



**POLITÉCNICA**

"Ingeniamos el futuro"

CAMPUS  
DE EXCELENCIA  
INTERNACIONAL



# **Graduado en Matemáticas e Informática**

Universidad Politécnica de Madrid

Escuela Técnica Superior de  
Ingenieros Informáticos

## **TRABAJO FIN DE GRADO**

Programación del crecimiento de bacterias E.coli en  
el simulador BactoSIM

Autor: Carolina Alonso Merino

Director: Alfonso Rodríguez-Patón Aradas

MADRID, JULIO 2018



---

A mis padres,  
por enseñarme a superar  
las situaciones más difíciles de la vida.  
Os quiero.

---

# Índice

Resumen.....	4
Abstract .....	5
1. INTRODUCCIÓN .....	6
1.1. Estructura del Documento .....	8
2. ESTADO DEL ARTE .....	9
2.1. GRO .....	11
2.2. BactoSIM .....	11
2.3. CellEngine.....	12
3. PLANTEAMIENTO DEL PROBLEMA .....	13
3.1. Solución actual .....	14
3.2. Análisis del motor físico actual.....	15
4. SOLUCIÓN PROPUESTA .....	16
4.1. Explicación de CellEngine .....	16
4.2. Detalle del algoritmo de CellEngine .....	18
4.3. Construcción del nuevo motor para BactoSim .....	19
5. RESULTADOS Y VALIDACIÓN .....	37
6. CONCLUSIONES Y FUTURAS LINEAS DE TRABAJO .....	40
6.1. Conclusiones.....	40
6.2. Futuras líneas de trabajo.....	41
Anexos.....	42
Bibliografía .....	43

## Resumen

En la actualidad se han desarrollado aplicaciones que permiten simular el comportamiento de las colonias formadas por bacterias. Una de las partes más importantes de estos simuladores es el motor físico, ya que es el encargado de resolver todas las fuerzas producidas entre las bacterias y conseguir que todas queden correctamente distribuidas en la colonia. Cuanto mayor potencia tenga el motor, la simulación será mejor y podremos conseguir una mejor aproximación a la realidad.

En el simulador BactoSIM, no existe un motor que haga esta función tan eficientemente por lo que el objetivo principal de este proyecto es la incorporación de un motor físico en él. El motor físico que se va a introducir es basado en el motor que posee actualmente el simulado GRO, CellEngine. Este es un motor que se centra en la simulación rápida de colonias bacterianas en crecimiento. En este proyecto veremos detalladamente la implantación del nuevo motor para el simulador BactoSIM.

## Abstract

Nowadays, there are applications that allow us to simulate the behaviour of colonies formed by bacteria. One of the most important parts of these simulators is the physical engine. Since it oversees solving all the forces produced by the bacteria and ensuring that all are correctly distributed in the colony. Thus, we get a better approximation to reality. The more power the motor has, the simulation will be better and easier to achieve than a better approximation to reality.

In the BactoSim simulator, there is no engine that performs this function so efficiently, so the main objective of this project is the incorporation of a physical engine in it. The physical engine that we are going to introduce is based on the engine which the GRO simulator owns, CellEngine, an engine focused on the fast simulation of growing bacterial colonies. This is an engine that focuses on rapid simulation of growing bacterial colonies. In this project, we will see in detail the implementation of the new engine for the BactoSIM simulator.

## 1. INTRODUCCIÓN

La biología sintética[1] se define como: *“Una disciplina emergente que utiliza principios de ingeniería para diseñar y ensamblar componentes biológicos.”*

La biología sintética [2] es una rama interdisciplinaria de la biología y la ingeniería. El sujeto combina diversas disciplinas de dentro de estos dominios, tales como la biotecnología, la ingeniería genética, la biología molecular, la ingeniería molecular, la biología de sistemas, la biofísica, la ingeniería eléctrica, la ingeniería informática, ingeniería de control y biología evolutiva. La biología sintética aplica estas disciplinas para construir sistemas biológicos artificiales para investigación, ingeniería y aplicaciones médicas.

Un simulador [3] se define como: *“Dispositivo o aparato que simula un fenómeno, el funcionamiento real de otro aparato o dispositivo o las condiciones de entorno a las que están sometidos una máquina, aparato o material.”*

Los simuladores [4] permiten la reproducción de un sistema, desde sensaciones físicas hasta comportamientos de equipos. Actualmente, se utilizan simuladores continuamente para entrenar a profesionales de distintos campos como a los pilotos o astronautas. Sin embargo, este proyecto se centrará más en los simuladores que reproducen el comportamiento de equipos. Estos simuladores se basan en modelos matemáticos para intentar imitar la realidad y poder ponerlos a prueba en varias situaciones.

Dentro de los simuladores que simulan el comportamiento de equipos, se encuentran los simuladores de bacterias. Éstos simuladores intentan reproducir como se comportan las bacterias en una colonia centrándose en crecimiento y reacciones a sustancias químicas. Entre estos simuladores existen varios tipos en relación con la forma de la bacteria, los que las ilustran en forma de bacilo (capsular) o en forma de coco (esférica). Dado que este proyecto se centra en la bacteria E.coli, la cual tiene

forma capsular, se usará un simulador que contemple esta forma.

