the current situation and then watches the new phenomena emerge from the agents' interactions. Sometimes the result is equilibrium, in other times it is an emergent pattern, and yet in other cases it could be an unintelligible selection.

The idea of agent-based modeling is that a process returns to equilibrium after it is disturbed, or in other words - the system adapts to internal and external stresses so as to maintain functionalities. The task of controlling that complexity requires consideration of the agents themselves - their diversity, connectedness, and level of interactions.

In our project we create an agent-based model to explore the effects of different hostile environments on bacteria *E.coli*. Although we do not follow the typical form of it, our model is conceptually closest to evolutionary agent-based programming. Our agents represent the bacteria *E.coli* and all agents have certain traits that define them. The agents then follow simple rules intended to mimic actual cell interaction in a colony. Below is a brief description of what the agents are and how they interact.

- Each agent has the same type of characteristics. We use the absolute minimum of range of characteristics that at the same support the bacteria's complex behavior and make our model simple.

- Each *E.coli* has the following characteristics

  - X and Y coordinates of the agent

  - Number of flagella (normally between 5 and 20)
  - Average length of flagella

    - Speed of eating - how much food per move the agent consumes
    - Maximum speed - the maxim jump that the agent can make each step. It dependents on the number of flagella; please note that step in our model means time, whereas jump means distance (jump/step is the speed)
    - Energy - measured in ATP between 1.65M - 3M scaled ATP
    - Number of steps after the last division
    - Energy status - the number of steps since the agent last consumed energy. If the energy status is more than the preservation steps the agent goes into preservation
    - Preservation – the rate at which the agent preserve it self. For example if preservation is set to 0.1 the agent will lose only ten percent of the energy instead of a hundred
    - Preservation steps - the number of steps the agent needs to make without food before it goes into preservation

- The characteristics of bacteria are the same type but they differ from each other by deviating from the average values to represent the genetic diversity in a colony
- The agents' steps correspond to real-world time with a certain mathematical relationship
- The agents are located on a virtual lattice (virtual Petri-dish) that is large enough to contain them and can be set to contain different distributions and percentages of food (energy)

- Each agent moves by choosing a direction and taking a step toward it; two counters records the steps taken – total steps before dividing and steps without food (preservation counter), so that the agent can react accordingly if the number of steps indicates the agent needs to change its behavior

- Movement is random (each agent chooses a random direction and a random angle)
- Each agent moves with a speed whose value has a deviation from the standard value of *E.coli*'s speed in our model, and is mathematically calculated by the number of flagella, length of flagella, and the cell's speed
- Each agent consumes energy, if such is present, at each step; each "food" consumed holds a different numerical value that is added to the energy of each consumer agent
- If an agent goes one step without food, the counter for steps before preservation adds +1, and if it takes a step with food, it sets the counter to 0
- Each agent needs to be over a set threshold of energy to exist; if it is under this threshold it dies
- In order for an agent to "reproduce", it should have equal or more energy than a given threshold (usually set to a minimum of twice its original energy) and it should pass a certain threshold of steps taken (corresponds to E.coli's preparation before division). When this energy and number of steps are reached, the agent divides. Please note that the agent will NOT divide if it is in a state of preservation.
- When an agent divides, it creates two new agents; one of them is identical to the parent and the other one has all the original's agent characteristics with slight differences (to account for *E.coli*'s aging and its effects on the progeny)
- The new agents follow the same rules and have all the characteristics that define them
- If an agent's counter for steps before preservation has a greater value than the counter for steps before division, then the agent goes into preservation

- If an agent is in a state of preservation, it preserves its energy by subtracting a small set percentage of it each foodless step

We established the model mathematically by fitting our variables in a way to repeat the experimental results of Jonathan E. Visick. We matched the variables until our data repeated Visick's graphs. Once we did this, we had a model that behaves realistically. In our algorithm and model, we incorporated some very important concepts that closely resemble real-life cellular functioning and cell interaction because our model repeated experimental results of such behavior. The agents in our model correctly expressed the genes for preservation of energy, as microbiological research indicates should happen, and competition was present. We then carefully changed the environmental conditions to create different hostile environments and observed how the *E.coli* agents reacted to them. We modeled high gravity levels by decreasing the average speed of the agents simultaniosly with increasung the energy they spend per step, and prolonged starvation by changing the percent of food distributed per site and the percent of the sites where there is a food on the lattice. Data was collected of the bacterial growth over the scaled time, and graphs were composed for comparison and clear data representation.

Technology Used

The hardware we had for running our application was provided by Harvard Medical School. We were allowed to use twenty processors on their Linux based cluster called Orchestra. One of the important decisions our team faced was the choice of software technology that we were going to use in developing the computational model we created. First of all, we needed to decide what programming language to write in. We created a list of the languages that are most commonly used for simulations. The list included C, C++, FORTRAN, and Java. We closely analyze these four languages in order to choose the best one. From the four languages, Java was the worst solution. Even though easy to develop, secure, and scalable, Java code is much slower when compiled (even natively). Even more, Java was going to be impossible to parallelize on the provided cluster (some of the nodes did not have JVM installed). The second language we eliminated was C++. Even though C++ is much more powerful than C and FORTRAN, C++ was adding the unnecessary complexity. We discovered that in our application there was no need of writing classes and instantiating object – the procedure programming was more than enough for us. Even more, code compiled in C++ was slower than the one compiled in C or FORTRAN. After eliminating C++ and Java, we needed to decide between C and FORTRAN. The languages' performance was quite similar, but we chose C over FORTRAN, because of our familiarity with the language. We also needed to parallelize our code – after examining the provided hardware (and installed software on it) from Harvard Medical School, we decided that the Message Passing Interface (MPI) is the best solution for us. After we chose C (with MPI) as our programming language, we needed to decide what Integrated Development Environment (IDE) to use. Even though writing in Notepad (or vi) and compiling on command line can be lots of fun, we wanted to use an environment that will allow us to develop fast as well as debug our application. After close consideration, we decided to use the open source IDE Dev-C++.

The Algorithm & the Code

Our computational model is written in ANSI C using the MPI parallel library. Please see the ***Appendix –B*** - **The Source Code**, and ***Appendix –C*** - **The Main Algorithm flow chart**. Below we provide documentation of all global variables, directives, and methods as well as explanation of the most important formulae and parameters used in the program.

Source Code Explanation

1.Description

The created program simulates the life of *Escherichia coli* in different conditions, the environment as well as the *Escherichia coli* themselves. The conditions of the environment as well as the characteristics of the bacteria could be changed by using defined and precompiled variables (directives).

2. Precompiled variables (directives)

File: SCC_v_1_1.c

HANDLE_ERRORS – when the variable is defined the program will be compiled with error checking. If any of the error checking failed the program will display a message containing:

- Information about the error
- The file the error occurred in

- The line number in that file

After the message is displayed the program will be terminated. Please, note that the message is send to the standard error output. If the variable is not defined the program does not check for errors and segmentation errors could occur.

VERBOSE – when the variable is defined the program will be compiled with messages. The messages will be display the progress of the program. Please note that the messages will be sent to the standard output. Please also note that if **USE_MPI** is defined the messages will be displayed only for the processor with id zero. If **VERBOSE** is not defined no messages are sent to the standard output.

USE_MPI – when the variable is defined the program will be compiled with the Message Passing Interface. **USE_MPI** must be defined only when you are planning to run the program on multi-processor machine (server, cluster, etc.). Please note that by defining the **USE_MPI** you can change the functionality of some methods (the description of the changes will be provided in the methods.

MAX_HOURS – this variable is defined only when the **USE_MPI** is defined. The variable is the maximum number of hours that we can simulate the system.

File: SCC_structs_v_1_1.c

E_DIV – the energy that an e_coli needs to reach, before it is able to divide

E_MIN – the minimal energy that a new e_coli (by new e_coli we mean initially created e_coli, the number of initially e_coli is set in the main methods with the *begin_e_coli* variable) can have.

E_MAX – the maximum energy that a new e_coli (by new e_coli we mean initially created e_coli, the number of initially e_coli is set in the main methods with the *begin_e_coli* variable) can have.

E_GAIN – the variable defines the average energy gain of an e_coli. Note that the actual gain is calculated as a Gaussian around **E_GAIN** and dispersion of the square root of the value of the variable. Please also note that an e_coli cannot eat more than the available food in its current position.

E_LOSS_PER_STEP – the variable defines the average energy loss of an e_coli each step. Note that the actual loss is calculated as a Gaussian around **E_LOSS_PER_STEP** and dispersion of the square root of the value of the variable.

E_CONST_LOSS_PER_STEP – the variable defines how much energy an e_coli will lose each step. Please note that this variable is independent from **E_LOSS_PER_STEP**, the actual loss of energy of the e_coli is the sum of these variables

E_DEATH – the variable defines the minimum energy an e_coli can have. If the energy is below that value the e_coli will die.

E_PER_PLACE – the variable defines the average food in each position of the Petridis that contains food

PLACE_DENSITY – – the variable defines the density of the food in the Petridis

STEP_FOR_DIVISION – the variable defines the number of steps that an e_coli needs before it can breed (divide). Please note that is roughly 20 min

PRESERVATION_OF_ENERGY_STEPS – the variable defines the number of steps that an e_coli can be without food before it goes into preservation

PRESERVATION – the variable defines the average preservation of the e_coli. The preservation shows what percent of the energy the e_coli loses. For example if **PRESERVATION** is set to 0.1 the e_coli will lose only ten percent of the energy instead of a hundred. Please note that **PRESERVATION** should never have a value of more than one.

min_e_coli_speed – the variable defines the minimum e_coli speed.

max_e_coli_speed – the variable defines the maximum e_coli speed.

min_e_coli_flagellas – the variable defines the minimum number of e_coli flagella's

max_e_coli_flagellas – the variable defines the maximum number of e_coli flagella's

min_length_flagella – the variable defines the minimum length of the e_coli's flagella's

max_length_flagella – the variable defines the maximum length of the e_coli's flagella's

max_e_coli_speed_of_eating – the variable is currently not used

min_e_coli_speed_of_eating – the variable is currently not used

eats_ATP_per_sec – the variable is currently not used

max_e_coli_percent_transform – the variable is currently not used

min_e_coli_percent_transform – the variable is currently not used

GRAVITY – the variable defines the gravity of the system and it is used in the moving of the e_coli

MAX_NUMBER_E_COLI – the variable defines the maximum number of e_coli that the Petridis can contain

MAX_AREA_X – the variable defines the maximum value of the x coordinate of the Petridis. Please note that all coordinate are integers.

MAX_AREA_Y – the variable defines the maximum value of the y coordinate of the Petridis. Please note that all coordinate are integers.

MIN_AREA_X – the variable defines the minimum value of the x coordinate of the Petridis. Please note that all coordinate are integers.

MIN_AREA_Y – the variable defines the maximum value of the y coordinate of the Petridis. Please note that all coordinate are integers.

MAX_ANGLES – the variable pre-defines the maximum angle in a circle aka 360 degrees

All other precompiled variables in the program are used for either random number generation (please check the official documentation of Numerical Recipes) or mathematic calculation (variables like PI, etc)

3. Global variables, structures, and arrays

File: SCC_v_1_1.c

count_e_coli_sum[MAX_HOURS] – the variable holds the number of e_coli in the system for every hour of the simulation. Please note that by hour we mean an hour for the e_coli. Please also note that this variable we not be defined if **USE_MPI** is not defined.

count_e_coli_sum_mpi[MAX_HOURS] – the variables is used to sum the number of e_coli for all processors for every hour of the simulation. Please note that by hour we mean an hour for the e_coli. Please also note that this variable we not be defined if **USE_MPI** is not defined.

avg_energy_sum [MAX_HOURS] – the variable holds the average energy of the e_coli in the system for every hour of the simulation. Please note that by hour we mean an hour for the e_coli. Please also note that this variable we not be defined if **USE_MPI** is not defined.

avg_energy_sum_mpi[MAX_HOURS] – the variables is used to sum of the average energy of the e_coli for all processors for every hour of the simulation. Please note that by hour we mean an hour for the e_coli. Please also note that this variable we not be defined if **USE_MPI** is not defined.

death_e_coli_sum [MAX_HOURS] – the variable holds the number of death e_coli in the system for every hour of the simulation. Please note that by hour we mean an hour for the e_coli. Please also note that this variable we not be defined if **USE_MPI** is not defined.

death_e_coli_sum_mpi[MAX_HOURS] – the variables is used to sum the number of death e_coli for all processors for every hour of the simulation. Please note that by hour we mean an hour for the e_coli. Please also note that this variable we not be defined if **USE_MPI** is not defined.

total_energy_sum [MAX_HOURS] – the variable holds total food(expressed in energy units) in the system for every hour of the simulation. Please note that by hour we mean an hour for the e_coli. Please also note that this variable we not be defined if **USE_MPI** is not defined.

total_energy_sum_mpi[MAX_HOURS] – the variables is used to hold total food(expressed in energy units) for all processors for every hour of the simulation. Please note that by hour we mean an hour for the e_coli. Please also note that this variable we not be defined if **USE_MPI** is not defined.

struct _e_coli – the structure is used to define the characteristics of the e_coli

x – the x coordinate of the e_coli. Please note that coordinates can only be integer

y – the y coordinate of the e_coli. Please note that coordinates can only be integer

number_of_flagellas – the number of flagella's of the e_coli

avg_length_flagella - the verage length og the flagella;s

speed_of_eating –how much energy an e_coli can eat for one step

max_speed – the maximum number of s

energy – the current energy of the c_coli

steps - the number of steps after the last division

energy_status – energy status is the number of steps since the e_coli last ate food (energy). If the energy status is more than the **preservationSteps** the e_coli goes into preservation

preservation – the rate at which the e_coli preserve it self. For example if **preservation** is set to 0.1 the e_coli will lose only ten percent of the energy instead of a hundred. Please note that **preservation** should never have a value of more than one.

preservationSteps – the number of steps the e_coli needs to make without food before it goes into preservation

struct _idum - the structure is used to define the seeds for the random number generators

init – used as seed for generating the initial conditions of the e_coli

density – used as seed for generating the density of the food

radius – used as seed for generating the displacement of the e_coli

area – used as seed for generating the food in a single cell of the Petridis

angle – used as seed for generating the angle on which the e_coli moves

idums for initial e_coli (the ones generated in the beginning of the program)

x – used as seed for generating the initial x coordinate of the initial e_coli

y – used as seed for generating the initial y coordinate of the initial e_coli

number_of_flagellas – used as seed for generating the number of flagellas of the initial e_coli

avg_length_flagella – used as seed for generating the average length of the flagellas of the initial e_coli

speed_of_eating – used as seed for generating the maximum speed of eating of the initial e_coli

energy – used as seed for generating the energy of the initial e_coli

preservation – used as seed for generating the preservation rate of the initial e_coli

preservationSteps – used as seed for generating the preservation steps of the initial e_coli

File: SCC_structs_v_1_1.c

STEP_TO_HOURS – the variables is used to convert from steps to hours. Note that 20 min is the breeding time of the e_coli which is in the program defined with **E_DIV**

STEP_TO_HOURS_L – is the integer version of ***STEP_TO_HOURS***

interval_energy – the variables is currently not used

interval_e_colli_speed – the variables is currently not used

interval_flagellas – the variables is currently not used

interval_length_flagellas – the variables is currently not used

File: E_COLI.c

struct __cos_sin – contains information for the cos and sin of a given angle

> ***cos*** – information about the cosine
>
> ***sin*** – information about the sine

_cos_sin[MAX_ANGLES] – contains pre-calculated information about the cosine and sine of all the inter number angles. The array is used to improve the performance of the program

area[MAX_AREA_X][MAX_AREA_Y] – the Petridis is represented by this area. Each field of the array may contain the food (energy)

****coli*** – dynamically allocated array that contains all the e_coli in the Petridis that are alive

count_e_coli – the number of all the e_coli in the Petridis

pres_number – used for testing purposes only, it counts the number of preservations that have occurred

count_death_e_coli – counts the number of death e_coli

avg_e_coli_energy – the average energy of all the e_coli

avg_born – the average rate of born e_coli for the current hour

avg_death – the average rate of deat e_coli for the current hour

total_food – the total food(energy) that is left in the Petridis

All other global variables, structures, and arrays in the program are used for either random number generation (please check the official documentation of Numerical Recipes) or tracking persistency (number of messages displayed)

4. Methods

File: SCC_v_1_1.c

main – the main method is used as an entry point in the program. In it we can set the number of day we want to simulate (by the variable ***number_days***) and the number of initial e_coli (by the variable ***begin_e_coli***). The method is compiled quite differently if **USE_MPI** is defined.

SCC_structs_v_1_1.c

calc_e_coli_max_speed – calculates the maximum speed of the e_coli based on their characteristics (flagella's, average flagella's length, etc)

init_e_coli – initializes the e_coli in the Petridis

e_coli_jump_energy_loss – calculates the loss of energy of an e_coli based on the e_coli characteristics and the length of the jump (in steps)

calc_e_coli_status – this method check the current energy of the e_coli. Te method returns 1 if the e_coli will die, returns 2 if the e-coli can reproduce, and returns 0 if the e_coli is regular

E_COLI.c

create_initial – creates the initial e_coli

init_area – initializes the Petridis and the food in it

move_e_coli – moves the last e_coli into the place of an already death e_coli into the dynamically allocated array ***coli***

make_new_e_coli – makes a new e_coli based on its parent one

eat – makes the e_coli to eat, if the current location of the Petridis contains any food

move_single – moves a single e_coli into the Petridis. The method also implement the boundary conditions of the Petridis.

move_all – this method takes care of the e_coli for one step. The method makes sure that all ecoli that are eligible move, eat, die, or go in preservation.

File: GENERAL.c

stop_pr – this method is used to tell thee operating system the pause the current execution of the program until the user presses a key

Error – this method will be compiled only if the precompiled variable **HANDLE_ERRORS** is defined. The method is used to display error to the standard error output

init_process – this method will be compiled only if the precompiled variable **VERBOSE** is defined. The method is used to write to the standard output when another method is entered as well as count the number of entries.

init_process – this method will be compiled only if the precompiled variable **VERBOSE** is defined. The method is used to write to the standard output when another method is entered as well as count the number of entries.

complete_process – this method will be compiled only if the precompiled variable **VERBOSE** is defined. The method is used to write to the standard output when another method is exited as well as count the number of exits

arb_gasdev_l **–** this method is used to give an integer number that is part of a Gaussian with given average value, dispersion, and seed that could be supplied to Numerical Recipe's gasdev

arb_gasdev_d – this method is used to give a real number that is part of a Gaussian with given average value, dispersion, and seed that could be supplied to Numerical Recipe's gasdev

init_random_numbers – this method initializes all the random number seeds. The method is highly affected if **USE_MPI** is defined

print_e_coli – this method prints all the information of one e_coli to the standard output

All other methods in the program are used for random number generation (please check the official documentation of Numerical Recipes)

Parameters and formulae

Creating a new e_coli (only when the program starts):

number_of_flagellas:

Gaussian with average max_e_coli_flagellas and standard deviation sqrt(max_e_coli_flagellas). The number_of_flagellas cannot be more than 2 * max_e_coli_flagellas and less than zero.

avg_length_flagella

Gaussian with average max_length_flagella and standard deviation sqrt(max_length_flagella). The avg_length_flagella cannot be more than 2 * max_length_flagella and less than zero.

speed_of_eating

Gaussian with average max_e_coli_speed_of_eating and standard deviation sqrt(max_e_coli_speed_of_eating). The speed_of_eating cannot be more than 2 * max_e_coli_speed_of_eating and less than zero.

max_speed

The max_speed is more than zero and it is calculated with the following formula:

1 / GRAVITY (min_e_coli_speed + (int)((e.number_of_flagellas - min_e_coli_flagellas)/interval_flagellas*(e.avg_length_flagella-min_length_flagella)/interval_length_flagellas*(double)interval_e_colli_speed))

energy

Gaussian with average E_MAX and standard deviation sqrt(E_MAX). The energy cannot be more than 2 * E_MAX - E_MIN and less than E_MIN.

preservation

Gaussian with average PRESERVATION and standard deviation sqrt(PRESERVATION). The preservation cannot be more than 2 * PRESERVATION and less than zero.

preservationSteps

Gaussian with average PRESERVATION_OF_ENERGY_STEPS and standard deviation sqrt(PRESERVATION_OF_ENERGY_STEPS). The preservationSteps cannot be more than 2 * PRESERVATION_OF_ENERGY_STEPS and less than zero.

x

Uniformly distributed number between MIN_AREA_X and MAX_AREA_X

y

Uniformly distributed number between MIN_AREA_Y and MAX_AREA_Y

Initializing the lattice:

area[][](contains the amount of ATP)

Gaussian with average E_PER_PLACE and standard deviation sqrt(E_PER_PLACE)

Dividing of e_coli:

number_of_flagellas:

Gaussian with average the number_of_flagellas of the mother cell and standard deviation square root of this average. Again, the number_of_flagellas cannot be more than 2 * max_e_coli_flagellas and less than zero.

avg_length_flagella

Gaussian with average the avg_length_flagella of the mother cell and standard deviation square root of this average. Again, the avg_length_flagella cannot be more than 2 * max_length_flagella and less than zero.

speed_of_eating

Gaussian with average the speed_of_eating of the mother cell and standard deviation square root of this average. Again, the speed_of_eating cannot be more than 2 * max_e_coli_speed_of_eating and less than zero.

max_speed

The max_speed is more than zero and it is calculated with the following formula:

min_e_coli_speed + (int)((e.number_of_flagellas - min_e_coli_flagellas)/interval_flagellas*(e.avg_length_flagella-min_length_flagella)/interval_length_flagellas*(double)interval_e_colli_speed)

energy

The energy of the new cell is half of the energy of the mother cell.

preservation

Gaussian with average the preservation of the mother cell and standard deviation square root of this average. The preservation cannot be more than 2 * PRESERVATION and less than zero.

preservationSteps

Gaussian with average the preservationSteps of the mother cell and standard deviation square root of this average. The preservationSteps cannot be more than 2 * PRESERVATION_OF_ENERGY_STEPS and less than zero.

x

The x position of the new cell is the same as the x position of the mother cell.

y

The y position of the new cell is the same as the y position of the mother cell.