BactoSim es un simulador enfocado a la conjugación y a la propagación del código genético a través de la colonia. Las células se modelan como círculos sobre una malla discreta con el consiguiente ahorro de cálculo, permitiendo observar la propagación del código genético a través de la colonia. Se considera una herramienta de prototipado. A este simulador le vamos a incorporar un motor físico, basado en CellEngine, que se centra en la simulación rápida de colonias bacterianas en crecimiento. Esta incorporación es interesante porque vamos a hacer que las simulaciones que BactoSim genera sean más eficientes y vamos a reemplazar ese espacio discretizado por un espacio continuo.



## 1.1. Estructura del Documento

- **Estado del arte.** Se explica la importancia que tienen los simuladores de colonias bacterianas y se muestran varios simuladores que están actualmente en uso.
- **Planteamiento del problema.** Inicialmente se explica cómo funciona de manera general los motores de física de cuerpos sólidos. En este apartado también se detalla cómo funciona en la actualidad el simulador BactoSim, así como su motor físico.
- **Solución propuesta.** Incorporación del nuevo motor físico en BactoSim. Explicación sobre el algoritmo que tiene CellEngine y como nos basamos en este para la obtención del nuevo motor de BactoSim.
- **Resultados y comparativas.** En este apartado se muestran los resultados del nuevo algoritmo, sus ventajas y desventajas.
- **Conclusiones y futuros trabajos.** Se realizará un repaso a lo investigado durante el documento, haciendo hincapié en los resultados conseguidos. Se explicará de forma breve futuras mejoras.

## 2. ESTADO DEL ARTE

La biología sintética [5,6] es una ciencia relativamente moderna, ya que hasta hace unos pocos años no se conocía bien la base para el diseño de nuevos sistemas biológicos. Es por ello, que el uso de simuladores de colonias de bacterias no ha tenido una aplicación funcional hasta estos últimos años. Estos simuladores son los encargados de simular la interacción que se produce tanto entre las propias bacterias como entre las bacterias y las señales del medio. De esa manera, el biólogo puede realizar experimentos orientativos acerca de cómo afecta una determinada sustancia a un conjunto de bacterias o de cómo se propaga cierta señal entre ellas.

El tiempo necesario para permitir que una colonia crezca en una placa de Petri o *biofilm* [7] hasta el tamaño deseado para un experimento suele llevar entre 24 y 48 horas en un laboratorio. Muchas de las pruebas que se efectúan sobre una colonia la dejan inservible para otra posible prueba, y para realizar una investigación cualquiera son necesarios cientos de experimentos como estos. Por tanto, una herramienta que simule dichos experimentos y que reduzca drásticamente ese tiempo es crucial para el desarrollo de cualquier investigación. Obviamente, los resultados de una colonia simulada no son tan reales como los realizados sobre una placa de Petri, pero sí que permite al biólogo descartar ciertos resultados claramente negativos en la búsqueda de su objetivo.

A estos simuladores también se los llama, modelos basados en individuos [8], ya que lo que se hace realmente es programar tanto la acción como la interacción a nivel de individuo. Esto entre la totalidad del conjunto de los individuos genera la colonia con sus características, características que a priori son desconocidas y no se han programado directamente.

Una de esas interacciones es la conjugación [9] cuya simulación tiene una gran importancia en los experimentos actuales. Consiste en la capacidad que tienen

algunos tipos de bacterias en traspasar código genético a sus vecinas. Ese código genético que se traspasan modifica el comportamiento a nivel de las bacterias y, por tanto, modifican los resultados generales del experimento.

Gran parte de lo que se considera un simulador de colonias de bacterias recae en el motor de físicas que este contenga. Un simulador de bacterias sin un motor de física no podría simular absolutamente nada, pero un motor de físicas individual ya podría darnos información relevante sobre ciertos fenómenos físicos ocurridos en una colonia bacteriana, de ahí su importancia.

Es además el encargado de resolver las colisiones a nivel de individuos y de la correcta colocación de estos en la colonia mediante un procedimiento de empuje. Este empuje consiste en desplazar el solapamiento que se va produciendo desde el interior hacia el exterior de la colonia, con el fin de relajar todas las fuerzas que están actuando en ella. Este procedimiento es clave en la simulación de una colonia y es el proceso que más cómputo se lleva de toda la simulación.

Para la simulación del cuerpo de una bacteria, se suele utilizar una forma capsular [10]. Cabe destacar que, aunque las simulaciones con este tipo de formas son muy fieles a la realidad, existen ligeras discrepancias debido a que las bacterias no son cuerpos sólidos como tales, y pueden deformarse y adaptarse ligeramente a las fuerzas que actúen sobre ellas. Sin embargo, son computacionalmente rápidas comparándolo con otro tipo de aproximaciones al tomarse éstas como el lugar geométrico a una determinada distancia de un segmento. La rapidez en la colisión es vital ya que se ejecutará cientos de miles de veces por fotograma.

A continuación, vamos a detallar sobre GRO y BactoSIM, dos simuladores de colonias de bacterias que tienen cierta relevancia en la actualidad.

## 2.1. GRO

Gro [11] es un simulador 2D de micro-colonias, desarrollado por el laboratorio de Klavins de la Universidad de Washington y de libre distribución. Lo característico de este simulador es que permite el uso de un lenguaje de muy alto nivel, desarrollado por ellos mismos, para describir y modelar comportamientos bacterianos, aislando al biólogo de la complejidad de la programación de menor nivel, además de permitir la traslación del experimento real al simulado con unas pocas líneas de código. El simulador es considerado de *prototipado*, aunque sus resultados son bastante fieles a los sucesos de la realidad.