Eating of e_coli:

If there is enough food the e_coli will eat E_GAIN, if there is not enough food the e_coli will eat the food that is left in the place

Moving of e_coli:

```
fi = (int)(360 * ran2(idum_angle) );
x = (int) (_cos_sin[fi].cos * r);
y = (int) (_cos_sin[fi].sin * r);
```

There are limited boundary conditions – if the e_coli tries to go out of the Petridis it will be stopped, or:

MIN_AREA_X < x < MAX_AREA_X

MIN_AREA_Y < y < MAX_AREA_Y

The loss of energy is calculated by the formula:

GRAVITY*(E_CONST_LOSS_PER_STEP + E_LOSS_PER_STEP * jump)

Results

We ran our model to reach 1.5 x 10^7 bacteria, on a 10^4 x 10^4 lattice. Each simulation lasted about 10-12 hours. Our final results are averge values from over 40 simulations we did by using 10 CPU's at the Orchestra Linux cluster at Harvard Medical School. We ran these simulations twice with two different gravity conditions. We monitored the evolution in time of the the average cell's growthe rate, length and number of flagella, time for expression of the preservation gene, and the average bacteria speed. Our results can be classified in six (I-VI) different groups:

I. We have an accurate graph of the cell's growth rate and the food changes in time, for the two different gravity levels. In normal gravity, our results closely resemble Jonathan Visick's experimental results. De facto, we put most of our efforts to establish the parameters of our model so that we can replicate Visick's experiment. This gave us the opportunity to compare the computer steps in our program to the real time, and to set up the time-scale in the model. In our model, 72 steps of the main loop correspond to 20 minutes real time. Also, we managed to scale our parameters so that the program is realistic and close to Visick's results. Fig 1. clearly represent lag phase, log phase, and long-term-stationary phase. The special hump on each of the growth graphs indicates the preservation gene expression. The condition of hypergravity is also on the same graph. The conditions are created by making the jump energy proportional to the gravity, and the speed of bacteria inversely proportional to the gravity. As we can expect, we have slowed processes and growth rate in hypergravity. When growth rate is expressed in percentages, however, we also notice a differene between the shape of the normal gravity curve and the hypergravity curve, which is due to the change of the genetic characteristics.

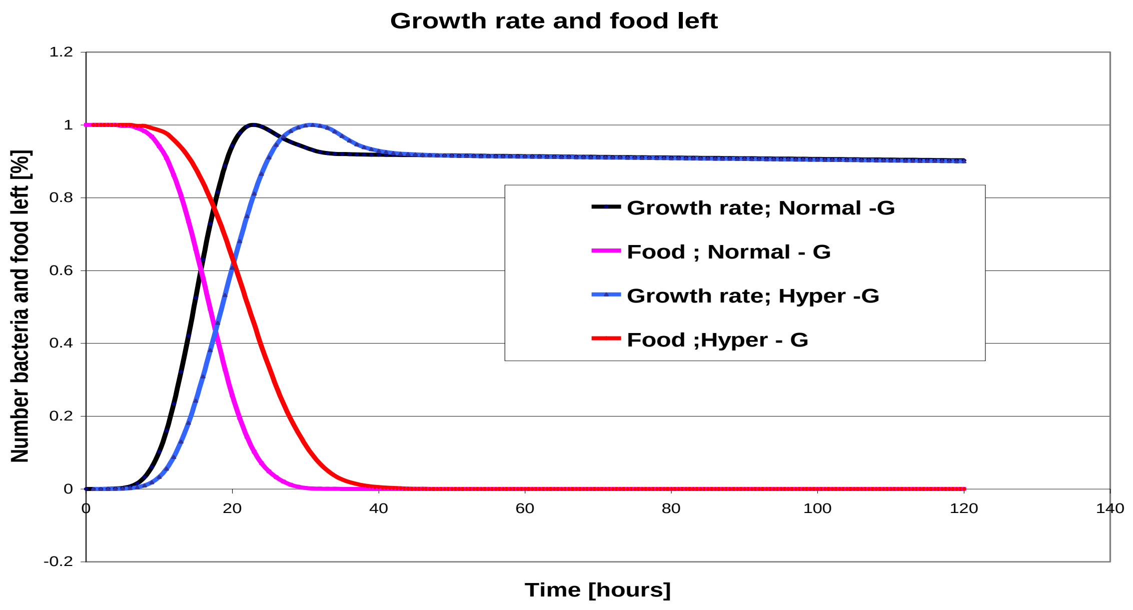


Fig 1. *This graph represents the average growth rate of the bacteria and food level for normal and hypergravity conditions. The number of bacteria and the amount of food left are expressed in percentages to focus on the different shapes of the curves in the two gravity conditions. The black and blue curves represent the growth rate in normal (G=1) and hypergravity (G=1.5) conditions, respectively; the pink and red curves represent the food level in normal and hyper gravity conditions, respectively.*

II. We monitored the average number of births and deaths in the colony and ploted the results on Fig 2. In both gravity conditions, the maximum death rate does not correspond to the exhaustion of the food resources, because of the preservation gene. This is logical, because the bacteria die with the highest rate right before the food is exhausted, but when the food is actually over, they have already turned on their preservation genes and the death rate has started to decline.

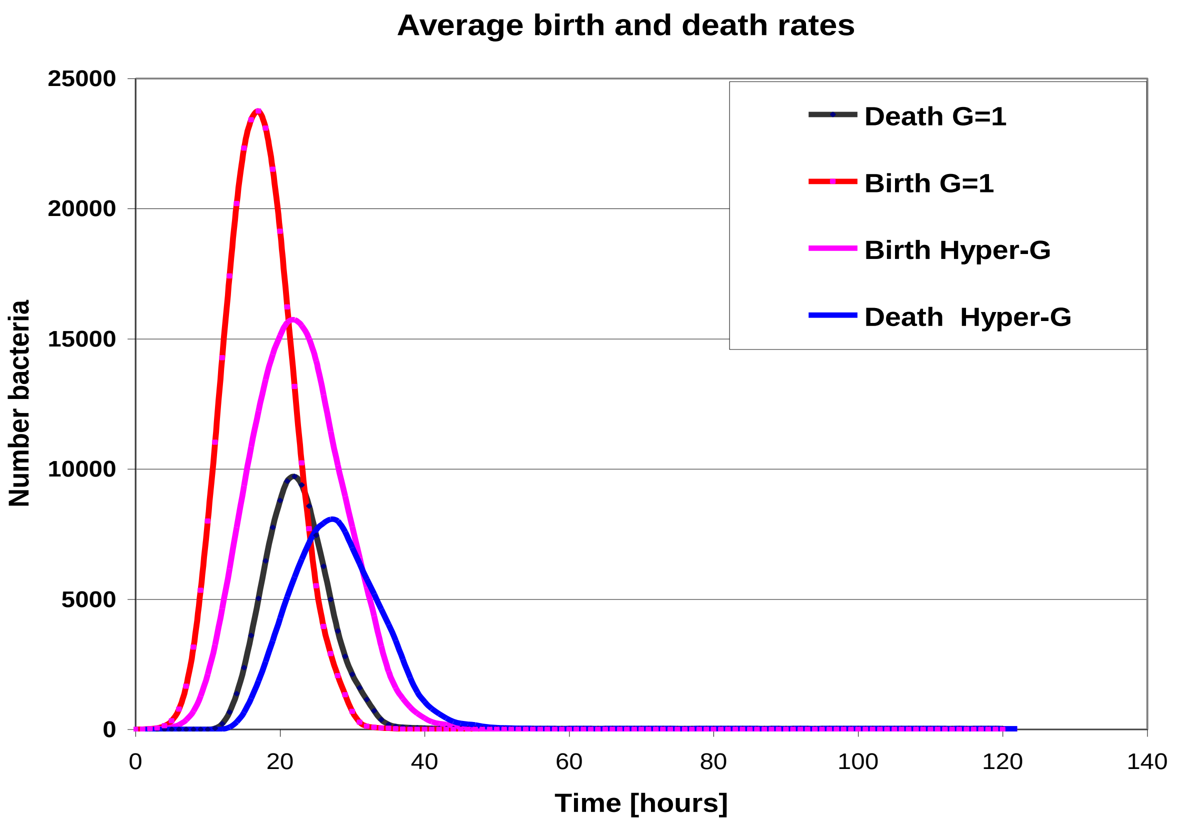


Fig 2. *This graph illustrates the average number of births and deaths in time in normal and hyper gravity condition. The red and black curves represent the birth and death rates, respectively, in normal gravity condition (G=1); the pink and blue curves represent the birth and death rates, respectively, in hyper gravity condition (G=1.5)*

III. This group represents the evolution of the gene characteristics in the two gravity conditions, averaged by the population size. Fig 3. represents the change in the average speed /number of flagela and their length/ with which the bacteria move, which is inherited genetically by Gaussian distribution to each generation. The graph indicates that while there is food, the bacteria with greater speed survive. When the food decreases, however, the average maximum speed is decreased, this decreases the energy spent by each bacteria. When the food is exhausted, a constant average speed is established in the generation. In hypergravity the behavior of the curve is similar, but the average speed is smaller.

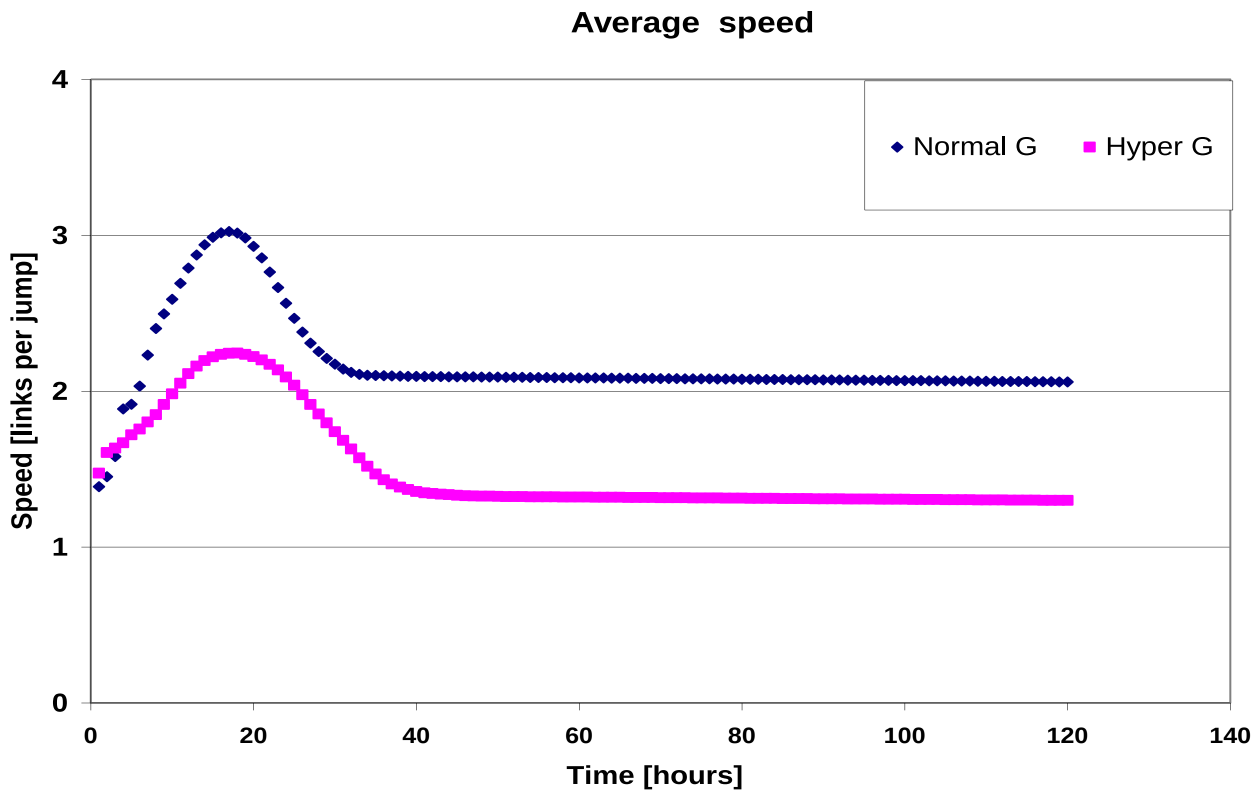


Fig 3. *This graph illustrates the average speed of a bacterium in time.The blue curve represents the average change of speed of a bacterium in normal gravity conditions (G=1), and the pink one represents the average change of speed of a bacterium in hypergravity conditions (G=1.5)*

IV. Fig 4. expresses the time and the conditions in which the gene responsible for preservation is expressed. The graph shows our results that in hypergravity, on average, the preservation gene is expressed earlier. This is logical because the bacteria in hypergravity are slower and they find food with less probability. Thus, only the ones that express the gene earlier survive to pass on their genes.

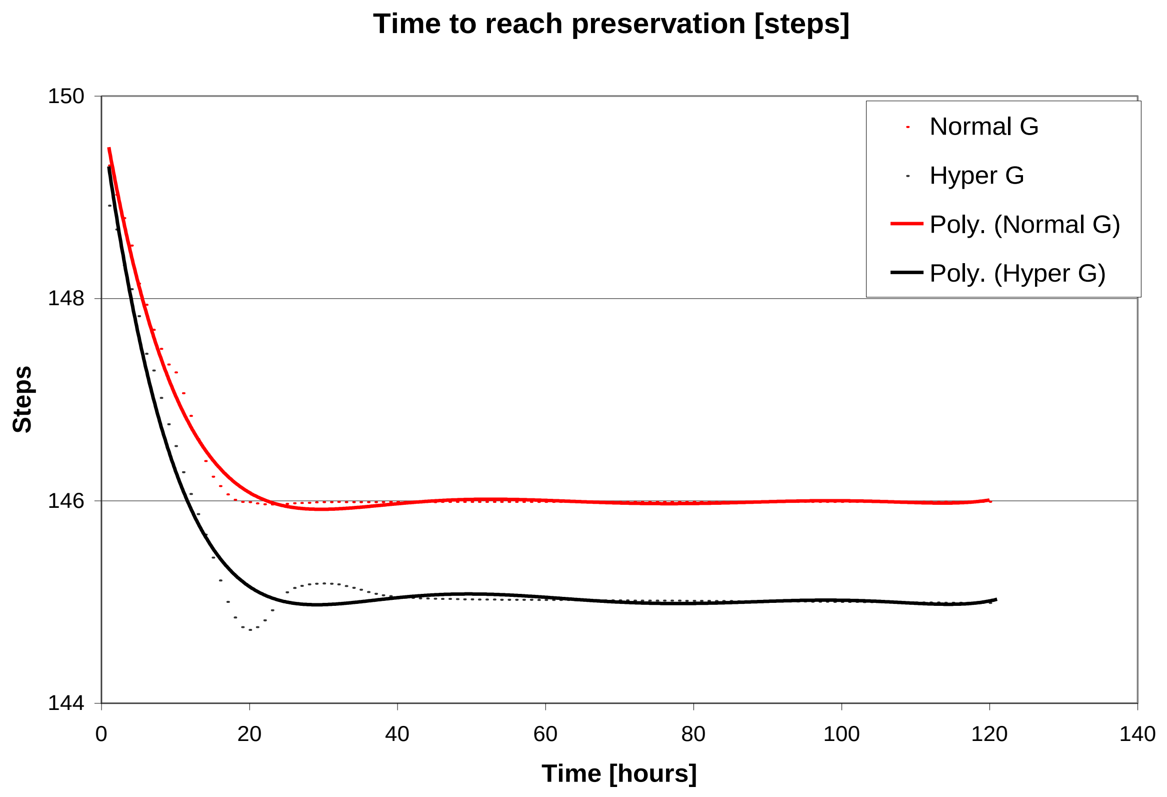
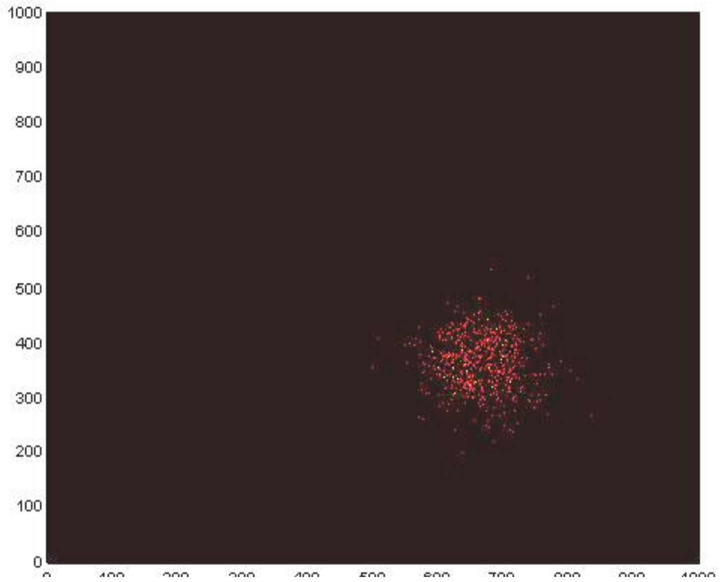


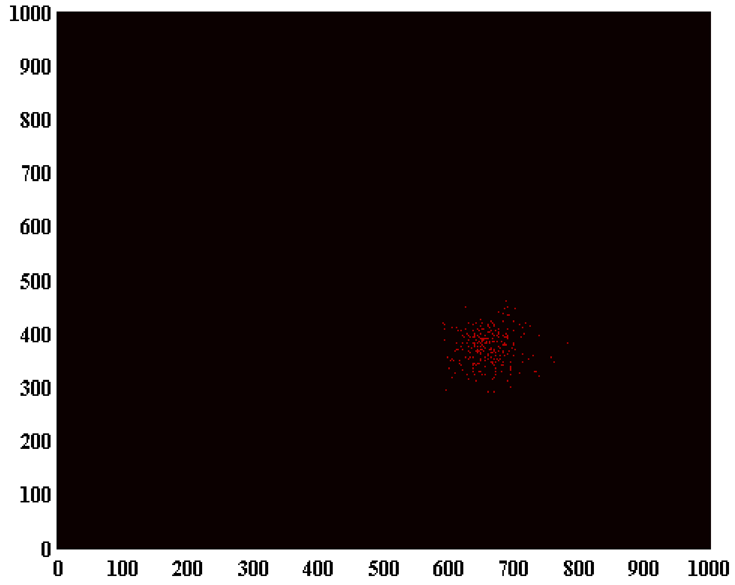
Fig 4. *This graph represents the average number of steps required for a bacterium to enter preservation in hyper and normal gravity conditions. The red polynomial curve, included for clearer illustration, represents average number of steps required for a bacterium to enter preservation in normal gravity conditions (G=1), and the black one is for average number of steps required for a bacterium to enter preservation in hyper gravity conditions (G=1.5).*

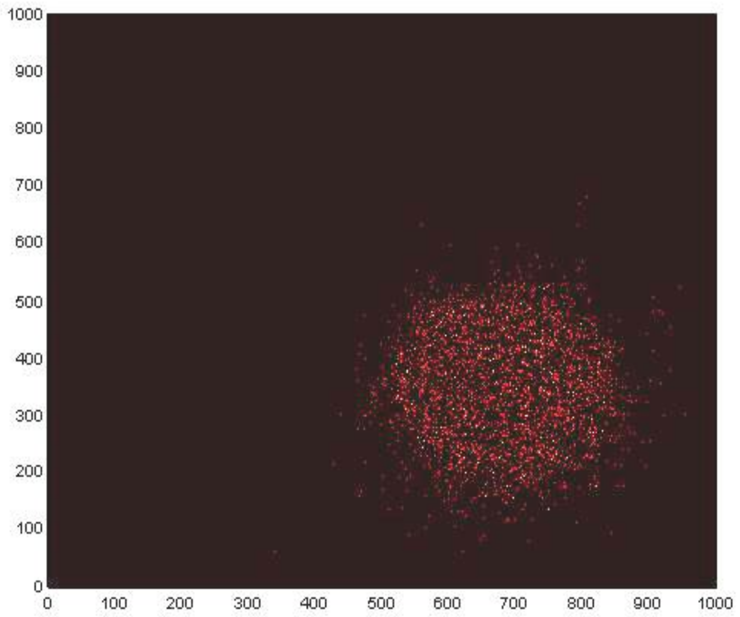
V. We created on MatLab a series of 3-D slides that show the time evolution of the bacteria on every four hours for a total of 120 hours. We can clearly see the diffusion character of the motion. Also, we can clearly see wave-like motion of the colony due to the fact that the food in the center of the lattice, from where the bacteria started, is exhausted faster, and they start dying with a faster frequency on this part of the Petri-dish. We call this the “Wave Effect”. The last slide on the 120th hour shows equally distributed bacteria on the lattice. (see Fig.5; slieds are read vertically for the two different gravity levels ; 1st slide set – 4th hour, 2nd slide set – 8th hour, 3rd slide set – 12th hour, 4th slide set – 16th hour, 5th hour set – 20th hour, 6th slide set – 24th hour, 7th slide set – 120th hour)

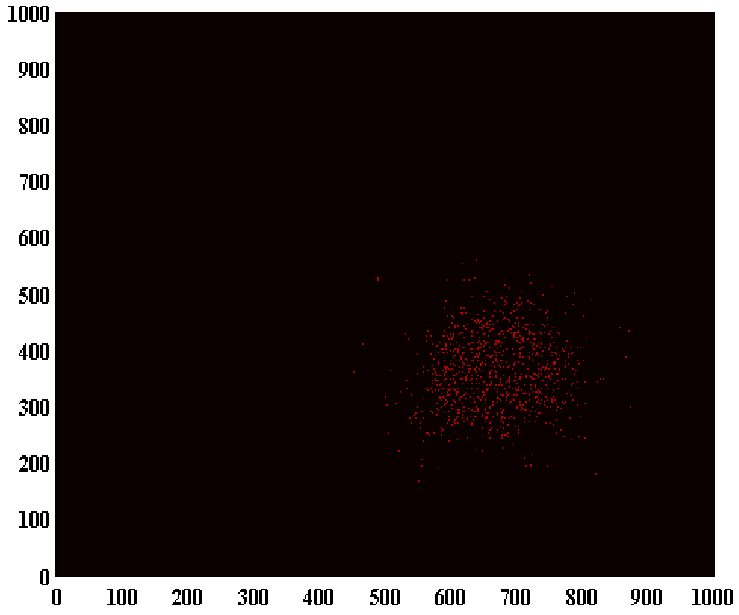
Normal Conditions

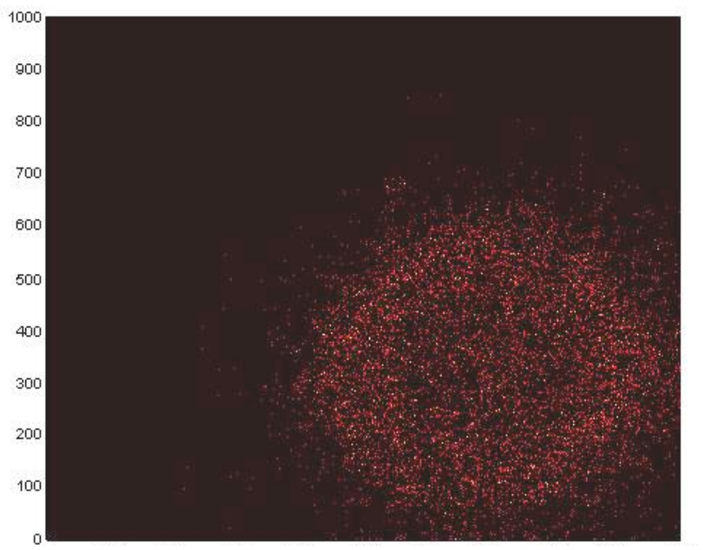
Hypergravity Conditions

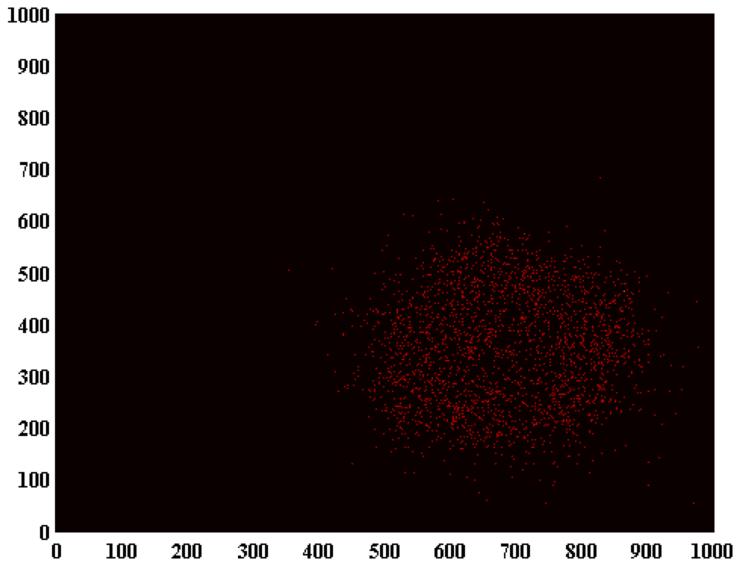


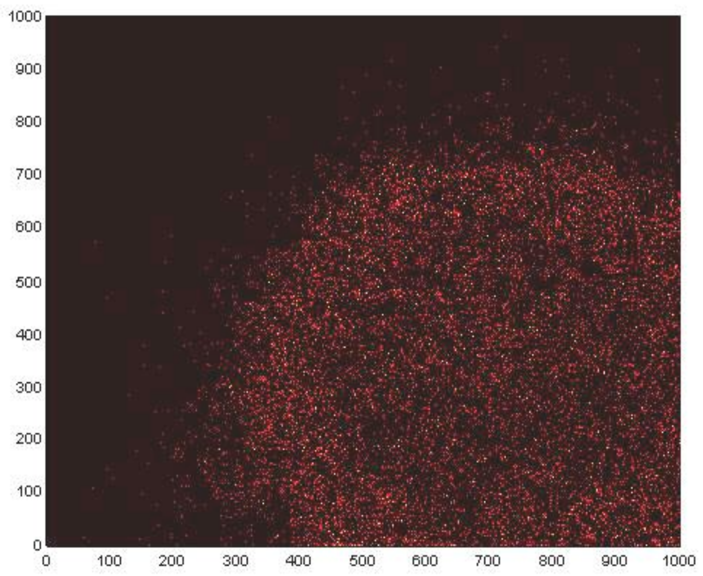


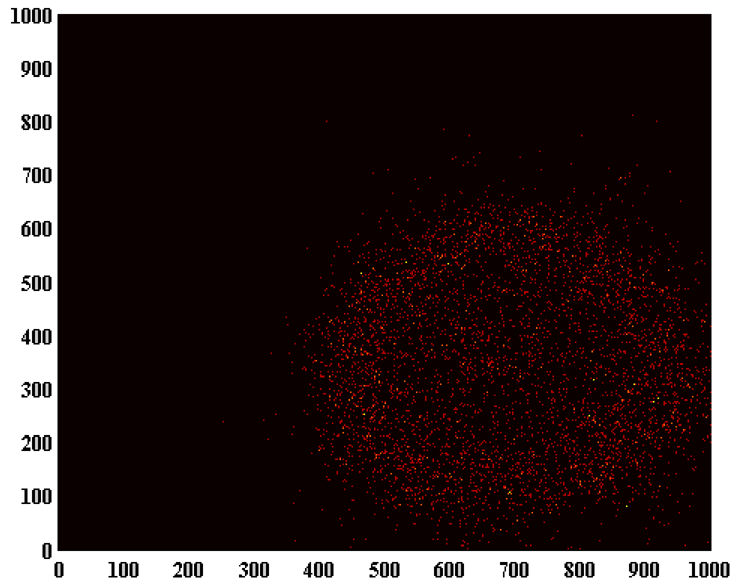


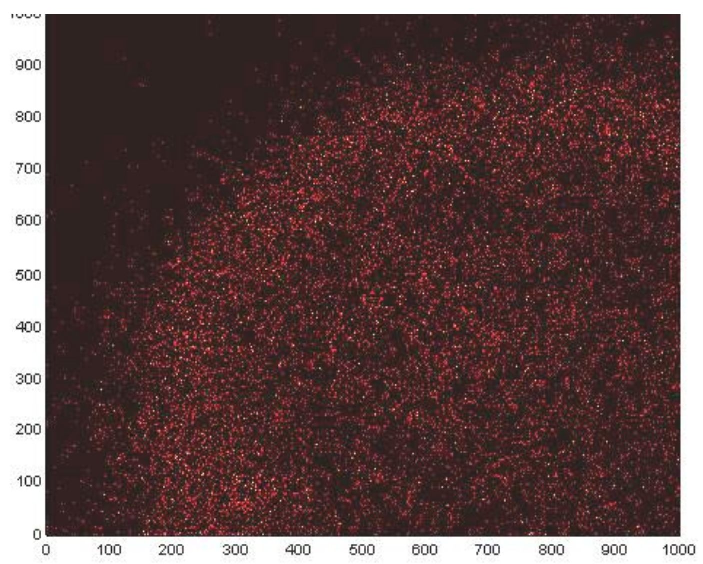


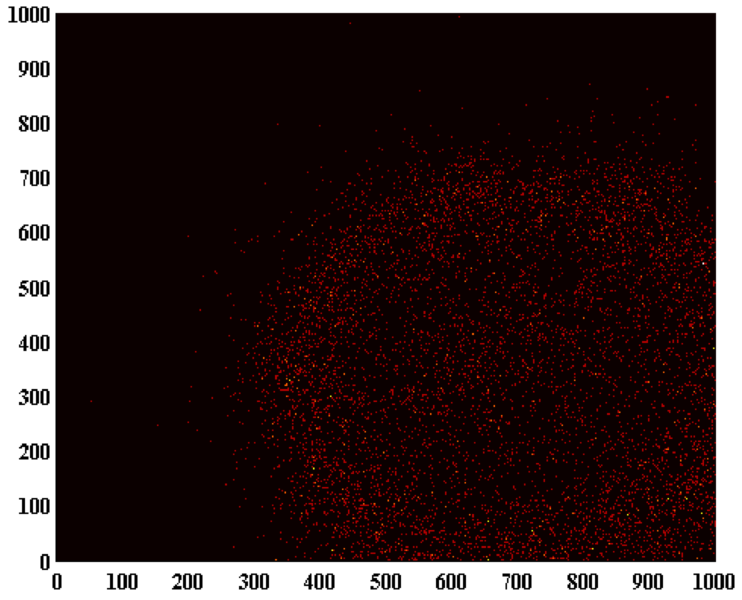

0 100 200 300 400 500 600 700 800 900 1000


0 100 200 300 400 500 600 700 800 900 1000
0 100 200 300 400 500 600 700 800 900 1000


0 100 200 300 400 500 600 700 800 900 1000
0 100 200 300 400 500 600 700 800 900 1000


0 100 200 300 400 500 600 700 800 900 1000
0 100 200 300 400 500 600 700 800 900 1000


0 100 200 300 400 500 600 700 800 900
0 100 200 300 400 500 600 700 800 900 1000


0 100 200 300 400 500 600 700 800 900 1000
0 100 200 300 400 500 600 700 800 900 1000

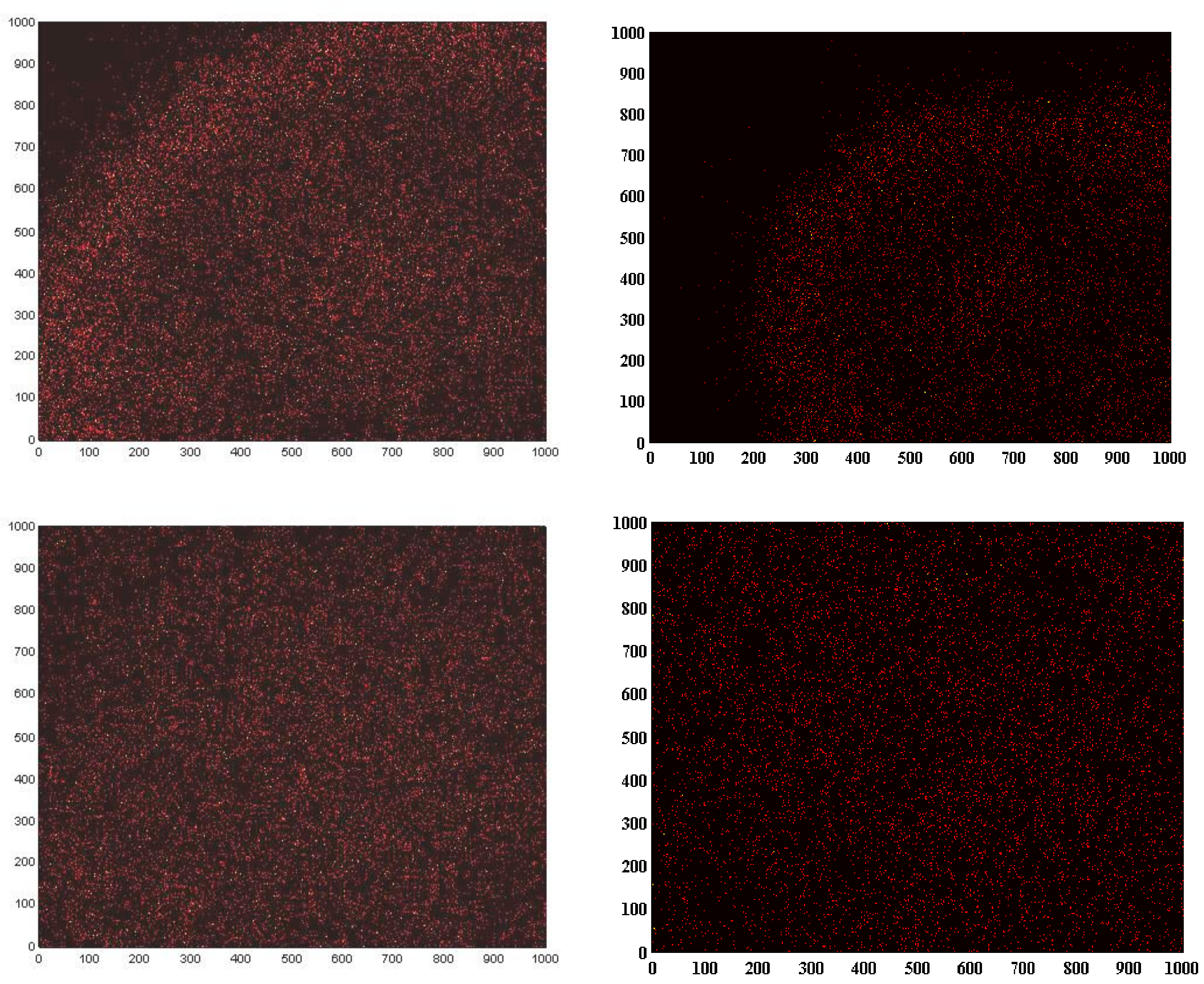


Fig.5 *We plotted the bacteria evolution for both normal gravity (the pale picture on the lefts) and hypergravity (darker pictures on the right) conditions. Both simulations are using one identical set of initial random seeds. The character of the movement is the same in both conditions as can be expected – the wave effect. But we can see the slower spread of the colony in the Petri Dish, and the lower density of the dots (corresponding to bacteria) on the hypergravity graph in comparison with the dots on the normal level gravity graph. The lower density corresponds to the slower reproduction rate of bacteria in hypergravity conditions.*

VI. The Main Result – The Memory Effect

When we competed in the *NASA Hyper-G Competition*, our main goal was to test for the presence of a memory effect. Here we have the chance to demonstrate it theoretically with our model.

The results we discuss so far show that the overall effects of gravity on the bacteria are

1. How soon they express their preservation gene,

2. How fast they are moving.

The formula for calculating the average speed for the Gaussian distribution of each agent is:

$$V_{Average} = V_{Min} + \left(\frac{(N_{fllagelas} - N^{Min}{}_{fllagelas})}{(N^{MAX}{}_{fllagelas} - N^{Min}{}_{fllagelas})} * \frac{(Length_{fllagelas} - Length^{Min}{}_{fllagelas})}{(Length^{MAX}{}_{fllagelas} - Length^{Min}{}_{fllagelas})} \right) * \frac{(V_{MAX} - V_{MIN})}{\alpha * g};$$

$$\alpha = \frac{1}{9.81}[\frac{s^2}{m}];\ g = n * 9.81\ [\frac{m}{s^2}];\ \text{Where n= } 1, 1.5, 2, 2.5 \ldots$$

It can be seen that in higher gravity, more and longer flagella are needed to result with the same speed. This is why the bacteria in hypergravity have a slower speed, but their average number of flagella and their length is higher than the same characteristics in bacteria in normal gravity (see Fig.6 and Fig.7 on pages 60-61). To restate this result: in hypergravity conditions the bacteria have a lower speed as well as more and longer flagella. If such bacteria are put back into conditions of normal gravity, their greater number of flagella and length of flagella will give them a much faster speed. You can clearly see this in the formula, because Gravity will equal 1 and only the higher values of number of flagella and their length will play a role in computing the speed. The controlled group, or the bacterial colony that was left in the normal conditions only, on average does not have such a high number of flagella and such length of flagella because these traits were not selected by natural selection as they were in hypergravity for the survival of the bacteria. Therefore, the bacteria transferred from hypergravity to normal gravity will be much faster than the ones produced in normal gravity. This is the memory effect – the agents "remember" the effect from the hypergravity and apply it in the new environment they are faced with. This is a consequence from natural selection in special hostile environment. In this case they would be much more physically fit in the normal gravity conditions because they will find food much faster and thus they will reproduce with a greater average rate. The two graphs are bell-shaped due to the starvation condition. In both graphs at both gravity levels, the average number and length of flagella increase because there is enough food and the faster the agents move, the more chances they have to find food, survive and pass on their characteristics to the next generation. At the peak of the bell, however, food has dropped to very low levels, and the bacteria must preserve their energy and move slower. Thus, they turn on their preservation gene and they also must decrease their speed. Therefore, only the bacteria with average smaller number and shorter length of flagella start surviving, and the bells take on a negative slope. The number and length of flagella reach a constant average value that is most suitable for these characteristics at these conditions; hence, the constant horizontal ends of the bell curves.