Para la realización de la física, Gro tiene en la actualidad implementado un motor físico llamado CellEngine.

## 2.2. BactoSIM

BactoSim [12] es un simulador desarrollado por el departamento LIA, de la UPM como parte del proyecto europeo PLASWIRES. Está sobre todo enfocado, a la conjugación y a la propagación del código genético a través de la colonia. Las células se modelan como círculos sobre una malla discreta con el consiguiente ahorro de cálculo, permitiendo observar la propagación del código genético a través de la colonia. Se considera una herramienta de prototipado.

### 2.3. CellEngine

CellEngine es un motor de física programado en C. Se centra en la simulación rápida de colonias bacterianas en crecimiento. En una colonia bacteriana, cada célula crece en un cierto porcentaje de su tamaño durante cada intervalo de tiempo. En el motor de física, este incremento de pequeño tamaño genera una superposición entre los cuerpos, que son la representación física de las células. El flujo de trabajo básico para resolver la superposición consta de dos etapas: detección de colisión y respuesta de colisión.

### 3. PLANTEAMIENTO DEL PROBLEMA

En los experimentos reales, las pruebas se hacen sobre colonias del orden de  $10^9$  -  $10^{10}$  individuos. Estos experimentos se realizan con distintos tipos de bacterias, aunque por lo general, se suele utilizar la E.coli, una de las más estudiadas. Esta bacteria tiene un tiempo medio de división de 20 minutos [13]. Este valor temporal real, pone un límite a los simuladores de bacterias. Podremos simular un número de bacterias tal que el tiempo que tarda en simularse la siguiente división sea menor que 20 minutos. Dicho de otra manera, una vez que tardemos el mismo tiempo en realizar un experimento en la realidad que en una simulación, esta última pierde gran parte de su sentido. Recordemos que el objetivo principal de simular un experimento es el de reducir su tiempo, y de esa manera poder estudiar más rápidamente el objetivo en cuestión.

El funcionamiento básico de un simulador de físicas de cuerpos sólidos es que cuando dos objetos colisionan, la colisión se resuelve mediante los parámetros físicos asociados a ellos (masa, velocidad, etc.) y por la cantidad de solapamiento producido en la colisión. El resultado, tras su resolución, son dos objetos libres de dicho solapamiento, ya que los cuerpos se han desplazado a distintos lugares. Esto ocurre para dos únicos objetos en el espacio. En cambio, cuando hay más de dos objetos en el espacio, es posible que la resolución de una colisión determine como destino de un objeto un lugar ya contenido por otro. Ello provocaría otro solapamiento que habría que resolver.

En una colonia de bacterias, esto pasa prácticamente siempre al estar todas en contacto. Cualquier resolución de un solapamiento producido entre dos bacterias genera otro solapamiento. ¿Cómo se resuelve esta situación?

### 3.1. Solución actual

Actualmente BactoSIM es un simulador compuesto por agentes, cada agente representa una única célula bacteriana. Los agentes en el simulador interactúan con otros agentes y con el entorno, el cual contiene nutrientes. La representación del entorno es mediante una capa de valor de *repast framework* y los agentes se representan en una cuadrícula de  $1000 \times 1000$ .

BactoSim simula en 20 minutos lo que en la realidad serían 600 minutos de crecimiento bacteriano, consiguiendo así una población final de  $10^5$  bacterias. A continuación, se ilustra en la Figura 1.

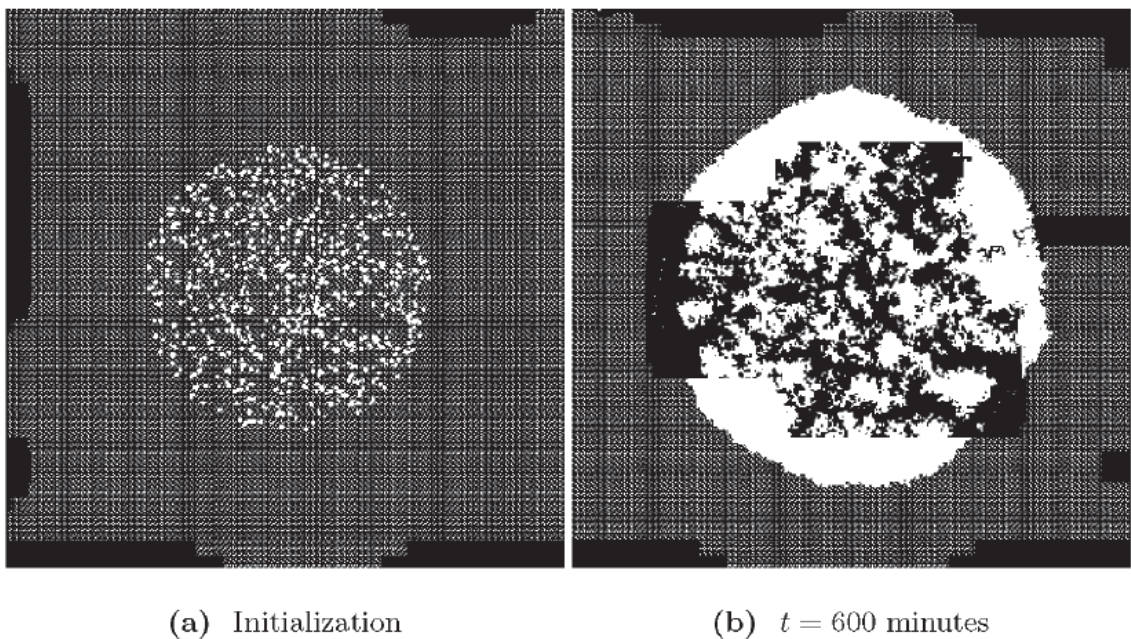


Figura 1: BactoSim simulation

Teniendo en cuenta que las células bacterianas evolucionan durante la simulación teniendo sus variables de estado actualizadas, los estados que puede tener una célula son tres: **R**: *plasmid free*, **D**: *plasmid donor* y **T**: *transconjugant cells*.