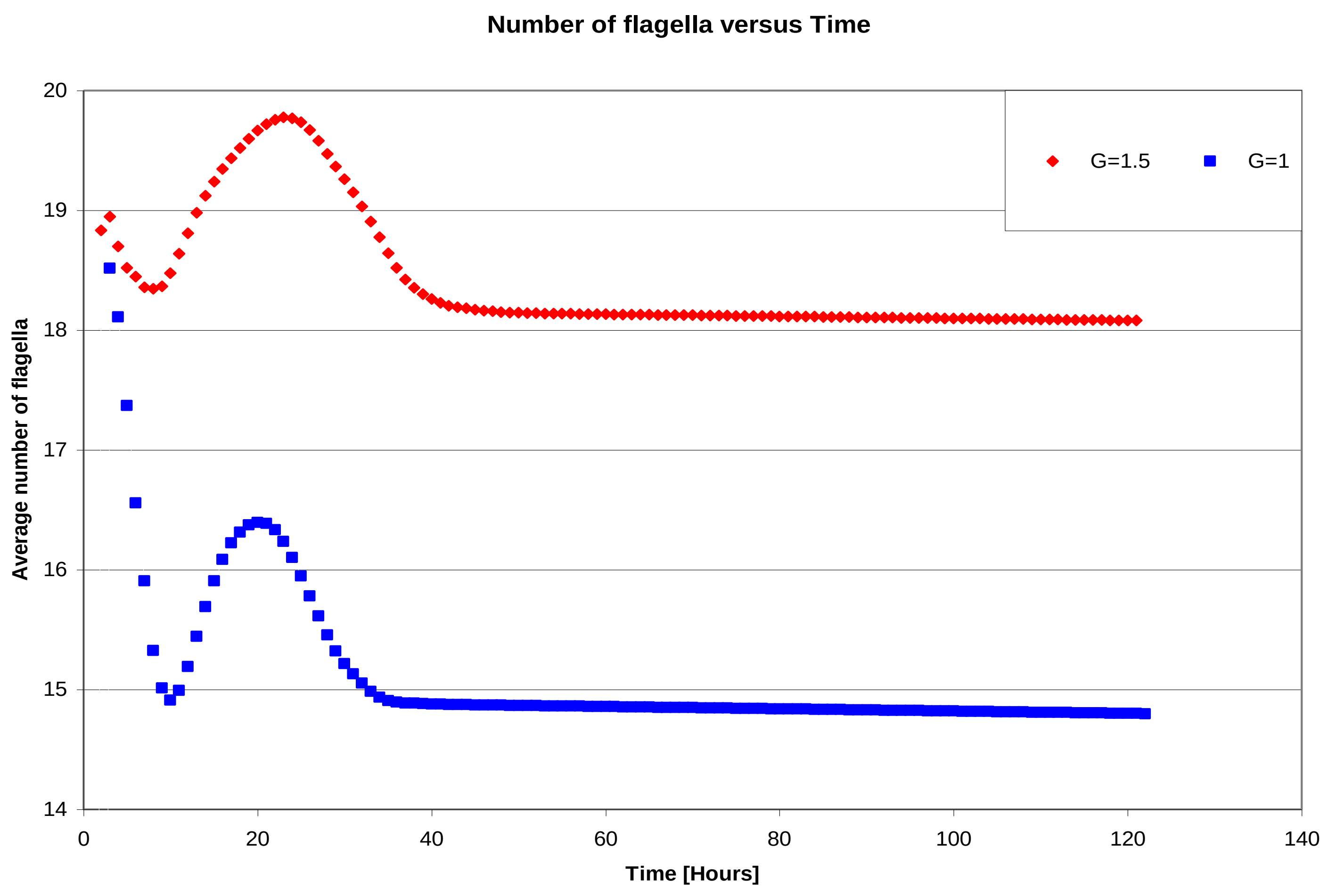


Fig.6 *This graph illustrates the change of the average number of flagella on each bacterium in time. The red bell-curve represents the evolutionary change of the average number of flagella on each bacterium in hypergravity conditions (G=1.5) and the blue one represents the change of number of flagella on each bacterium in normal gravity conditions (G=1).*

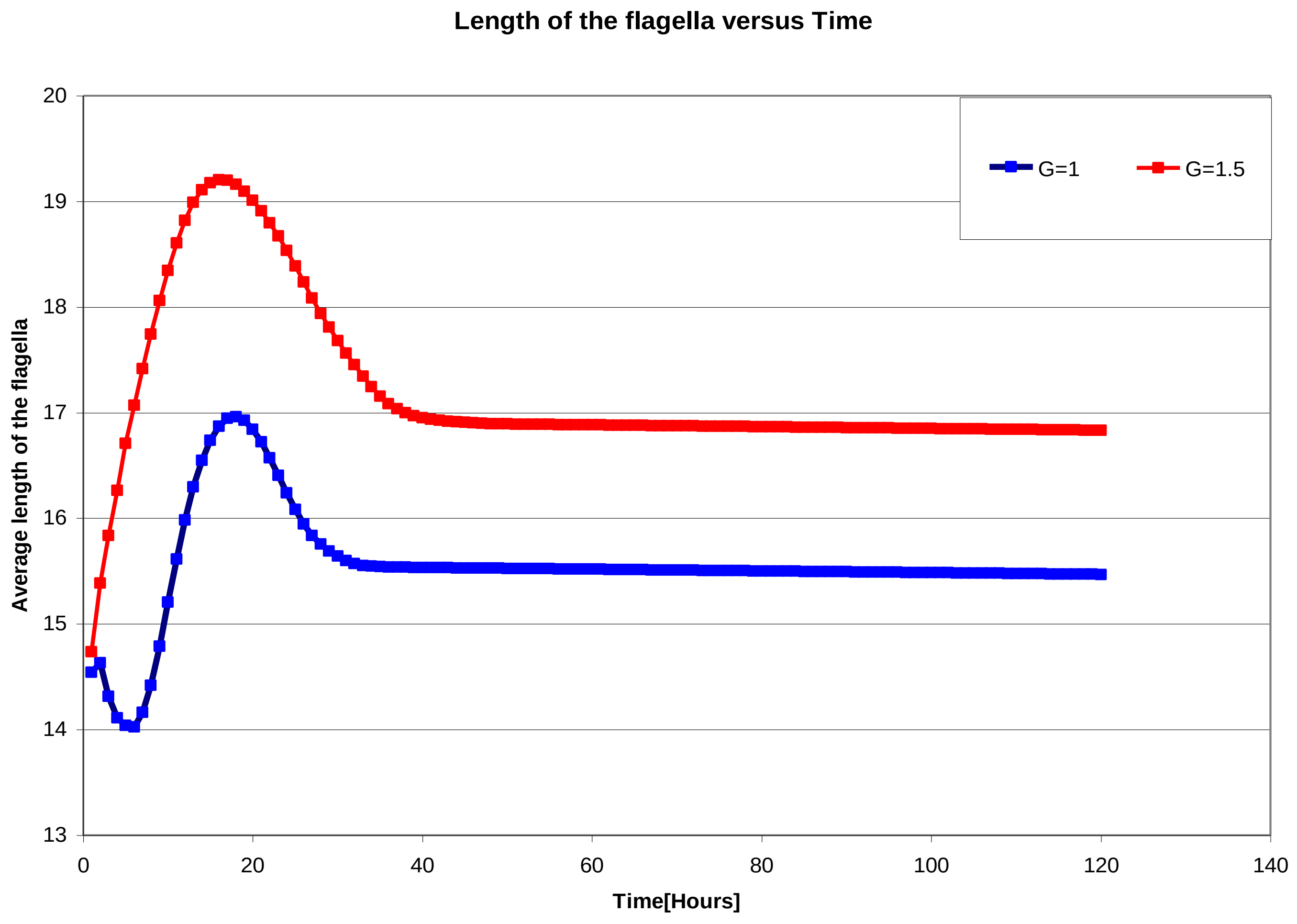


Fig.7 *This graph illustrates the change of the average length of the flagella on each bacterium in time. The red bell-curve represents the evolutionary change of the length of flagella on each bacterium in hypergravity conditions (G=1.5), and the blue one represents the change of length of flagella on each bacterium in normal gravity conditions (G=1)*

Conclusions

Our main conclusions can be summarized in five compact points:

I. The Evolutionary Agent-based model we created realistically describes evolution in time, the random walk type movement, and the growth rate of *E.coli* in starvation conditions.

II. The model shows the expected declining of the growth rate in hypergravity conditions.

III. We believe that transferring the gene characteristics, although phenomenological and based on Gaussian distribution with certain standard deviation, leads to reasonable results. Our results show that in conditions of hypergravity and starvation, the evolved bacteria will have an average longer and greater number of flagella, to compensate for the impedement from the gravity.

IV. Bacteria in hypergrvity express their genes for preservation of energy earlier than bacteria grown on normal earth conditions.

V. Due to the memory effect, bacteria produced in hyper gravity conditions will be more physically fit when transferred to normal gravity conditions because they will have a higher average speed.

Generally speaking, we can conclude that we have achieved in creating a realistic model that describes natural selection of a bacterial population in conditions of hostile environment. The most significant original achievement on the project is the demonstration of the memory effect. It proves that through natural selection, bacteria can be genetically altered to be physically fitter for their environment.

Applications

Our method would provide scientists with a way to modify bacteria to be physically strong to survive in a human host and this can be used for medical purposes, such as killing cancer tumors or, in the future, serving as agonists of strong white blood cells. Even more strikingly, if new helper T cells (white blood cells) that are physically fitter and able to withstand a body's immune system attacks, are introduced in the bone marrow of a person, AIDS could be eliminated as a detrimental immunodeficiency illness.

In our project, we result in bacteria that eat, reproduce and move much faster, than the same kind of bacteria grown on Earth's natural conditions. In a much more complex experiment, maybe involving special chemicals in the environment as well as specific outside environmental factors, Helpr T Cells can be made to fight HIV, disabling the virus from using their CD4 molecule and gaining access to the cell [47].

Most importantly, however, any biologist can use this method for making other desired characteristics in agents, making them able to perform specific desirable tasks.

In a late 2006 article, "THE EFFECT OF CHAGES OF GRAVITY ON HUMAN MONOCYTE CELL (TUR) PHAGOCYTOSIS" [48], scientists take normal human monocytes and put them into microgravity. Thus, the normal earth gravity conditions, being higher than the microgravity ones, play the role of the hyperravity conditions. Their results indicate a 60% increase in the cell's incidence of phagocytosis in the microgravity, which illustrates experimentally what we mean by the memory effect. Namely, the monocytes apperantly would be much stronger agents in the immune system and they could elllimante many more viruses, bacteria, and other pathogenic intruders in the human body than normally. However, the scientists also test the incidence of phagocytosis in monocytes under hypergravity. Their results are consistent with the idea that too much gravity will damage the cells' development. In our project, by establishing a relatively realistic model, we discover that above G=1.5 *E.coli* cells are so pressured that they can hardly move and as a result they decrease their average number and length of flagella, to feed on the resources around them and minimize energy expenditure (see fig.8 in Appendix A). By checking how microgravity and hypergravity would affect the monocytes, these scientists indirectly prove that the method of natural selection is a powerful tool to create fit organisms for useful, directed purposes or skilled tasks. As they have shown with their results, this could have an enormous impact on medicine, most significantly in the field of immunology. Moreso, if scientists do experiments entirely focused on our new concept of the memory effect, a whole new approach could be taken in making drugs, medicaments, and curing techniques.

Acknowledgments

We would like to thank Professor Dr. Anny Usheva for her help, ideas, and cooperation that made possible the organization of this project, Mr. L. B. Aleksandrov for his kindness and for consulting and helping us with the code and MPI, Mrs. Diane Medford for sponsoring the Supercomputing Challenge, and Supercomputing Consulter for help on the concept of hostile environments.

References

[1] Anderson, Christopher. “Tumor-Killing Bacteria.” Blood. Vol. 94, No. 8. June 02, (2006).

[2] Saint-Ruf, Claude, Taddei, François and Matic, Ivan. “Stress and Survival of Aging Escherichia coli in rpoS colonies.” Genetics. 168(1): 541–546; (2004).

[3] Finkel, Steven and Kolter, Roberto. “Evolution of Microbial Diversity During Prolonged Starvation.” Vol. 96, Issue 7, 4023-4027, March 30, (1999).

[4] Stewart EJ, Madden R, Paul G, Taddei F. “Aging and Death in an Organism That Reproduces by Morphologically Symmetric Division.” PLoS. Biol. 3(2): e45 (2005).

[5] Visick, Jonathan. PhD Dept. of Biology. “Protein Repair in Aging E.coli.” <http://jonathan.visick.faculty.noctrl.edu/research/aging.htm>.

[6] “Environmental Change and Our Health.” ECOHEALTH Glossary. <http://www.ecohealth101.org/glossary.html>.

[7] “Biosociety and the Knowledge Based Bio-Economy.” © European Commissions. Bio Glossary. 1995-2007. <http://ec.europa.eu/research/biosociety/library/glossarylist_en.cfm?Init=E>.

[8] Onconomics Intranet Glossary on the Web. Onconomics Corporation. 2003. <http://www.bscs.org/onco/glossary.htm>.

[9] BioIndustry Glossary on the Web. BioIndustry Association, London, England. <http://www.bioindustry.org/cgi-bin/contents_view.pl?LEVEL1=9>.

[10] “Glossary of Water Terms.” Reprinted from The Drinking Water Dictionary. American Water Works Association. © 2000. <http://www.mwdh2o.com/mwdh2o/pages/yourwater/glossary/glossary01.html>.

[11] “Proteobacteria”. Wikipedia, the free encyclopedia. Wikimedia Foundation, Inc. Last modified 19:46, 10 March 2007. <http://en.wikipedia.org/wiki/Proteobacteria>.

[12] “Escherichia coli.” Wikipedia, the free encyclopedia, Wikimedia Foundation, Inc. Last modified 20:23, 1 April 2007. <http://en.wikipedia.org/wiki/Escherichia_coli>.

[13] “Facultative Anaerobe.” Wikipedia, the free encyclopedia, Wikimedia Foundation, Inc. Last modified 14:06, 9 March 2007. <http://en.wikipedia.org/wiki/Facultative_anaerobe>.

[14] “Model Organism.” Wikipedia, the free encyclopedia, Wikimedia Foundation, Inc. Last modified 17:23, 29 March 2007. <http://en.wikipedia.org/wiki/Model_organism>.

[15] “Gram Negative.” Wikipedia, the free encyclopedia, Wikimedia Foundation, Inc. Last modified 02:11, 22 March 2007. <http://en.wikipedia.org/wiki/Gram_negative>.

[16] Neil A. Campbell and Jane B. Reece, *BIOLOGY*, 7-th AP - Edition, page 98, (2005).

[17] Burgard, A., Vaidyaraman, S., and Maranas, C. "Minimal reaction sets for Escherichia coli metabolism under different growth requirements and uptake environments. "Biotechnology Progress. 2001 Sep-Oct;17(5):791-7. <http://www.ncbi.nlm.nih.gov/entrez/query.fcgi?cmd=Retrieve&db=PubMed&list_uids=11587566&dopt=Abstract>.

[18] Todar, Kenneth. “Todar's Online Textbook of Bacteriology.” University of Wisconsin Madison Department of Bacteriology. <http://textbookofbacteriology.net/regulation.html>.

[19] “Journal of Clinical Microbiology.” October 2003: 4852-4854, Vol. 41, No. 10. <http://jcm.asm.org/cgi/content/full/41/10/4852>.

[20] “E.coli Statistics.” Institute for Bimolecular Design. Project CyberCell. Database: CCDB. Last updated June 28, 2006. <http://redpoll.pharmacy.ualberta.ca/CCDB/cgi-bin/STAT_NEW.cgi>.

[21] Sullivan, Jim. "How Bacteria Swim and Tumble." Cellsalive.com. ©2006. <http://www.cellsalive.com/animabug.htm>.

[22] Stewart, Eric, Madden, Richard, Paul, Gregory, and Taddei, François. "Aging and Death in an Organism That Reproduces by Morphologically Symmetric Division." PloS, Public Library of Science research article. 1 Feb 2005. <http://biology.plosjournals.org/perlserv/?request=getdocument&doi=10.1371/jornal.pbio.0030045>.

[23] "Aging and Death in E. coli." Public Library of Science. PLoS Biol 3(2): e58. (2005)

[24] "Bacterial Conjugation." Wikipedia, the free encyclopedia. Wikimedia Foundation, Inc. Last modified 01:33, 27 March 2007. <http://en.wikipedia.org/wiki/Bacterial_conjugation>.

[25] Campbell, Neil and Reece, Jane. *BIOLOGY*. 7-th AP - Edition, page 350 (2005).

[26] "Section L." HARDY Diagnostics. Microglossary lexicon of microbiology terms & abbreviation. <http://www.hardydiagnostics.com/Glossary-L.html>.

[27] Finkel, Steven and Kolter, Roberto. "Evolution of microbial diversity during prolonged starvation . " PANS National Academy of Sciences.

Vol. 96, Issue 7, 4023-4027, March 30, 1999.
<http://www.pnas.org/cgi/content/abstract/96/7/4023>.

[28] Saint-Ruf, Claude, Taddei, François, and Matic, Ivan. "Stress and Survival of Aging Escherichia coli rpoS Colonies." Genetics Journal List on the Web. 2004 September; 168(1): 541–546. doi: 10.1534/genetics.104.028704.
<http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=1448099>.

[29] Record, M. T., W. S. Reznikoff, M. L. Craig, K. L. McQuade and P. J. Schlax, *"*Escherichia coli polymerase (Eσ70), promoters, and the kinetics of the steps of transcription initiation.*" Escherichia coli and Salmonella*. Cellular and Molecular Biology. Edited by F. C. Neidhardt and R. Curtiss. ASM Press, Washington, DC. (1996): 792–821

[30] Storz, G., and R. Hengge-Aronis. "Bacterial Stress Responses." ASM Press. Washington, DC. (2000).

[31] Ferenci, T. "Hungry bacteria—definition and properties of a nutritional state. "Environ. Microbiol. **3:** (2001): 605–611

[32] Farrell, M. J., and S. E. Finkel. "The growth advantage in stationary-phase phenotype conferred by rpoS mutations is dependent on the pH and nutrient environment*."* J. Bacteriol. 185: 7044–7052, (2003).

[33] Zinser, Erik and Kolter, Roberto**.** Journal of Bacteriology. (2000): 4361-4365, Vol. 182, No. 15.

[34] Kacena A., Leonard P., Todd P., and Luttges W. "Low gravity and inertial effects on the growth of E. coli and B. subtilis in semi-solid media." Aviation, space, andenvironmental medicine on the Web. 1997, vol. 68, no12, pp. 1104-1108 (21 ref.) <http://cat.inist.fr/?aModele=afficheN&cpsidt=2092221>.

[35] "Prokaryotic Cytoskeleton." Wikipedia, the free encyclopedia, Wikimedia Foundation, Inc. Last modified 20:50, 1 April 2007. <http://en.wikipedia.org/wiki/Cytoskeleton#The_prokaryotic_cytoskeleton>.

[36] Miller, Karen and Dr. Phillips, Tony. "Curious Skeletons." Exploration NASA on the Web, November 7, 2005. <http://exploration.nasa.gov/articles/19jun_cytoskeletons_lite.html,>.

[37] "Agent Based Model Theory." Wikipedia, the free encyclopedia. Wikimedia Foundation, Inc. Last modified 13:25, 14 March 2007. <http://en.wikipedia.org/wiki/Agent_based_model#Theory>.

[38] Prof. Fischer, Teresa. “Prokaryotic Structure Overview” Faculty WebPage.
<http://faculty.ircc.edu/faculty/tfischer/micro%20resources.htm>.

[39] Brain, Marshall. “Cell.” HowStuffWorks, Inc. © 1998-2007.
<http://science.howstuffworks.com/cell1.htm>.

40] “Lifecycle of e.coli.jpg.” English Wikipedia encyclopedia. Wikimedia Foundation, Inc.
<http://en.wikipedia.org/wiki/Image:Lifecycle_of_e_coli.jpg#filehistory>.

[41] “BacterConjugation.png.” Wikipedia, the free encyclopedia, Wikimedia Foundation, Inc.
<http://en.wikipedia.org/wiki/Image:BacterConjugation.png>.

[42] “Population Dynamics.” Geosciences at the University of Arizona.
<http://www.geo.arizona.edu/Antevs/nats104/00lect21.html>.

[43] Prof.Dr. Toprak, Hikmet.“The Microscope.” Toprak Home Page on the Web. © 2006.
<http://web.deu.edu.tr/atiksu/toprak/ani4101.html>.

[44] Edge, Tom. “Tracking Sources of Bacteria.” Environment Canada on the Web. Last updated: 29 March 2004. <http://www.nwri.ca/envirozine/issue33-e.html>.

[45] John van Wyhe. “Charles Darwin.” The Victorian Web. National University of Singapore. History & philosophy of science, Cambridge University. <http://www.victorianweb.org/science/darwin/intro.html>.

[46] Sullivan, Jim. “How Bacteria Swim and Tumble.” Cellsalive.com. ©2006. <http://www.cellsalive.com/animabug.htm>.

[47] Campbell, Neil, and Reece, Jane. “AIDS.” Biology. 7th ed. (2005): 918.

[48] Johnson, Cassidy. “The Effect of Changes of Gravity on Human Monocyte Cell (TUR) Phagocytosis.” Gravitational and Space Biology 19(2) August 2006. <http://asgsb.org/bulletins/v19n2/147%20-%20148.pdf>

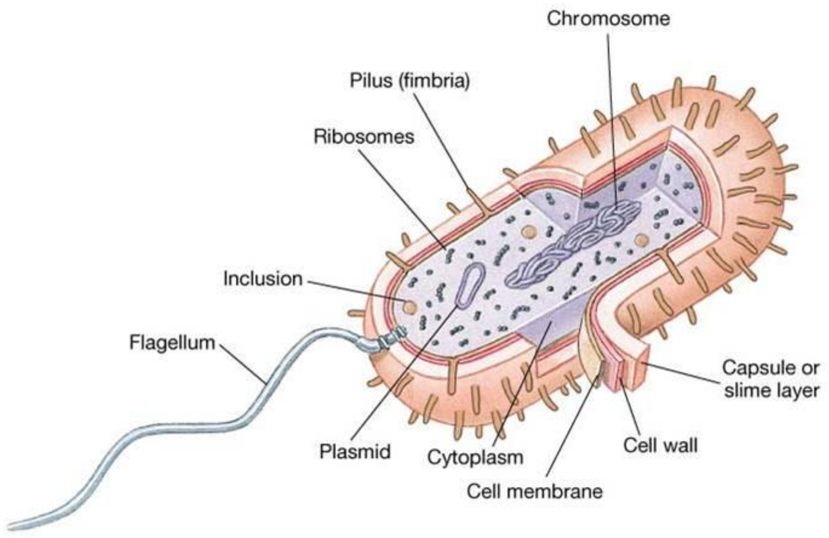
[49] Kato, Yuko. Mogami, Yoshihiro. Baba, Shoji A. Responses to Hypergravity in Proliferation of Paramecium tetraurelia < http://ci.nii.ac.jp/naid/110006162822/en/>

[50] Phillips, Robert W. Space Effects on Life Forms. <http://ncmr04610.cwru.edu/events/mssp2003/space_phillips.pdf>

[51] Zobell, Claude E. Cobet, Andre B. Growth, reproduction, and death rates of Escherichia Coli at increased Hydrostatic Pressures. <http://www.pubmedcentral.nih.gov/picrender.fcgi?artid=278050&blobtype=pdf>

Appendix A

Graphs & Pictures



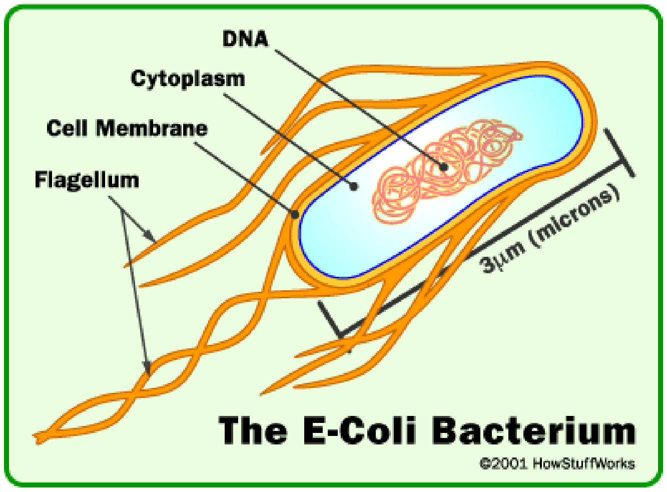


Fig. 1.1

Fig. 1.2

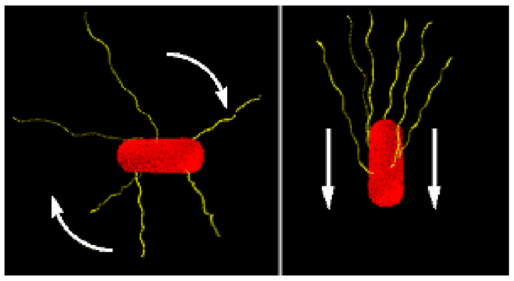


Fig 1.3

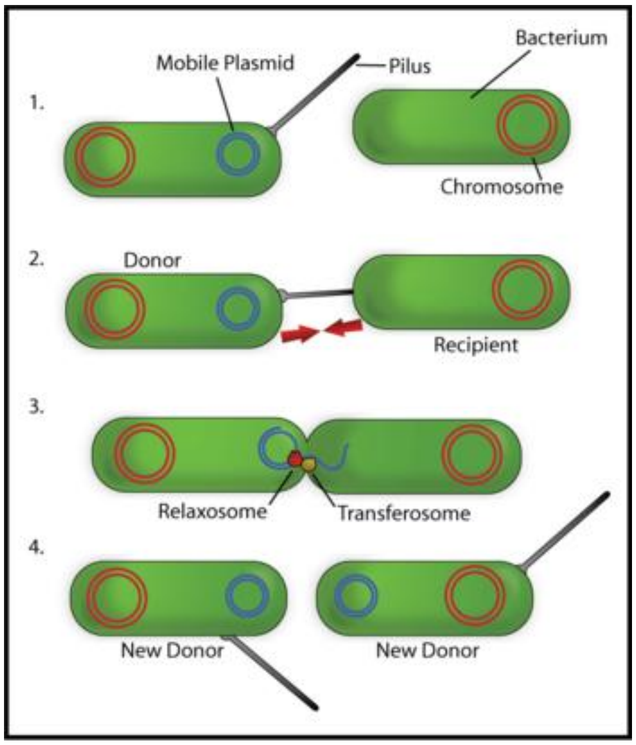


Fig 1.5 ←

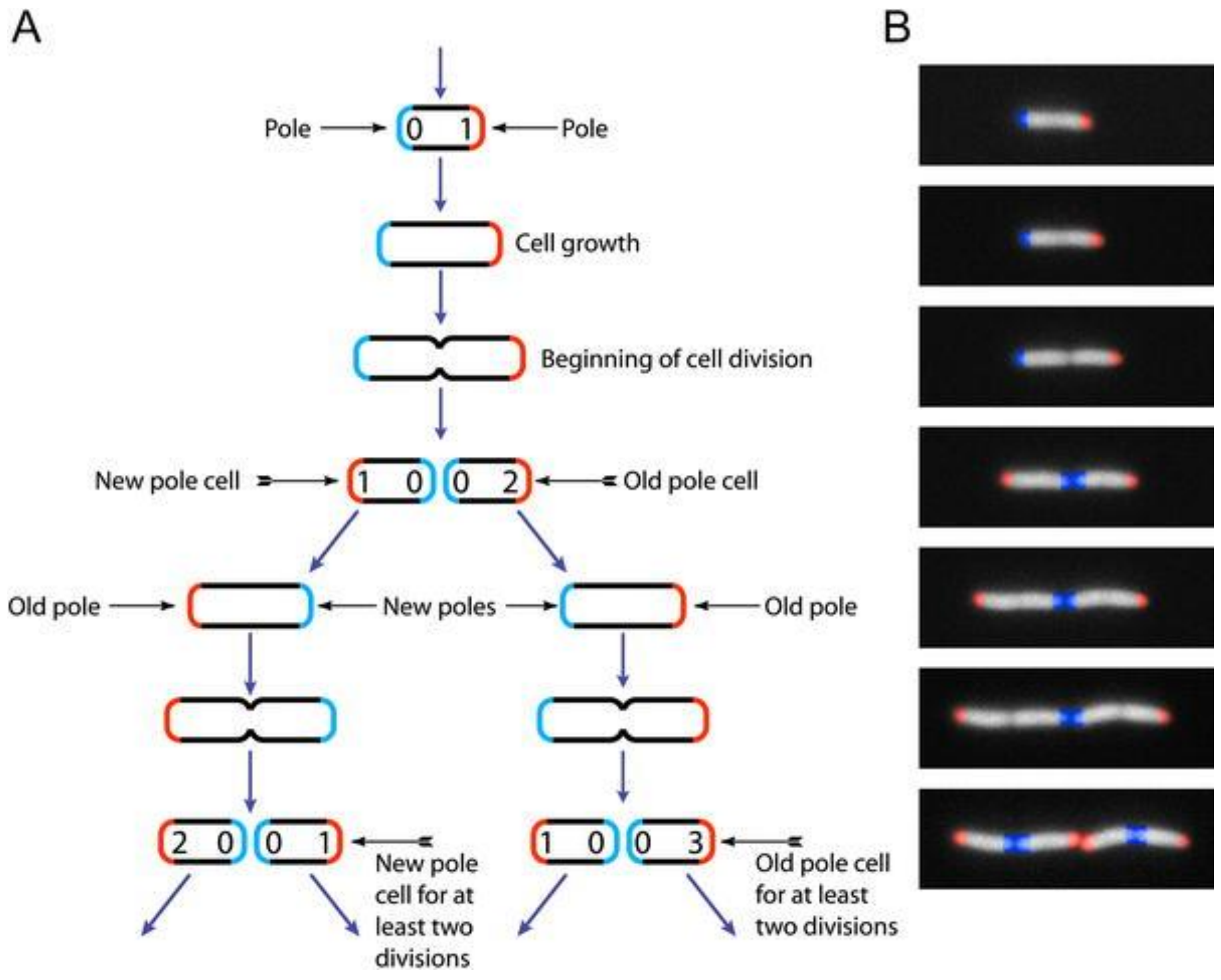

A
B
Pole
0
1
Pole
Cell growth
Beginning of cell division
New pole cell
1
0
0
2
Old pole cell
Old pole
New poles
Old pole
2
0
0
1
New pole cell for at least two divisions
1
0
0
3
Old pole cell for at least two divisions

Fig. 1.4

Fig. 1.6

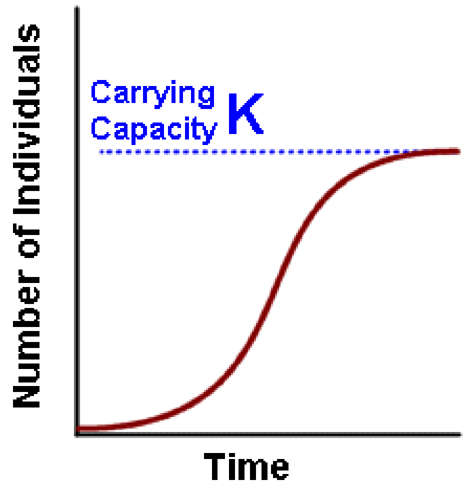
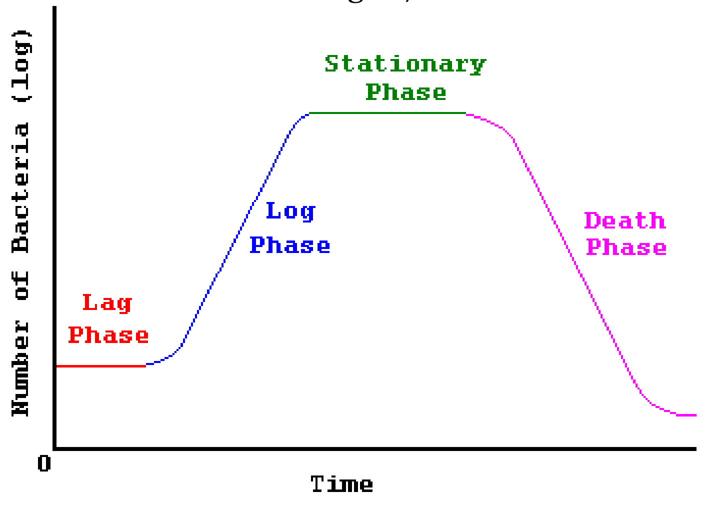