En BactoSim los colores utilizados para la visualización son:

- Verde: para las células bacterianas en estado R
- Rojo: para las células bacterianas en estado D
- Azul: para las células bacterianas en estado T.

También utiliza los colores: verde claro, rojo claro y azul claro para representar las zonas de nutrientes agotadas donde las células ya no se dividen.

### 3.2. Análisis del motor físico actual

En términos del motor físico que utiliza este simulador en la actualidad, BactoSim tiene una implementación que consiste en que, por cada agente y cada simulación, se calcula un vector para el desplazamiento de la célula bacteriana simulada en función de las dimensiones de sus vecinos. El funcionamiento es correcto, pero no presenta un gran rendimiento por lo que nuestro proyecto consistirá en la incorporación de un motor físico de mayor potencia para solventar esta carencia que actualmente posee el simulador.



## 4. SOLUCIÓN PROPUESTA

Se pretende incorporar un motor de físicas que mejore el rendimiento que actualmente posee BactoSim. Para el desarrollo de este motor, estudiaremos otro motor existente y eficiente cuyo nombre es CellEngine y basándonos en su metodología iremos desarrollando el nuevo motor de BactoSim.

A continuación, vamos a ver en que consiste CellEngine y cuáles son sus características.

### 4.1. Explicación de CellEngine

CellEngine es un motor físico implementado en lenguaje de programación C[14]. Se centra en la simulación rápida de colonias bacterianas en crecimiento. El cálculo de la física se ha optimizado para cuerpos rígidos con forma de “barra”, como la bacteria E.coli, en sistemas 2D sin inercia. En una colonia bacteriana, cada célula crece un cierto porcentaje de su tamaño durante cada intervalo de tiempo. En un motor de física este incremento de pequeño tamaño genera una superposición entre los cuerpos, que son la representación física de las células. El flujo de trabajo básico para resolver la superposición consiste en dos etapas:

1. Detección de colisión: identifica superposiciones entre bacterias y almacena contactos con información sobre la colisión, como el punto de contacto, la magnitud y la dirección. Figura2 (a)
2. Respuesta de colisión: realiza la disposición física de las células. Calcula el desplazamiento lineal y angular de los cuerpos. Figura2 (b)

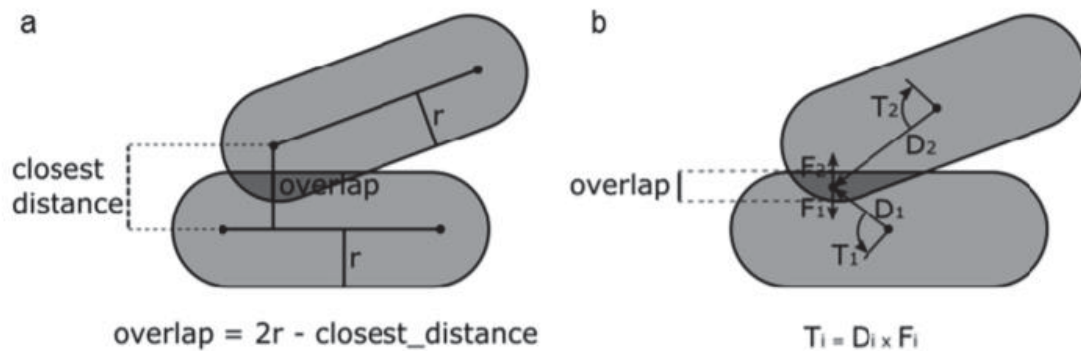


Figura 2: Estado superposición bacterias.

Normalmente, después de la resolución de la superposición de dos células, se generan otras colisiones que deben resolverse. Por lo tanto, las colisiones se resuelven mediante un método iterativo que finalmente desplaza la superposición hacia afuera de la colonia. Este tipo de problema pertenece al problema de muchos cuerpos, cuyos métodos directos se calculan en  $O(N^2)$  siendo  $N$  el número de cuerpos involucrados. En nuestro caso la solución crece exponencialmente, haciendo que las simulaciones se vuelvan demasiado costosas desde el punto de vista computacional. La última solución propuesta proporciona un método de resolución capaz de calcular una aproximación precisa en  $O(N)$  para el mismo problema. La reducción de complejidad computacional se basa en dos supuestos fundamentales:

1. La ubicación y la orientación angular de una bacteria dependen principalmente de las bacterias vecinas.
2. Las colonias tienden a crecer hacia afuera radialmente.

## 4.2. Detalle del algoritmo de CellEngine

La aproximación que tiene implementada se basa en la división de la colonia en anillos concéntricos de un ancho de bacteria. Los anillos se agrupan en conjuntos de anillos con ancho  $w$  que representan el radio  $k$  de la acción de fuerza local, mientras que las bacterias de los anillos internos ejercerán una fuerza global. El algoritmo transforma la colonia en subcolonias anulares concéntricas de cuerpo con ancho  $w$ . Cuando la población de la colonia aumenta, el número de anillos aumenta, pero el ancho de cada subconjunto,  $w$ , se mantendrá constante y se obtendrá una solución global  $O(N)$ . El algoritmo describe la transferencia de superposición en la colonia hacia afuera usando los anillos como guías. Se ejecuta en cada paso de tiempo y tiene dos etapas principales:

1. Etiquetado de anillo : asigna a cada bacteria de la colonia un anillo.  
Comienza desde el borde hacia adentro de la colonia. El etiquetado tiene dos fases: detección de borde y asignación de anillo. Figura 3
2. Expansión: Implica la agrupación de anillos y busca relajar la presión en la colonia. Se lleva a cabo desde el centro de la colonia hasta el borde y se compone de dos fases: relajación y reubicación. Figura 4

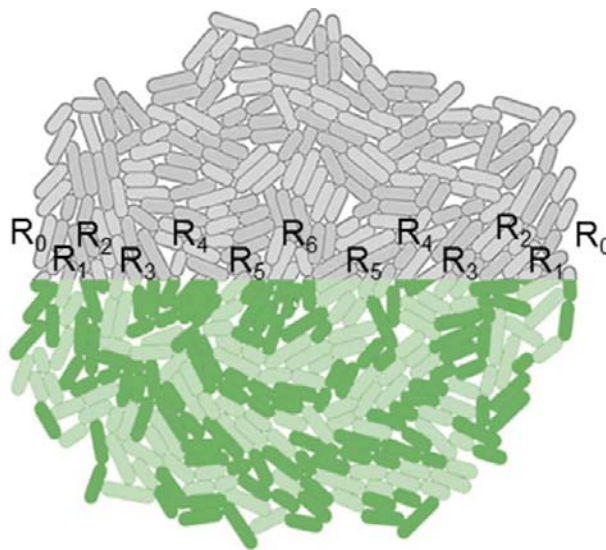


Figura 3: Etiquetado de anillo.

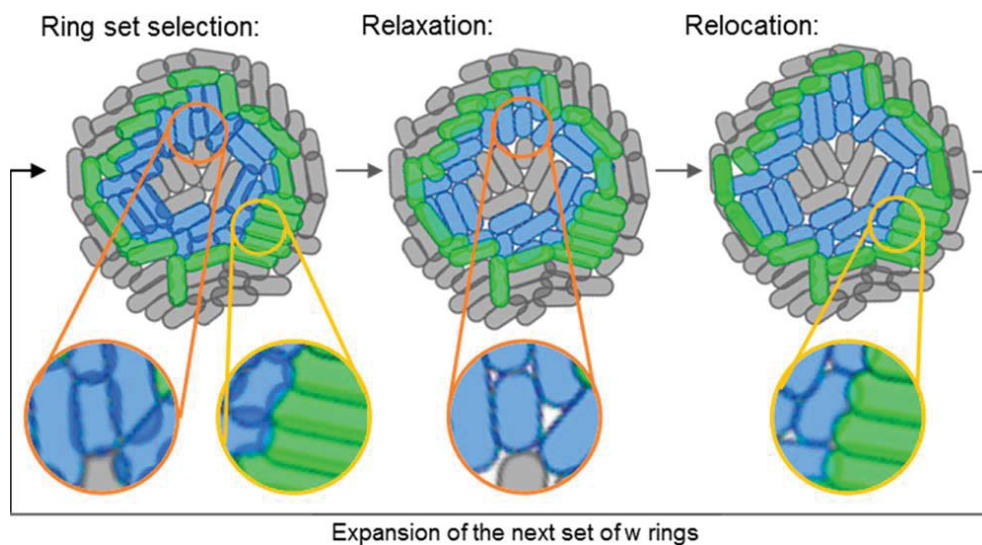


Figura 4: Expansión.

Teniendo en cuenta estas limitaciones, se ejecuta un ciclo de detección y respuesta de colisión para aliviar la presión del conjunto. Esta fase genera superposiciones con bacterias en el grupo externo. Para resolver esta nueva superposición, se ingresa una fase de reubicación. En esta fase, el anillo superpuesto externo se mueve hacia afuera conservando los puntos de contacto y las posiciones relativas con sus vecinos.

### 4.3. Construcción del nuevo motor para BactoSim

La incorporación del nuevo motor en BactoSim hará que este último tenga una capacidad de simulación más eficiente solventando así la carencia que poseía inicialmente.

Dado que nos vamos a basar en CellEngine para crear nuestro motor de simulación hay que tener en cuenta la manera en que se tratan los agentes en BactoSim para así poder realizar una buena implementación.