Fig. 1.7



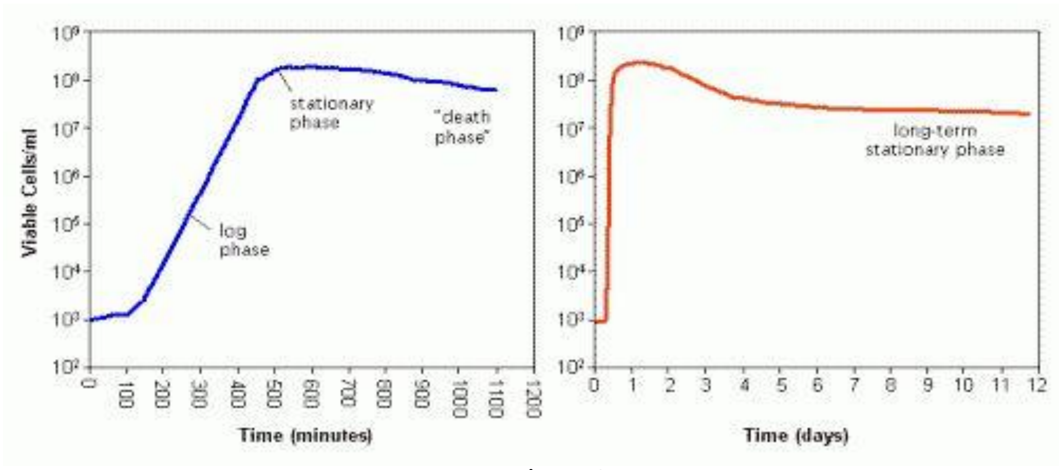


Fig. 1.8

Fig. 2.0

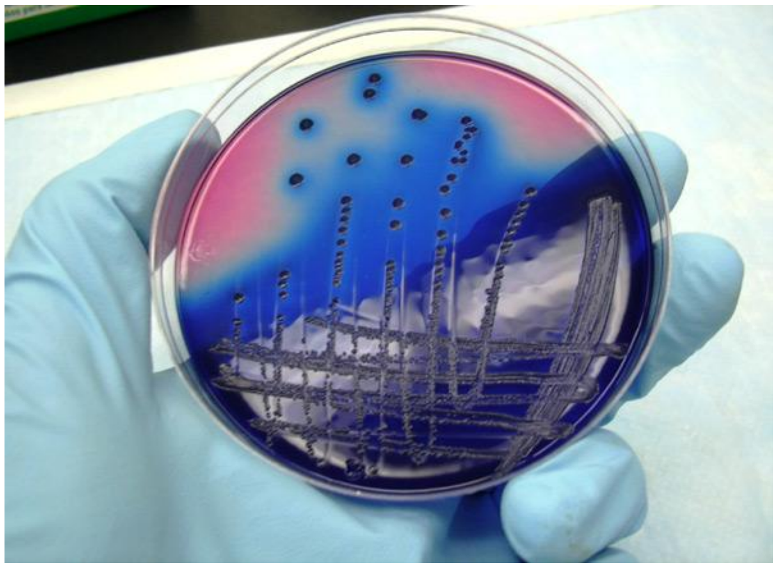


Fig. 1.9

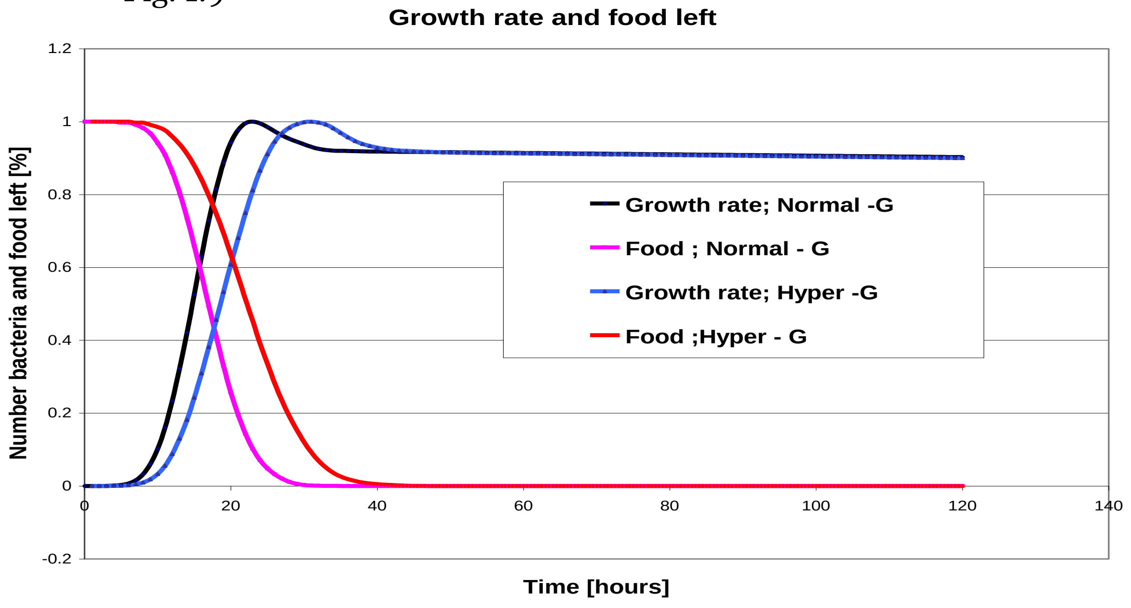
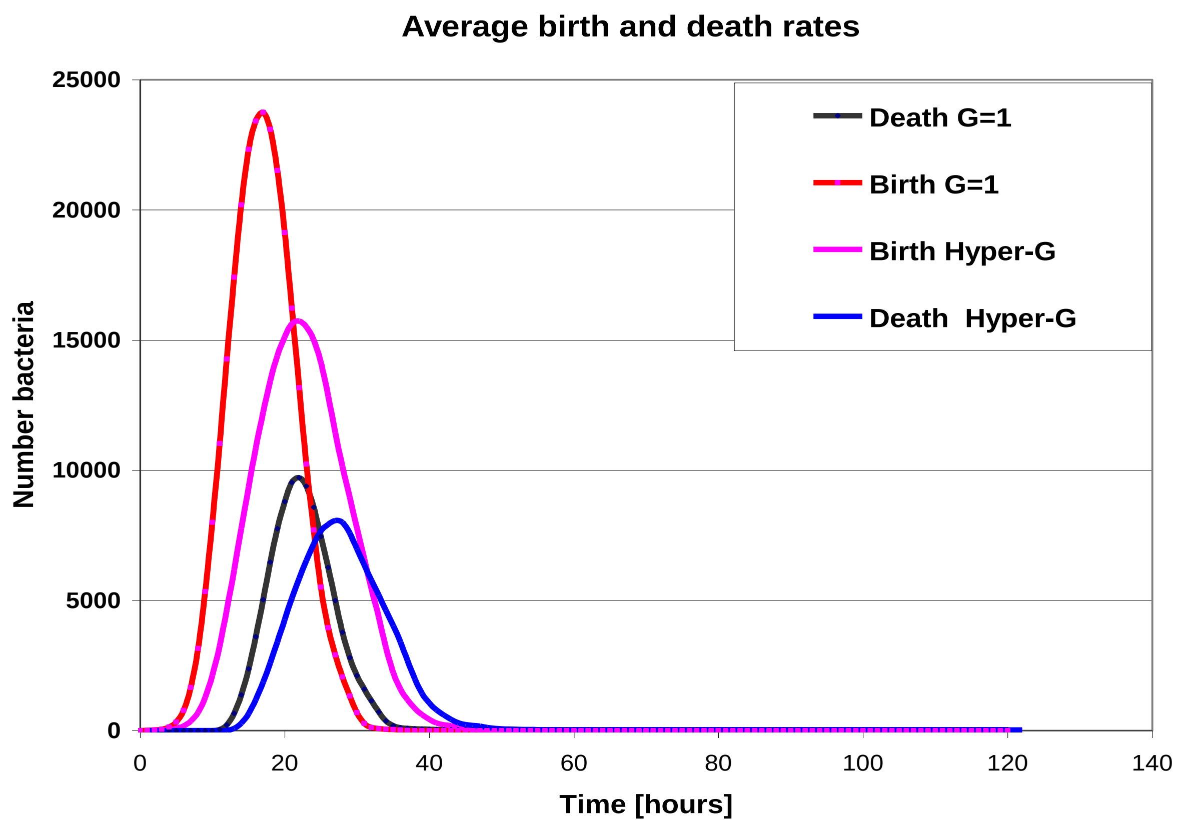
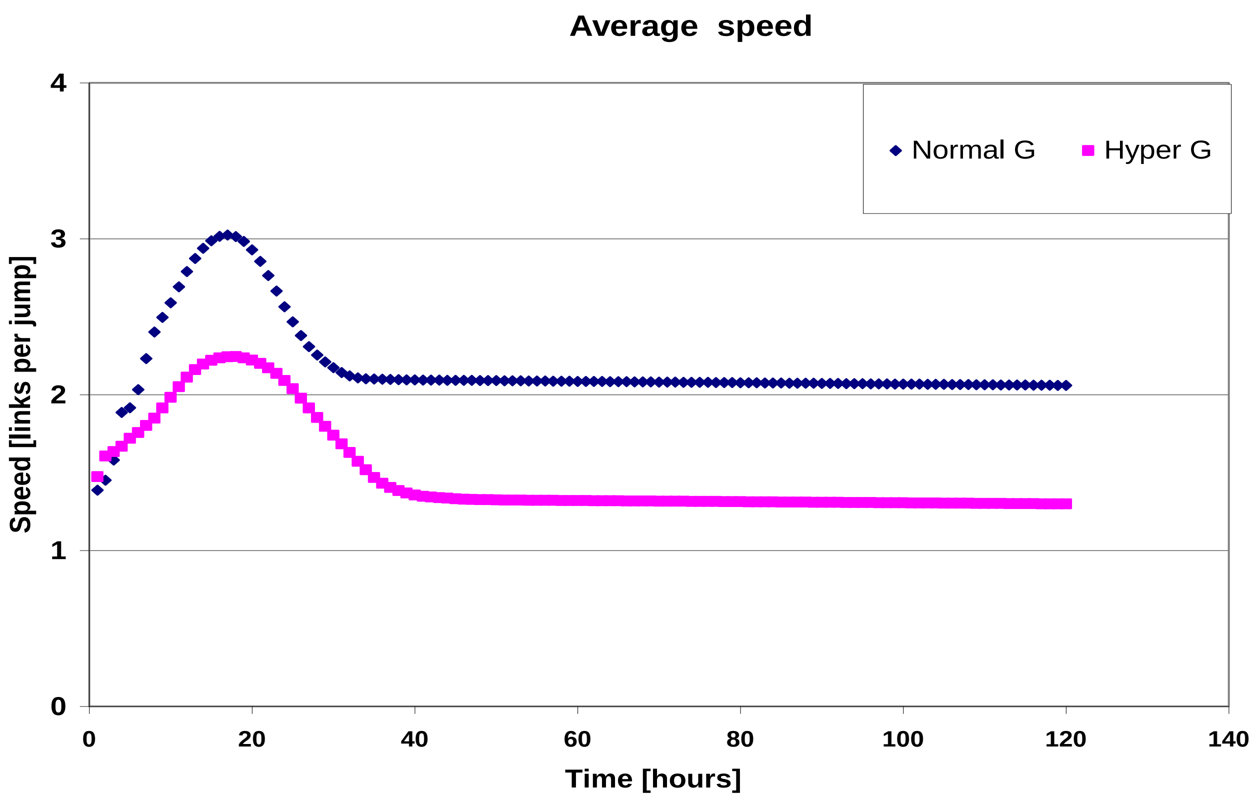


Fig.1



Average birth and death rates
Number bacteria
25000
20000
15000
10000
5000
0
0
20
40
60
80
100
120
140
Time [hours]
Death G=1
Birth G=1
Birth Hyper-G
Death Hyper-G

Fig. 2



Average speed
Speed [links per jump]
4
3
2
1
0
0
20
40
60
80
100
120
140
Time [hours]
Normal G
Hyper G

Fig 3

Fig. 3

Fig.7

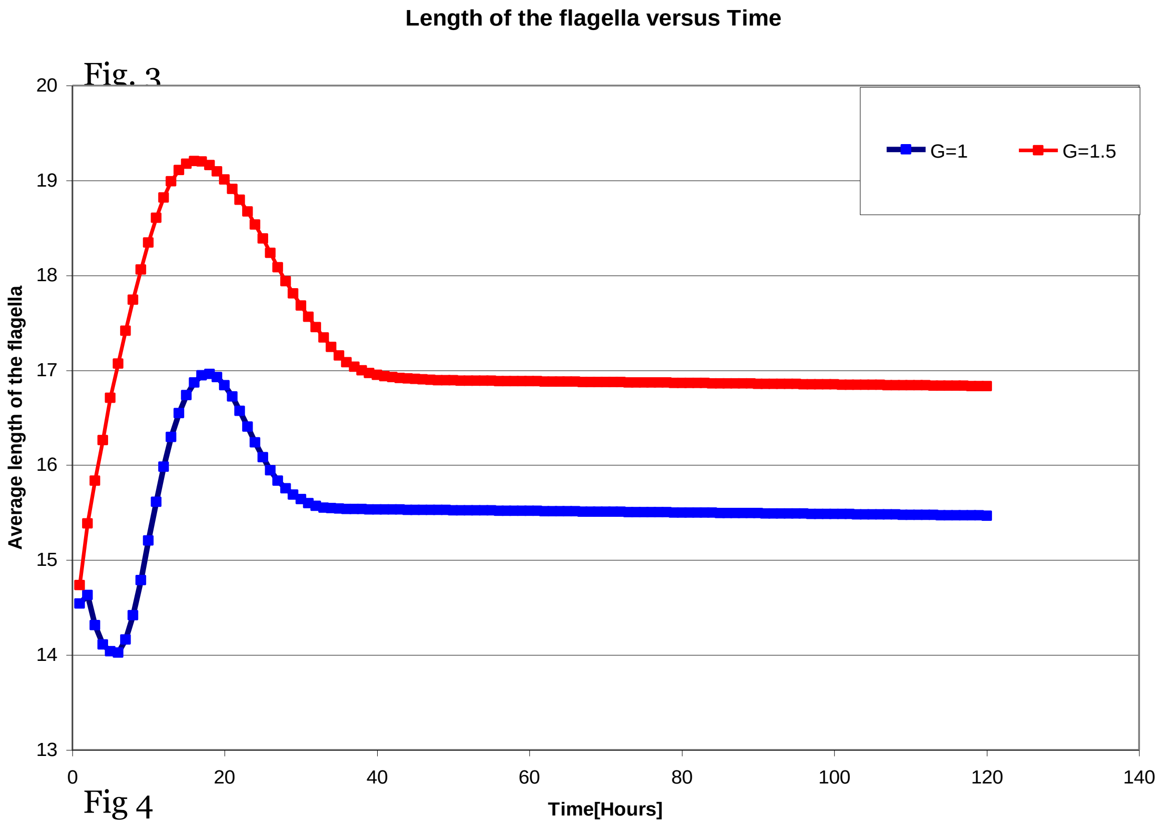
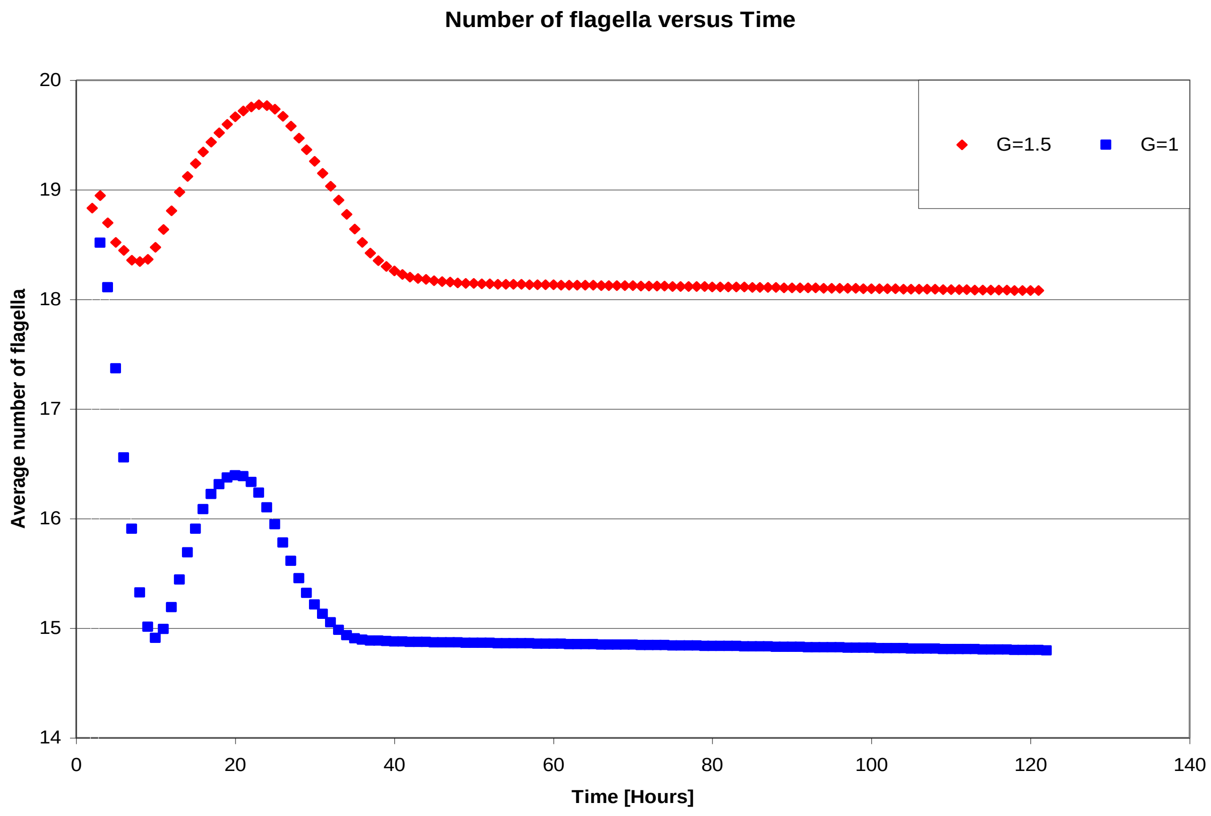


Fig 4

Fig.6



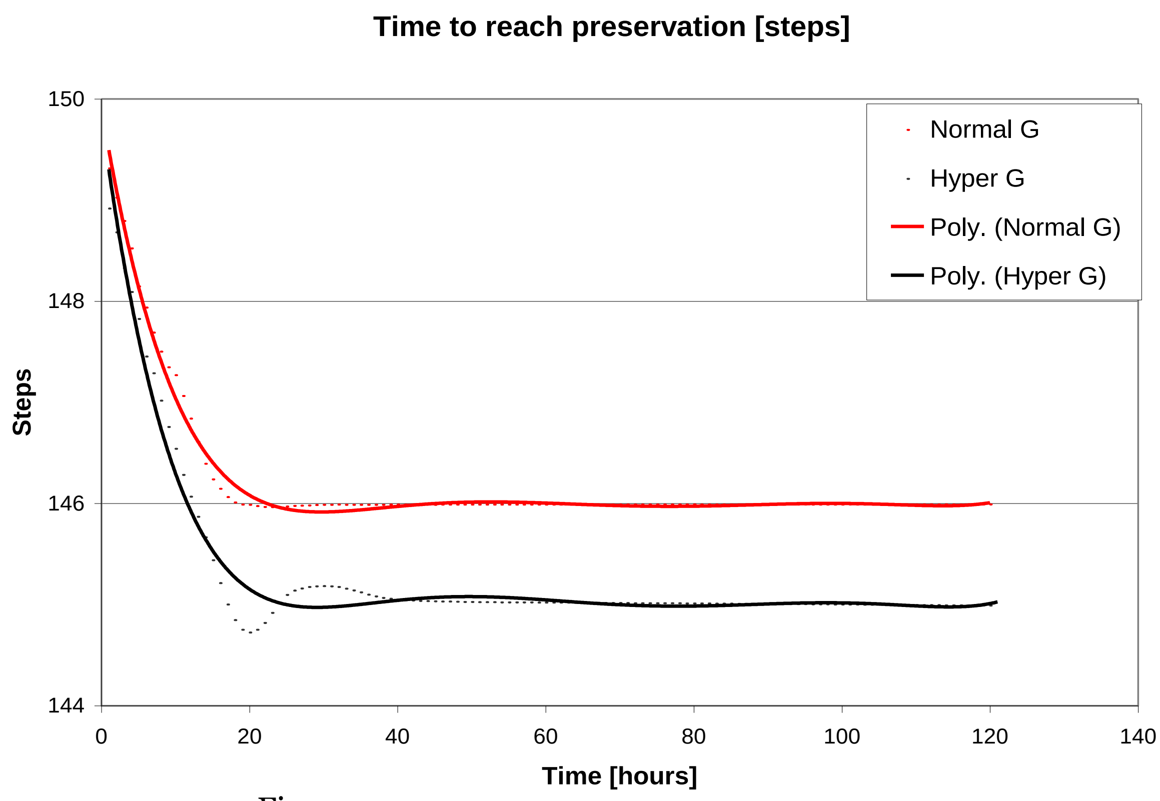

Time to reach preservation [steps]
150
148
146
144
Steps
0
20
40
60
80
100
120
140
Time [hours]
Normal G
Hyper G
Poly. (Normal G)
Poly. (Hyper G)

Fig.4

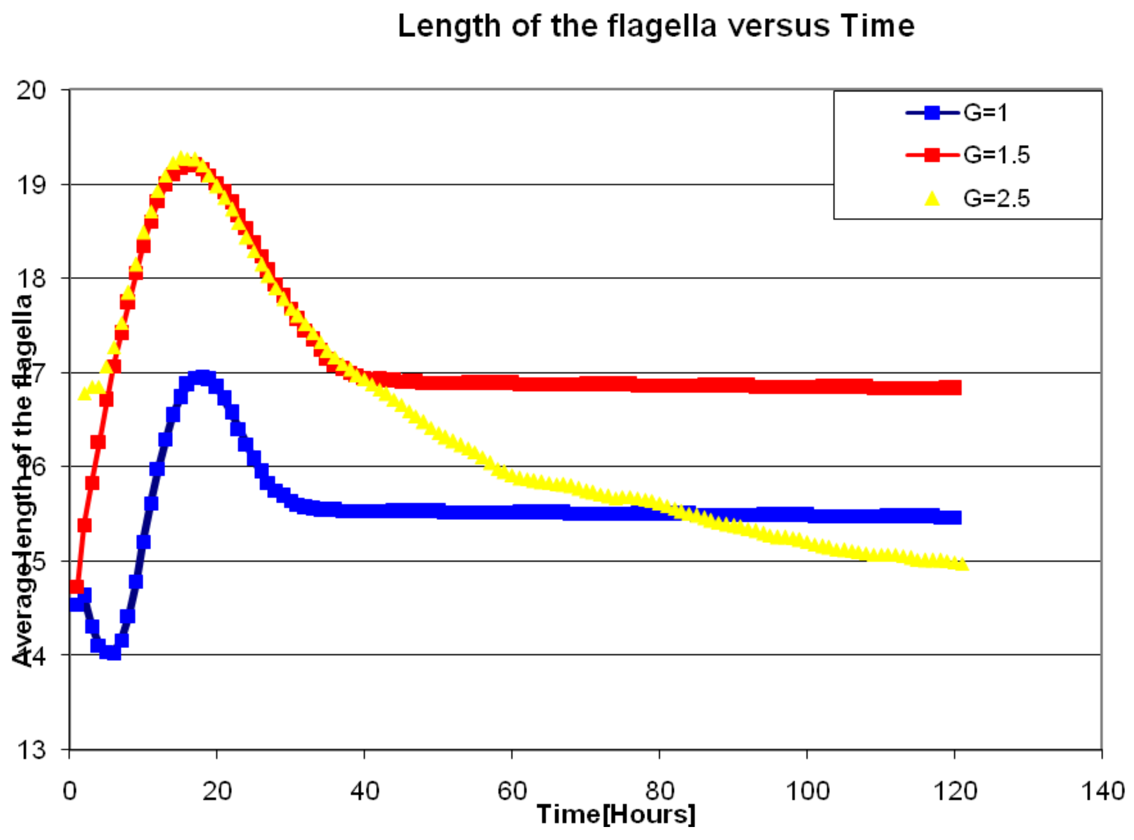

Length of the flagella versus Time
20
19
18
17
16
15
14
13
Average length of the flagella
0
20
40
60
80
100
120
140
Time[Hours]
G=1
G=1.5
G=2.5