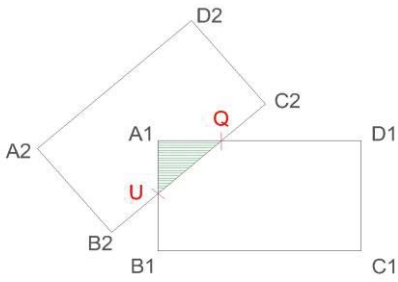
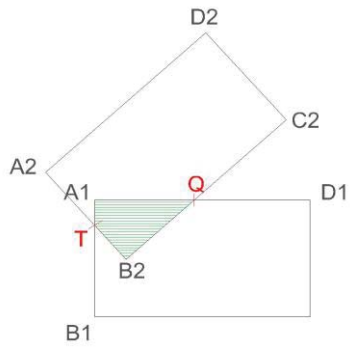
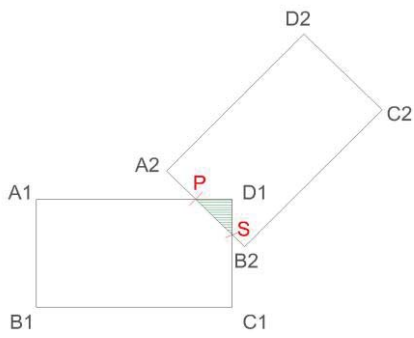
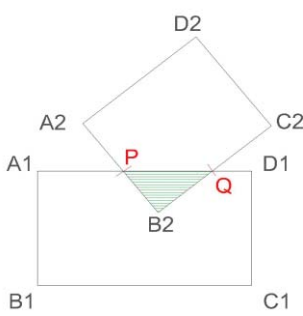
Debido a las diferencias entre simuladores, Gro y BactoSim, partiremos de cero en la implementación del código siguiendo la lógica de CellEngine aplicándola a nuestro simulador BactoSim.

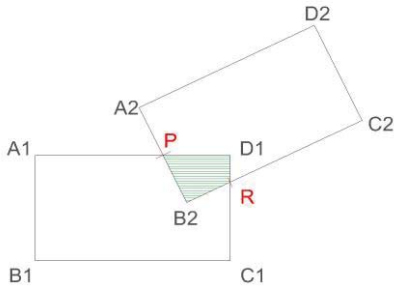
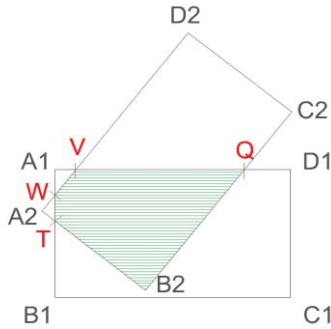
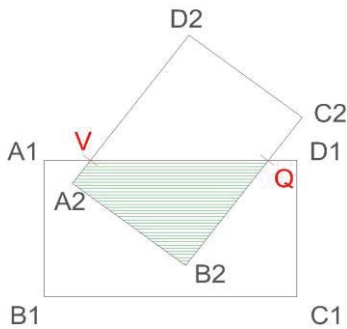
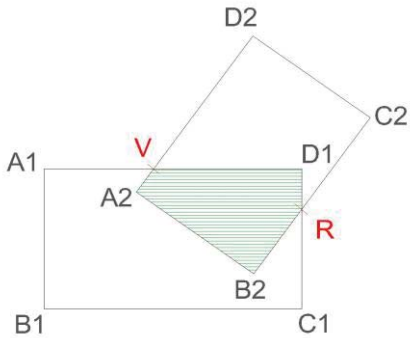
A lo que la complejidad del **algoritmo** se refiere, está **compuesto por cuatro fases**. La **fase de detección** de bordes en la que se calculan los puntos de contacto de todas las

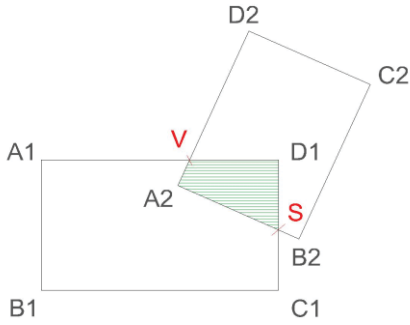
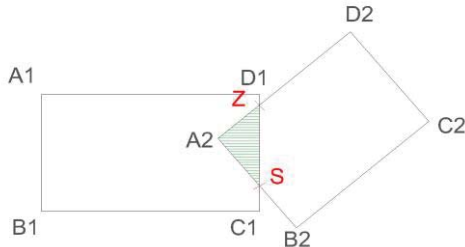
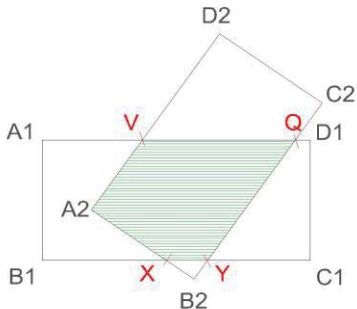
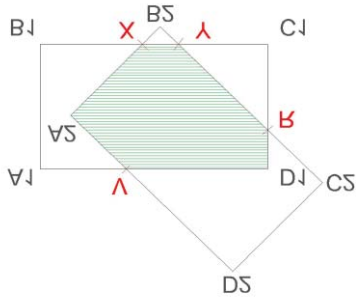
bacterias. Se puede hacer  $O(N)$  ya que el número de vecinos se puede calcular en tiempo constante. Entonces el orden computacional en esta fase es lineal:  $O(N)$ . La siguiente fase sería la **fase de formación del anillo**, para la creación de un nuevo anillo es necesario iterar sobre la bacteria en el último anillo creado. Para la creación de todos los anillos, se realiza una única iteración en toda la colonia, por lo que tenemos  $O(N)$  en esta fase. La tercera **fase** es la **de relajación**, en esta fase se encuentra la detección de la colisión y la respuesta de colisión (explicado en el punto 4.1), estas operaciones requieren computación constante por bacteria. Esta fase se realiza con algunos conjuntos de anillos, pero el proceso mueve el conjunto hacia afuera secuencialmente, por lo que el orden computacional total de todas las llamadas a esta fase es  $O(N)$ . La cuarta **fase** es la **de reubicación**, en esta fase solo se tiene en cuenta un anillo. El cálculo de estas bacterias se realiza con los puntos de contacto, que son independientes de la población, para cada bacteria. Dado que la fase de reubicación se ejecuta para cada anillo, el orden computacional total de todas las llamadas a esta fase es  $O(N)$ .