Fig. 8

Appendix B

Source Code

File: SCC_v_1_1.c

```c
#define HANDLE_ERRORS
#define VERBOSE
#define USE_MPI
#undef USE_MPI
/*Standard Libraries*/
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <time.h>
#ifdef USE_MPI
  #include "mpi.h"
#endif

#ifdef USE_MPI
#define MAX_HOURS 1000
  /*MPI variables*/
  static long count_e_coli_sum[MAX_HOURS], count_e_coli_sum_mpi[MAX_HOURS];
  static double avg_energy_sum[MAX_HOURS], avg_energy_sum_mpi[MAX_HOURS];
  static long death_e_coli_sum[MAX_HOURS], death_e_coli_sum_mpi[MAX_HOURS];
  static double total_energy_sum[MAX_HOURS], total_energy_sum_mpi[MAX_HOURS];
  static int numprocs, myid;
#endif
/*Common varaibles*/
struct _e_coli
{
     int x, y; /*the cordinates of e_colies */
     short number_of_flagellas;  /*the number of all flagellas, normally between 5 and 20*↙
    /
     double avg_length_flagella; /*the average length of a flagella*/
     double speed_of_eating; /*how much food per move will the e colli eat*/
     int max_speed; /*the maxim jump that the e.coli can make. It is calculated by the    ↙
    number of flagellas*/
     double energy; /* measured in ATP between 1.65M - 3M scaled ATP*/
     long steps; /*the steps after the last division*/
     int energy_status;
     double preservation;
     long preservationSteps;
};

static double Anumber_of_flagella, Aavg_length_flagella, Aspeed_of_eating, Amax_speed,    ↙
    Apreservation, ApreservationSteps;

/*Idums used with ran2 and gasdev*/
struct _idum
{
       /*general*/
       long init;
       long density;
       long radius;
       long area;
       long angle;
       /*e coli specific*/
       long x;
       long y;
       long number_of_flagellas;
       long avg_length_flagella;
       long speed_of_eating;
       long energy;
       long preservation;
       long preservationSteps;
} idum;

/*Custom Libraries*/
#include "ran2.c"
#include "gasdev.c"
#include "general.c"
#include "SCC_structs_v_1_1.c"
```

```
#include "e_coli.c"



int main(int argc, char *argv[])
{
    int step, begin_e_coli = 300, e_coli_i, x, y;
    double number_days = 5;
    long total_steps;
    #ifdef USE_MPI
    long current_mpi_step = 0 , mpi_step, mpi_loop;
     MPI_Init(&argc, &argv);
     MPI_Comm_size(MPI_COMM_WORLD,&numprocs);
     MPI_Comm_rank(MPI_COMM_WORLD,&myid);
        #ifdef VERBOSE
          if (myid == 0) printf("The program was started on %ld processors\n", numprocs);
        #endif
    #endif
    FILE *out, *map_energy, *map_count, *avg;
    char map_energy_name[100], map_count_name[100];
    time_t start, end, t;
    #ifdef VERBOSE
    init_process("Starting the program");
    complete_process();
    #endif
    STEP_TO_HOURS_L = (long) (STEP_TO_HOURS);
    idum = init_random_numbers(); /*initilize the idums*/
    create_initial(begin_e_coli, idum);
    init_area(&idum.density, &idum.area);
    #ifdef USE_MPI
    #else
    out = fopen("E_coli_output.txt", "w");
    avg = fopen("E_coli_avg.txt", "w");
    fprintf(avg,"number of flagella\tavg length_flagella\tspeed_of_eating\tmax_speed\    ↙
    tpreservation\tpreservationSteps\n");
    #endif
    #ifdef VERBOSE
    printf("1 hour = %.0lf steps\n", STEP_TO_HOURS);
    #endif
    total_steps = (long)(number_days * STEP_TO_HOURS * 24);
    #ifdef VERBOSE
    printf("The total simulation is %.2lf days or %ld steps\n\n", number_days,          ↙
    total_steps);
    #endif
    for(step = 0; step <= total_steps; step++)
    {
      if (step % STEP_TO_HOURS_L == 0)
      {

      #ifdef VERBOSE
       #ifdef USE_MPI
        if (myid == 0)
        {
       #endif
          printf("%.2lf\t%ld\t%ld\t%.3lf\t%.3lf\t%.3lf\t%.2e\t%ld\n",  step / STEP_TO_HOURS↙
    , count_e_coli, count_death_e_coli, avg_death/STEP_TO_HOURS_L, avg_born/               ↙
    STEP_TO_HOURS_L, avg_e_coli_energy, total_food, pres_number);
       #ifdef USE_MPI
        }
       #endif
      #endif
      #ifdef USE_MPI
      count_e_coli_sum[current_mpi_step] = count_e_coli;
      avg_energy_sum[current_mpi_step] = avg_e_coli_energy;
      death_e_coli_sum[current_mpi_step] = count_death_e_coli;
      total_energy_sum[current_mpi_step] = total_food;
      current_mpi_step++;
      #else
        fprintf(out,"%.2lf\t%ld\t%ld\t%.3lf\t%.3lf\t%.3lf\t%.2e\n",  step / STEP_TO_HOURS,↙
     count_e_coli, count_death_e_coli, avg_death/STEP_TO_HOURS_L, avg_born/STEP_TO_HOURS_L↙
    , avg_e_coli_energy, total_food);
        fprintf(avg,"%lf\t%lf\t%lf\t%lf\t%lf\t%lf\n",Anumber_of_flagella/count_e_coli,    ↙
    Aavg_length_flagella/count_e_coli, Aspeed_of_eating/count_e_coli, Amax_speed/         ↙
    count_e_coli, Apreservation/count_e_coli, ApreservationSteps/count_e_coli);
      if ( (step % (STEP_TO_HOURS_L * 24) == 0) || (step <= 12 * STEP_TO_HOURS_L) ) /*one ↙
    day*/
      {
      /*  sprintf(map_energy_name,"E_Coli_EnergyMap_%ld_hours", step);
        sprintf(map_count_name,"E_Coli_countMap_%ld_hours", step);
        map_energy = fopen(map_energy_name, "w");
        map_count = fopen(map_count_name, "w");
        for (x = MIN_AREA_X; x < MAX_AREA_X; x++)
         for (y = MIN_AREA_Y; y < MAX_AREA_Y; y++)
         {
             area_count[x][y] = 0;
             area_energy[x][y] = 0;
         }
        for(e_coli_i = 0; e_coli_i < count_e_coli; e_coli_i++)
         {
          area_count[coli[e_coli_i].x][coli[e_coli_i].y]++;
          area_energy[coli[e_coli_i].x][coli[e_coli_i].y] += coli[e_coli_i].energy;
         }
        for (x = MIN_AREA_X; x < MAX_AREA_X; x++)
         for (y = MIN_AREA_Y; y < MAX_AREA_Y; y++)
         {
              if (area_energy[x][y]!= 0)
                 fprintf(map_energy,"%ld\t%ld\t%lf\n", x, y, area_energy[x][y]);
              if (area_count[x][y]!= 0)
                 fprintf(map_count,"%ld\t%ld\t%ld\n", x, y, area_count[x][y]);
         }
        fclose(map_count);
        fclose(map_energy);*/
      }
      #endif
      avg_death = 0;avg_born=0;
      }
      move_all(&idum.angle, &idum.radius, idum);
      if (count_e_coli == 0) break;
    }
    #ifdef USE_MPI
    #else
    fclose(out);
    #endif
    #ifdef VERBOSE
      #ifdef USE_MPI
      if (myid == 0)
      {
      #endif
        init_process("Program completed");
        complete_process();
      #ifdef USE_MPI
      }
      #endif
    #endif
    stop_pr();
    #ifdef USE_MPI
    MPI_Reduce(&current_mpi_step, &mpi_step, 1, MPI_LONG, MPI_MAX, 0, MPI_COMM_WORLD);
    MPI_Reduce(count_e_coli_sum, count_e_coli_sum_mpi, MAX_HOURS, MPI_LONG, MPI_SUM, 0,   ↙
    MPI_COMM_WORLD);
    MPI_Reduce(avg_energy_sum, avg_energy_sum_mpi, MAX_HOURS, MPI_DOUBLE, MPI_SUM, 0,     ↙
    MPI_COMM_WORLD);
    MPI_Reduce(death_e_coli_sum, death_e_coli_sum_mpi, MAX_HOURS, MPI_LONG, MPI_SUM, 0,   ↙
    MPI_COMM_WORLD);
    MPI_Reduce(total_energy_sum, total_energy_sum_mpi, MAX_HOURS, MPI_DOUBLE, MPI_SUM, 0, ↙
    MPI_COMM_WORLD);

    out = fopen("E coli output mpi.txt", "w");
    for(mpi_loop = 0; mpi_loop < mpi_step; mpi_loop++)
     fprintf(out,"%ld\t%.0lf\t%.0lf\t%.3e\t%.3e\n", mpi loop, count e coli sum mpi ↙
    [mpi loop]/numprocs, death e coli sum_mpi[mpi_loop]/numprocs, avg_energy_sum_mpi/ ↙
    numprocs, total_energy_sum/numprocs);
    fclose(out);
      #ifdef VERBOSE
        printf("Processor %ld finnished!\n", myid);
      #endif
    MPI Finalize();
    #endif
    return 0;
}
```

File: SCC_structs_v_1_1.c

```c
/*If the energy is more than E_DIV the _e_coli can divide*/
/*E_DIV scaled= 110M *3/100 = 3.3M*/
#define E_DIV 20000
/*E_MIN scaled= 40M *3/100 = 1.2M*/
#define E_MIN 12000
/*E_MAX scaled= 60M *3/100 = 1.8M*/
#define E_MAX 18000
/*E_GAIN scaled= 1300 *3/100 = 29*/
#define E_GAIN 330
/*E_LOSS_PER_STEP scaled= 33 *3/100 = 1*/
#define E_LOSS_PER_STEP 15
#define E_CONST_LOSS_PER_STEP 0
/*Under E_DEATH the _e_coli dies*/
/*E_DEATH scaled= 30M *3/100 = 900K*/
#define E_DEATH 3750
/*E_PER_PLACE scaled= 55G *3/100 = 1.65G*/
#define E_PER_PLACE 4000
/*The number pf steps to divide the e_coli about 20 mins*/
#define STEP_FOR_DIVISION 19
static double STEP_TO_HOURS = 3 * STEP_FOR_DIVISION;
static long STEP_TO_HOURS_L;
#define PLACE_DENSITY 0.85
#define PRESERVATION_OF_ENERGY_STEPS 150
#define PRESERVATION 0.001
#define min_e_coli_speed 1 /*in steps, one step 20 micro meters*/
#define max_e_coli_speed 4 /*in steps aka 80 micro meters, one step 20 micro meters */
#define min_e_coli_flagellas 5
#define max_e_coli_flagellas 20
#define min_length_flagella 5
#define max_length_flagella 15
#define max_e_coli_speed_of_eating 25 /* times per sec*/
#define min_e_coli_speed_of_eating 15 /* times per sec*/
#define eats_ATP_per_sec 35
#define max_e_coli_percent_transform 90 /* times per sec*/
#define min_e_coli_percent_transform 70 /* times per sec*/
#define GRAVITY 1
#define MAX_NUMBER_E_COLI 20000001
#define MAX_AREA_X 10000
#define MAX_AREA_Y 10000
#define MIN_AREA_X 0
#define MIN_AREA_Y 0
#define MAX_ANGLES 360


static double interval_energy = E_MAX - E_MIN;
static double revGravity = 1 / GRAVITY;
static int interval_e_colli_speed = max_e_coli_speed - min_e_coli_speed;
static double interval_flagellas = max_e_coli_flagellas - min_e_coli_flagellas;
static double interval_length_flagellas = max_length_flagella - min_length_flagella;

int calc_e_coli_max_speed(struct _e_coli e)
{
    return revGravity *(min_e_coli_speed + (int)((e.number_of_flagellas -                ↙
    min_e_coli_flagellas)/interval_flagellas*(e.avg_length_flagella-min_length_flagella)/ ↙
    interval_length_flagellas*(double)interval_e_colli_speed));
}


struct _e_coli init_e_coli(struct _e_coli new_e_coli, struct _idum idum)
{
       do/*the speed of the e_coli cannot be less than zero*/
       {
        do
        {
          new_e_coli.number_of_flagellas = (short)arb_gasdev_d(max_e_coli_flagellas,       ↙
    (long)(sqrt(max_e_coli_flagellas)), &idum.number_of_flagellas) ;
        }
        while((new_e_coli.number_of_flagellas < 0)||(new_e_coli.number_of_flagellas > 2 * ↙
    max_e_coli_flagellas));

        do
        {
          new_e_coli.avg_length_flagella = arb_gasdev_d(max_length_flagella, sqrt          ↙
    (max_length_flagella), &idum.avg_length_flagella);
        }
        while((new_e_coli.avg_length_flagella < 0)||(new_e_coli.avg_length_flagella > 2 * ↙
    max_length_flagella));

        do
        {
          new_e_coli.speed_of_eating = arb_gasdev_d(max_e_coli_speed_of_eating, sqrt      ↙
    (max_e_coli_speed_of_eating), &idum.speed_of_eating);
        }
        while((new_e_coli.speed_of_eating < 0)||(new_e_coli.speed_of_eating > 2 *          ↙
    max_e_coli_speed_of_eating));

        new_e_coli.max_speed = calc_e_coli_max_speed(new_e_coli);
       }
       while(new_e_coli.max_speed < 0);


       do
       {
         new_e_coli.energy = arb_gasdev_d(E_MAX, sqrt(E_MAX), &idum.energy);
       }
       while((new_e_coli.energy < E_MIN)||(new_e_coli.energy > 2 * E_MAX - E_MIN));

       new_e_coli.steps = 0;
       new_e_coli.energy_status = 0;

       do
       {
          new_e_coli.preservation = fabs(arb_gasdev_d(PRESERVATION, sqrt(PRESERVATION), & ↙
    idum.preservation));
       }
       while((new_e_coli.preservation < 0)||(new_e_coli.preservation > 2 * PRESERVATION));

       do
       {
         new_e_coli.preservationSteps = arb_gasdev_d(PRESERVATION_OF_ENERGY_STEPS, sqrt   ↙
    (PRESERVATION_OF_ENERGY_STEPS), &idum.preservationSteps);
       }
       while((new_e_coli.preservationSteps < 0)||(new_e_coli.preservationSteps > 2 *      ↙
    PRESERVATION_OF_ENERGY_STEPS));

       new_e_coli.x = (int)( (MAX_AREA_X - MIN_AREA_X)  * ran2(&idum.x) ) + MIN_AREA_X;
       new_e_coli.y = (int)( (MAX_AREA_Y - MIN_AREA_Y)  * ran2(&idum.y) ) + MIN_AREA_Y;
       return new_e_coli;
}


double e_coli_jump_energy_loss(double jump)
{
       return GRAVITY*( E_CONST_LOSS_PER_STEP + E_LOSS_PER_STEP * jump);
}

int calc_e_coli_status(double energy)/*returns 1 if the e_coli will die, returns 2 if the ↙
    e-coli can reproduce, returns 0 if the e_coli is regular*/
{
    if (energy < E_DEATH) return 1;
    if (energy > E_DIV) return 2;
    return 0;
}
```

File: e_coli.c

```
struct __cos_sin
{
       double cos, sin;
};

static struct __cos_sin _cos_sin[MAX_ANGLES];
static long area[MAX_AREA_X][MAX_AREA_Y];
/*static long area_count[MAX_AREA_X][MAX_AREA_Y];
static double area_energy[MAX_AREA_X][MAX_AREA_Y];*/
static struct _e_coli *coli;
static long count_e_coli, pres_number;
static long  count_death_e_coli;
static double avg_e_coli_energy;
static double avg_born;
static double avg_death;
static double total_food;


void create_initial(long numer_e_coli, struct _idum idum)
{
    long i, a;
    #ifdef VERBOSE
        #ifdef USE_MPI
        if (myid == 0)
        {
       #endif
         init_process("Initilzing the e_colis");
       #ifdef USE_MPI
        }
       #endif
    #endif
    #ifdef HANDLE_ERRORS
     if (numer_e_coli >= MAX_NUMBER_E_COLI)
        Error(__LINE__, __FILE__, "The number of e_coli you want to create must be less ↙
    than MAX_NUMBER_E_COLI");
    #endif

    coli = (struct _e_coli *) malloc ( MAX_NUMBER_E_COLI * sizeof(struct _e_coli));/*   ↙
    allocates memory for the e_coli*/
    if (coli == NULL)/*if there is not enough memory the program exists*/
    {
     #ifdef HANDLE_ERRORS
     Error(__LINE__, __FILE__, "There is not enough memory to create all the e_coli");
     #else
     fprintf(stderr, "There is not enough memory to create all the e_coli");
     stop_pr();
     exit(0);
     #endif
    }
     for(a = 0; a < MAX_ANGLES; a++)/*initilizing sin and cos*/
     {
        _cos_sin[a].cos = cos(rad * a);
       _cos_sin[a].sin = sin(rad * a);
     }

     count_e_coli = numer_e_coli;
     count_death_e_coli = 0;
     avg_e_coli_energy = 0;
     for(i = 0; i < numer_e_coli; i++)/*creating all the e_colies*/
     {
       coli[i] = init_e_coli(coli[i], idum);
     }
    #ifdef VERBOSE
        #ifdef USE_MPI
        if (myid == 0)
        {
        #endif
```

```
         complete_process();
        #ifdef USE_MPI
        }
        #endif
    #endif
}

void init_area(long *density, long *area_idum)
{
     double a;
     int x, y;
     #ifdef VERBOSE
        #ifdef USE_MPI
        if (myid == 0)
        {
        #endif
         init_process("Initilizing the area");
        #ifdef USE_MPI
        }
        #endif
     #endif
     total_food = 0;
     for (x = MIN_AREA_X; x < MAX_AREA_X; x++)
         for (y = MIN_AREA_Y; y < MAX_AREA_Y; y++)
         {
             if ( ran2(density) > PLACE_DENSITY ) area[x][y] = 0;
                else
                {
                  area[x][y] = arb_gasdev_l(E_PER_PLACE, (long)sqrt(E_PER_PLACE),      ↙
    area_idum);
                  total_food = total_food + (double)area[x][y];
                }
         }
     #ifdef VERBOSE
        #ifdef USE_MPI
        if (myid == 0)
        {
        #endif
         complete_process();
         printf("The total food is %e\n", total_food);
        #ifdef USE_MPI
        }
        #endif
     #endif
}

void move_e_coli(int a, int b)
{
     coli[a].number_of_flagellas = coli[b].number_of_flagellas;
     coli[a].avg_length_flagella = coli[b].avg_length_flagella;
     coli[a].speed_of_eating = coli[b].speed_of_eating;
     coli[a].max_speed = coli[b].max_speed;
     coli[a].energy = coli[b].energy;
     coli[a].x = coli[b].x;
     coli[a].y = coli[b].y;
     coli[a].steps = coli[b].steps;
     coli[a].energy_status = coli[b].energy_status;
     coli[a].preservation= coli[b].preservation;
}

void make_new_e_coli(int new_e_coli, int original, struct _idum idum)
{
     /*the old e_coli*/
     coli[original].energy = .5 * coli[original].energy;
     coli[original].steps = 0;
     coli[new_e_coli].energy_status = 0;
     /*the new e_coli*/
     do
     {
      do
      {
        coli[new_e_coli].number_of_flagellas = (short)arb_gasdev_d(coli[original].        ↙
    number_of_flagellas, (long)(sqrt(coli[original].number_of_flagellas)), &idum.         ↙
    number_of_flagellas);
      }
      while((coli[new_e_coli].number_of_flagellas < 0)||(coli[new_e_coli].               ↙
    number_of_flagellas > 2 * max_e_coli_flagellas));

      do
      {
        coli[new_e_coli].avg_length_flagella = arb_gasdev_d(coli[original].              ↙
    avg_length_flagella, sqrt(coli[original].number_of_flagellas), &idum.                 ↙
    avg_length_flagella);
      }
      while((coli[new_e_coli].avg_length_flagella < 0)||(coli[new_e_coli].               ↙
    avg_length_flagella > 2 * max_length_flagella));

      do
      {
        coli[new_e_coli].speed_of_eating = arb_gasdev_d(coli[original].speed_of_eating,   ↙
    sqrt(coli[original].speed_of_eating), &idum.speed_of_eating);
      }
      while((coli[new_e_coli].speed_of_eating < 0)||(coli[new_e_coli].speed_of_eating > 2 ↙
    * max_e_coli_speed_of_eating));

     coli[new_e_coli].max_speed = calc_e_coli_max_speed(coli[new_e_coli]);
     }
     while(coli[new_e_coli].max_speed<0);

     coli[new_e_coli].energy = coli[original].energy;
     coli[new_e_coli].x = coli[original].x;
     coli[new_e_coli].y = coli[original].y;
     coli[new_e_coli].steps = 0;
     coli[new_e_coli].energy_status = 0;
     do
     {
       coli[new_e_coli].preservation = fabs(arb_gasdev_d(coli[original].preservation, sqrt↙
     (coli[original].preservation), &idum.preservation));
     }
     while((coli[new_e_coli].preservation < 0)||(coli[new_e_coli].preservation > 2 *      ↙
    PRESERVATION));

     do
     {
       coli[new_e_coli].preservationSteps = abs(arb_gasdev_d(coli[original].             ↙
    preservationSteps, sqrt(coli[original].preservationSteps), &idum.preservationSteps));
     }
     while((coli[new_e_coli].preservationSteps < 0)||(coli[new_e_coli].preservationSteps >↙
     2 * PRESERVATION_OF_ENERGY_STEPS));


}

void eat(int number)
{
    if (area[coli[number].x][coli[number].y] == 0) return;
     if (area[coli[number].x][coli[number].y] > E_GAIN)
     {
       coli[number].energy += E_GAIN;
       area[coli[number].x][coli[number].y] -= E_GAIN;
       total_food -= (double)E_GAIN;
     }
     else
     {
```

```c
      coli[number].energy += area[coli[number].x][coli[number].y];
      total_food -= (double)area[coli[number].x][coli[number].y];
      area[coli[number].x][coli[number].y] = 0;
     }
}

void move_single(int number, long *idum_angle, long *idum_r)
{
     int x, y, fi;
     double val;
     double r = coli[number].max_speed;
     r = arb_gasdev_d(r, sqrt(r), idum_r);
     fi = (int)(360 * ran2(idum_angle) );
     x = (int) (_cos_sin[fi].cos * r);
     y = (int) (_cos_sin[fi].sin * r);
     coli[number].x += x;
     /*Limited Boundary Conditions for X*/
     if (coli[number].x >= MAX_AREA_X)
        coli[number].x = MAX_AREA_X - 1;
     if (coli[number].x < MIN_AREA_X)
        coli[number].x = MIN_AREA_X;

     coli[number].y += y;
     /*Limited Boundary Conditions for Y*/
     if (coli[number].y >= MAX_AREA_Y)
        coli[number].y = MAX_AREA_Y - 1;
      if (coli[number].y < MIN_AREA_Y)
        coli[number].y = MIN_AREA_Y;
    if  (coli[number].energy_status >= coli[number].preservationSteps)
    {
    /* val = coli[number].energy_status / PRESERVATION_OF_ENERGY_STEPS;
     if (val > 1) val = 1;
     val = 1- val;*/
     coli[number].energy -= ( coli[number].preservation* e_coli_jump_energy_loss( fabs(r) ↙
     ) );
     pres_number++;
    }
    else coli[number].energy -= e_coli_jump_energy_loss(fabs(r));
}

void move_all(long *idum_angle, long *idum_r, struct _idum idum)
{
     int i;
     int status;
     double e;
     avg_e_coli_energy = 0;
     pres_number = 0;
     Anumber_of_flagella  = 0;
     Aavg_length_flagella = 0;
     Aspeed_of_eating     = 0;
     Amax_speed           = 0;
     Apreservation        = 0;
     ApreservationSteps   = 0;
     for(i = 0; i < count_e_coli; i++)
     {
           status = calc_e_coli_status(coli[i].energy);
           if ( status == 1 )
           {/*kills the e-coli*/
              move_e_coli(i, count_e_coli - 1);
              count_death_e_coli++;
              i--;
              count_e_coli--;
              avg_death++;
           }
           else
           {
               /* average calculations */
```

```
                  Anumber_of_flagella  += coli[i].number_of_flagellas;
                  Aavg_length_flagella += coli[i].avg_length_flagella;
                  Aspeed_of_eating     += coli[i].speed_of_eating;
                  Amax_speed           += coli[i].max_speed;
                  Apreservation        += coli[i].preservation;
                  ApreservationSteps   += coli[i].preservationSteps;
               /* average calculations */
             if ( (coli[i].steps >= STEP_FOR_DIVISION) && (coli[i].energy_status < coli[i]↙
    .preservationSteps))/*check the division steps aka time*/
             {
              if (status == 2)/*check division status aka energy*/
              {
               make_new_e_coli(count_e_coli, i, idum);/*divides*/
               count_e_coli++;
               avg_born++;
              }
             }
             else
             {
               coli[i].steps++;
             }
             e = coli[i].energy;
             move_single(i, idum_angle, idum_r);
             eat(i);
             if (e >= coli[i].energy)/*preservation*/
                coli[i].energy_status++;
             else coli[i].energy_status = 0;
             avg_e_coli_energy += coli[i].energy;
           }
     }
     if (count_e_coli == 0) avg_e_coli_energy = 0;
        else avg_e_coli_energy /= count_e_coli;
}
```

File: general.c

```
#define PI 3.14159265358979323846264338327950
#define rad 0.01745329251994329576923690768488 6

void stop_pr()
{
     system("Pause");
}

#ifdef HANDLE ERRORS
void Error(int line, char file[], char message[])
{
     char error[300], stop;
     #ifdef USE_MPI
     sprintf(error,"Runtime error occured in myid %ld!\nFile: %s\nLine#: %ld\nError 
    Message: %s", myid, file, line, message);
     #else
     sprintf(error,"Runtime error occured!\nFile: %s\nLine#: %ld\nError Message: %s", file
    , line, message);
     #endif
     fprintf(stderr, "%s\n", error);
     stop pr();
     exit(0);
}
#endif

#ifdef VERBOSE
static int init_process_count = 0;

void init_process(char message[])
{
 init process count++;
 fprintf(stderr,"%s...", message);
}

void complete_process()
{
 init process count--;
 fprintf(stderr,"Completed!\n");
}
#endif

long arb_gasdev_l(long avg, long std, long *idum)
{
     return ((long)(std * gasdev(idum)) + avg);
}

double arb_gasdev_d(double avg, double std, long *idum)
{
     return std * gasdev(idum) + avg;
}

struct _idum init_random_numbers()
{
       FILE *open;
       int i;
       struct  idum current;
       open = fopen("random_numbers.txt", "r");
      #ifdef USE MPI
        for(i = 0; i <= myid; i++)
        {
          fscanf(open, "%ld", &current.init);
          fscanf(open, "%ld", &current.density);
          fscanf(open, "%ld", &current.radius);
          fscanf(open, "%ld", &current.area);
          fscanf(open, "%ld", &current.avg length_flagella);
          fscanf(open, "%ld", &current.energy);
          fscanf(open, "%ld", &current.number_of_flagellas);
          fscanf(open, "%ld", &current.preservation);
          fscanf(open, "%ld", &current.preservationSteps);
          fscanf(open, "%ld", &current.speed_of_eating);
          fscanf(open, "%ld", &current.x);
          fscanf(open, "%ld", &current.y);
        }
        #ifdef VERBOSE
          printf("Processor %ld idum's are:\n", myid);
          printf("%ld\t%ld\t%ld\t%ld\t%ld\t%ld\t%ld\t%ld\t%ld\t%ld\t%ld\t%ld\n", current. ↙
    init, current.density, current.radius, current.area, current.avg_length_flagella,     ↙
    current.energy, current.number_of_flagellas, current.preservation, current.           ↙
    preservationSteps, current.speed_of_eating, current.x, current.y);
        #endif
      #else
          fscanf(open, "%ld", &current.init);
          fscanf(open, "%ld", &current.density);
          fscanf(open, "%ld", &current.radius);
          fscanf(open, "%ld", &current.area);
          fscanf(open, "%ld", &current.avg_length_flagella);
          fscanf(open, "%ld", &current.energy);
          fscanf(open, "%ld", &current.number_of_flagellas);
          fscanf(open, "%ld", &current.preservation);
          fscanf(open, "%ld", &current.preservationSteps);
          fscanf(open, "%ld", &current.speed_of_eating);
          fscanf(open, "%ld", &current.x);
          fscanf(open, "%ld", &current.y);
          #ifdef VERBOSE
           printf("The idum's are:\n");
           printf("%ld\t%ld\t%ld\t%ld\t%ld\t%ld\t%ld\t%ld\t%ld\t%ld\t%ld\t%ld\n", current.↙
    init, current.density, current.radius, current.area, current.avg_length_flagella,     ↙
    current.energy, current.number_of_flagellas, current.preservation, current.           ↙
    preservationSteps, current.speed_of_eating, current.x, current.y);
          #endif
      #endif
      fclose(open);
return current;
}


void print_e_coli(struct _e_coli ec)
{
     int energy_status;
     double preservation;
     long preservationSteps;
     fprintf(stderr, "The ecoli is at X: %ld and Y: %ld\n", ec.x, ec.y);
     fprintf(stderr, "The ecoli has:\n");
     fprintf(stderr, "Number of flagellas: %ld\n", ec.number_of_flagellas);
     fprintf(stderr, "Average length of flagella: %lf\n", ec.avg_length_flagella);
     fprintf(stderr, "Spead of eating: %lf\n", ec.speed_of_eating);
     fprintf(stderr, "Max speed: %ld\n", ec.max_speed);
     fprintf(stderr, "Energy: %lf\n", ec.energy);
     fprintf(stderr, "Steps made after last division: %ld\n", ec.steps);
     fprintf(stderr, "Energy status: %ld\n", ec.energy_status);
     fprintf(stderr, "Preservation: %lf\n", ec.preservation);
     fprintf(stderr, "Preservation Steps: %ld\n", ec.preservationSteps);

}
```

File: gasdev.c

```
C:\Documents and Settings\All Users\Desktop\E_coli\gasdev.c                                         1

double gasdev(long *idum)
{
    static int iset=0;
    static double gset;
    double fac,rsq,v1,v2;

    if (*idum < 0) iset=0;
    if  (iset == 0) {
        do {
            v1=2.0*ran2(idum)-1.0;
            v2=2.0*ran2(idum)-1.0;
            rsq=v1*v1+v2*v2;
        } while (rsq >= 1.0 || rsq == 0.0);
        fac=sqrt(-2.0*log(rsq)/rsq);
        gset=v1*fac;
        iset=1;

        return v2*fac;
    } else {
        iset=0;

        return gset;
    }
}
```

File: ran2.c

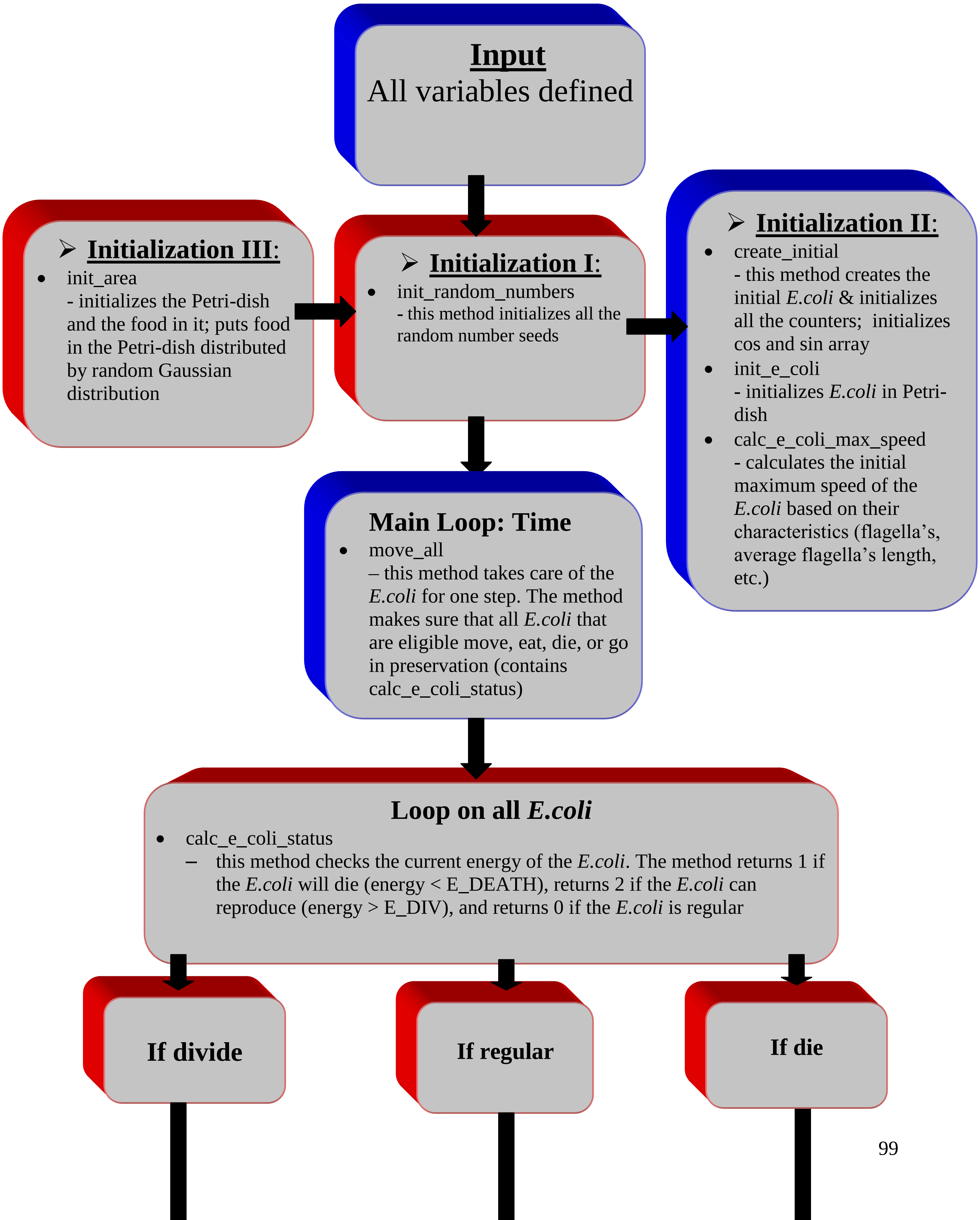
```
#define IM1 2147483563
#define IM2 2147483399
#define AM (1.0/IM1)
#define IMM1 (IM1-1)
#define IA1 40014
#define IA2 40692
#define IQ1 53668
#define IQ2 52774
#define IR1 12211
#define IR2 3791
#define NTAB 32
#define NDIV (1+IMM1/NTAB)
#define EPS 1.2e-7
#define RNMX (1.0-EPS)

float ran2(long *idum)
{
    int j;
    long k;
    static long idum2=123456789;
    static long iy=0;
        static long iv[NTAB];
    float temp;

    if (*idum <= 0) {
        if (-(*idum) < 1) *idum=1;
        else *idum = -(*idum);
        idum2=(*idum);
        for (j=NTAB+7;j>=0;j--) {
            k=(*idum)/IQ1;
            *idum=IA1*(*idum-k*IQ1)-k*IR1;
            if (*idum < 0) *idum += IM1;
            if (j < NTAB) iv[j] = *idum;
        }
        iy=iv[0];
    }
    k=(*idum)/IQ1;
    *idum=IA1*(*idum-k*IQ1)-k*IR1;
    if (*idum < 0) *idum += IM1;
    k=idum2/IQ2;
    idum2=IA2*(idum2-k*IQ2)-k*IR2;
    if (idum2 < 0) idum2 += IM2;
    j=iy/NDIV;
    iy=iv[j]-idum2;
    iv[j] = *idum;
    if (iy < 1) iy += IMM1;
    if ((temp=AM*iy) > RNMX) return RNMX;
    else return temp;
}
#undef IM1
#undef IM2
#undef AM
#undef IMM1
#undef IA1
#undef IA2
#undef IQ1
#undef IQ2
#undef IR1
#undef IR2
#undef NTAB
#undef NDIV
#undef EPS
#undef RNMX
```

Appendix C

Flow Chart



Input

All variables defined

➢ **Initialization I:**

- init_random_numbers
  - this method initializes all the random number seeds

➢ **Initialization III:**

- init_area
  - initializes the Petri-dish and the food in it; puts food in the Petri-dish distributed by random Gaussian distribution

➢ **Initialization II:**

- create_initial
  - this method creates the initial *E.coli* & initializes all the counters; initializes cos and sin array
- init_e_coli
  - initializes *E.coli* in Petri-dish
- calc_e_coli_max_speed
  - calculates the initial maximum speed of the *E.coli* based on their characteristics (flagella's, average flagella's length, etc.)

Main Loop: Time

- move_all
  – this method takes care of the *E.coli* for one step. The method makes sure that all *E.coli* that are eligible move, eat, die, or go in preservation (contains calc_e_coli_status)

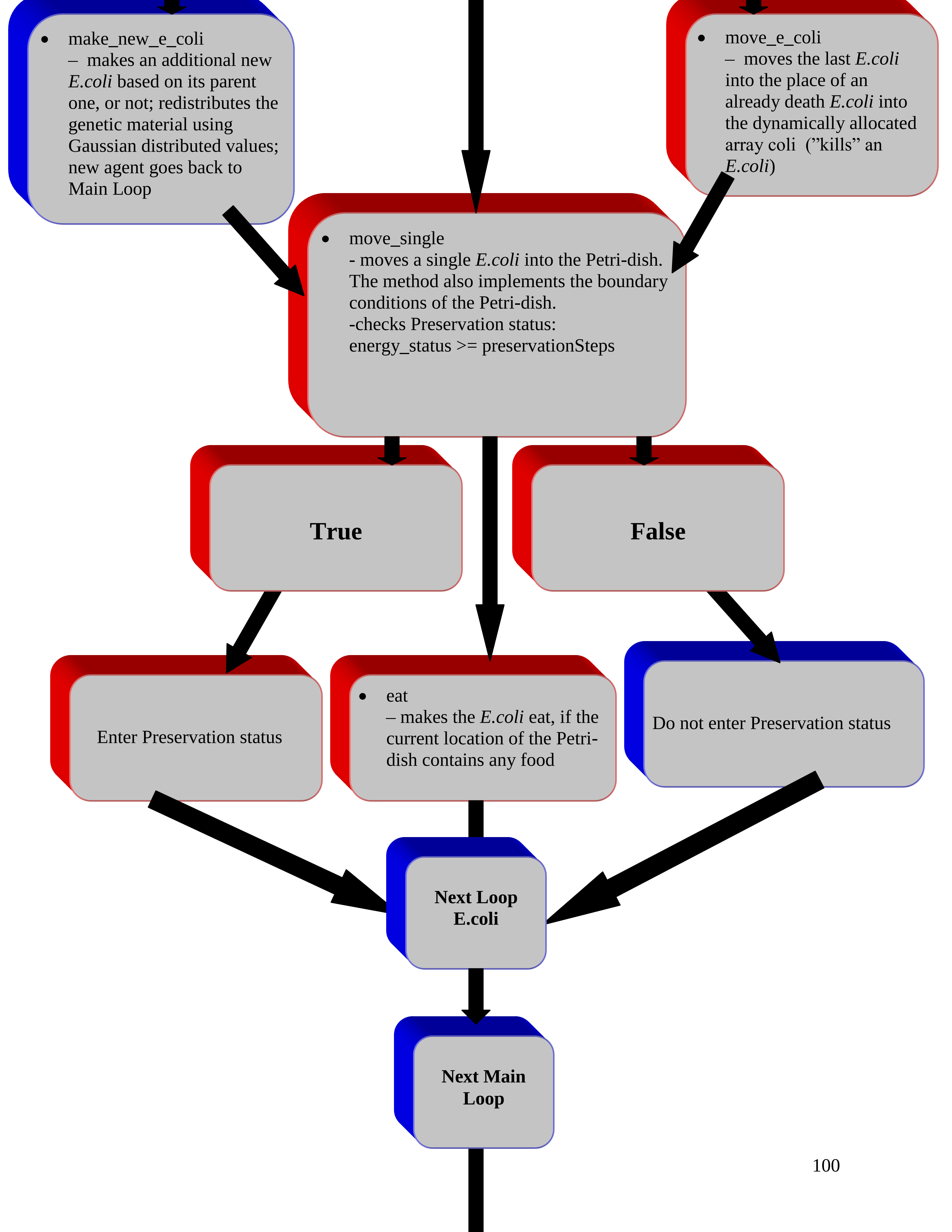
Loop on all *E.coli*

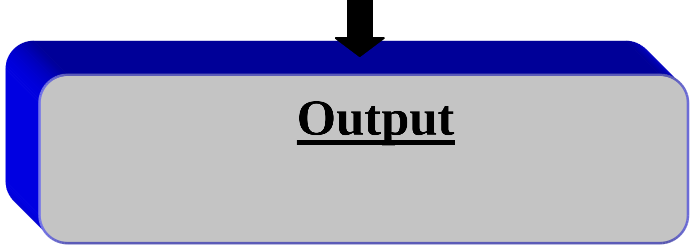
- calc_e_coli_status
  – this method checks the current energy of the *E.coli*. The method returns 1 if the *E.coli* will die (energy < E_DEATH), returns 2 if the *E.coli* can reproduce (energy > E_DIV), and returns 0 if the *E.coli* is regular

If divide

If regular

If die


• make_new_e_coli
– makes an additional new *E.coli* based on its parent one, or not; redistributes the genetic material using Gaussian distributed values; new agent goes back to Main Loop
• move_e_coli
– moves the last *E.coli* into the place of an already death *E.coli* into the dynamically allocated array coli ("kills" an *E.coli*)
• move_single
- moves a single *E.coli* into the Petri-dish. The method also implements the boundary conditions of the Petri-dish.
-checks Preservation status:
energy_status >= preservationSteps
True
False
Enter Preservation status
• eat
– makes the *E.coli* eat, if the current location of the Petri-dish contains any food
Do not enter Preservation status
Next Loop E.coli
Next Main Loop


Output