La **solución que hemos planteado** nosotros mantiene las fases uno, dos y tres del algoritmo de CellEngine de la misma manera, teniendo una función que detecta los bordes, otra que nos genera los anillos y en la tercera fase, la detección de la colisión también es una función de nuestro programa que mira la distancia en la que se encuentran dos células y determina si hay colisión o no entre ellas. La última **fase de reubicación** es la que no hemos seguido de la misma manera que CellEngine, es un algoritmo que está a alto nivel abstracto y partiendo desde cero nosotros hemos creado una manera de resolverlo diferente. Esta solución planteada para la reubicación de las bacterias que están colisionando vamos a explicarla a continuación.

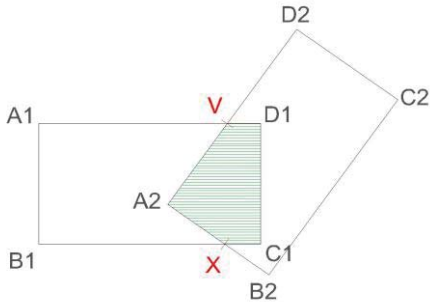
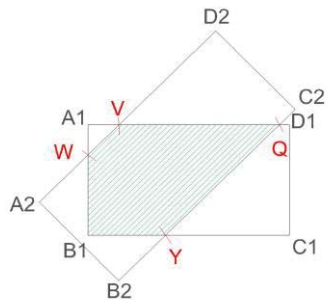
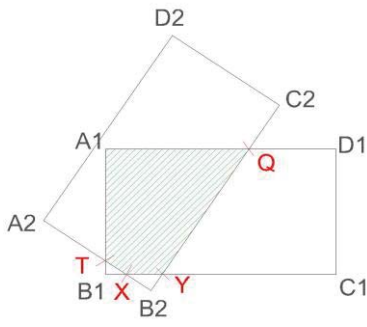
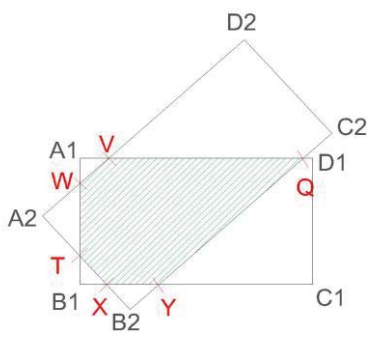
Vamos a estudiar la manera en que dos bacterias pueden colisionar y vamos a crear un algoritmo capaz de resolver esas colisiones y desplazar las células de tal manera que se vayan reubicando en una nueva posición. Hemos definido unos patrones en los que, dadas dos bacterias, podemos ubicarlas en uno de ellos. Los patrones son los siguientes:

	<p><b>CASO 1:</b></p> $y_{A1} > y_{A2}$ $y_{B1} < y_{B2} < y_{A1}$ $x_P < x_{A1}$ $x_{B2} < x_{A1}$
	<p><b>CASO 2:</b></p> $y_{A1} > y_{A2}$ $y_{B1} < y_{B2} < y_{A1}$ $x_P < x_{A1}$ $x_B > x_{A1}$
	<p><b>CASO 3:</b></p> $y_{A1} > y_{A2}$ $y_{B1} < y_{B2} < y_{A1}$ $x_P > x_{A1}$ $x_{B2} > x_{D1}$
	<p><b>CASO 4:</b></p> $y_{A1} > y_{A2}$ $y_{B1} < y_{B2} < y_{A1}$ $x_P > x_{A1}$ $x_{B2} < x_{D1}$ $x_Q < x_{D1}$

	<p><b>CASO 5:</b></p> $yA1 > yA2$ $yB1 < yB2 < yA1$ $xP > xA1$ $xB2 < xD1$ $xQ > xD1$
	<p><b>CASO 6:</b></p> $yB1 < yA2 < yA1$ $yB1 < yB2 < yA1$ $xC1 > xB2$ $xA2 < xA1$
	<p><b>CASO 7:</b></p> $yB1 < yA2 < yA1$ $yB1 < yB2 < yA1$ $xC1 > xB2$ $xA2 > xA1$ $xQ < xD1$
	<p><b>CASO 8:</b></p> $yB1 < yA2 < yA1$ $yB1 < yB2 < yA1$ $xC1 > xB2$ $xA > xA1$ $xQ > xD1$

	<p><b>CASO 9:</b></p> $y_{B1} < y_{A2} < y_{A1}$ $y_{B1} < y_{B2} < y_{A1}$ $x_{C1} < x_{B2}$ $x_V < x_{D1}$
	<p><b>CASO 10:</b></p> $y_{B1} < y_{A2} < y_{A1}$ $y_{B1} < y_{B2} < y_{A1}$ $x_{C1} < x_{B2}$ $x_V > x_{D1}$
	<p><b>CASO 11:</b></p> $y_{B1} < y_{A2} < y_{A1}$ $y_{B2} < y_{B1}$ $x_{A2} > x_{A1}$ $x_Y < x_{C1}$ $x_Q < x_{D1}$
	<p><b>CASO 12:</b></p> $y_{B1} < y_{A2} < y_{A1}$ $y_{B2} < y_{B1}$ $x_{A2} > x_{A1}$ $x_Y < x_{C1}$ $x_Q > x_{D1}$



	<p><b>CASO 13:</b></p> $y_{B1} < y_{A2} < y_{A1}$ $y_{B2} < y_{B1}$ $x_{A2} > x_{A1}$ $x_Y > x_{C1}$ $x_X > x_{C1}$
	<p><b>CASO 14:</b></p> $y_{B1} < y_{A2} < y_{A1}$ $y_{B2} < y_{B1}$ $x_{A2} < x_{A1}$ $x_X < x_{B1}$
	<p><b>CASO 15:</b></p> $y_{B1} < y_{A2} < y_{A1}$ $y_{B2} < y_{B1}$ $x_{A2} < x_{A1}$ $x_X > x_{B1}$ $x_V < x_{A1}$
	<p><b>CASO 16:</b></p> $y_{B1} < y_{A2} < y_{A1}$ $y_{B2} < y_{B1}$ $x_{A2} < x_{A1}$ $x_X > x_{B1}$ $x_V > x_{A1}$

	<p><b>CASO 17:</b></p> $y_{A2} < y_{B1}$ $y_{B2} < y_{B1}$ $x_Y > x_{C1}$
	<p><b>CASO 18:</b></p> $y_{A2} < y_{B1}$ $y_{B2} < y_{B1}$ $x_Y < x_{C1}$ $x_Q > x_{D1}$
	<p><b>CASO 19:</b></p> $y_{A2} < y_{B1}$ $y_{B2} < y_{B1}$ $x_Y < x_{C1}$ $x_Q < x_{D1}$ $x_{A2} < x_{B1}$
	<p><b>CASO 20:</b></p> $y_{A2} < y_{B1}$ $y_{B2} < y_{B1}$ $x_Y < x_{C1}$ $x_Q < x_{D1}$ $x_{A2} > x_{B1}$ $x_V > x_{A1}$

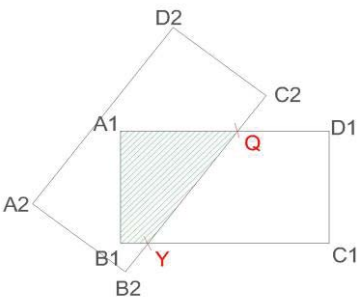
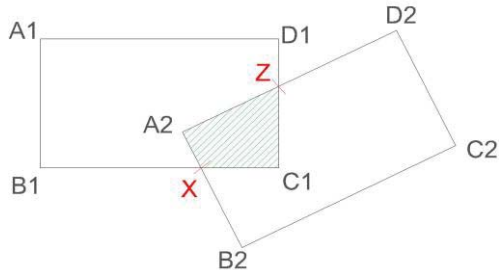
	<p><b>CASO 21:</b></p> $y_{A2} < y_{B1}$ $y_{B2} < y_{B1}$ $x_Y < x_{C1}$ $x_Q < x_{D1}$ $x_{A2} > x_{B1}$ $x_V < x_{A1}$
	<p><b>CASO 22:</b></p> $y_{A2} < y_{A1}$ $y_{B2} < y_{B1}$ $x_V > x_{D1}$

Figura 5: Representación superposición bacterias.

Considerando los siguientes puntos de intersección:

Punto	Intersección de las rectas que unen
P	A1-D1 y A2-B2
Q	A1-D1 y B2-C2
R	C1-D1 y B2-C2
S	C1-D1 y A2-B2
T	A1-B1 y A2-B2
U	A1-B1 y B2-C2
V	A1-D1 y A2-D2
W	A1-B1 y A2-D2
X	B1-C1 y A2-B2
Y	B1-C1 y B2-C2
Z	C1-D1 y A2-D2
N	B1-C1 y A2-D2

*Figura 6: Puntos de intersección.*

**Gracias** a que hemos explotado la **geometría simétrica del sistema** nos basta con considerar estos casos puesto que el resto de los casos se puede hacer corresponder con uno de los anteriores que estudiamos en detalle mediante transformaciones geométricas simples como son, simetrías respecto de los ejes coordenados.

El **movimiento** se produce porque **unas bacterias empujan a otras**, por lo que se generan unas **fuerzas de presión constantes** entre ellas. No son choques violentos ni hay bacterias que se están moviendo velozmente, ni hay choques elásticos o inelásticos que darían lugar a una física distinta y a unas iteraciones distintas. En este modelo, la fuerza que ejercen unas bacterias con otras es por presión de su pared celular a la adyacente porque una se desplaza y empuja a la otra.

Hemos tomado como medida de la presión de una bacteria sobre la adyacente, el **área de superposición** de las bacterias tomando como hipótesis que pudieran penetrarse y pudiera existir un área de intersección. Por lo tanto, en nuestro modelo consideramos que las bacterias se mueven libremente y calculamos el área de intersección que lo utilizamos como una medida de la presión que ejerce una bacteria sobre la otra.

Esta **aproximación** que hemos escogido **la basamos en CellEngine**, que también considera que se puede producir una superposición entre dos células adyacentes generando una fuerza que utilizará para calcular su resultado.

Para **calcular el área de intersección** de dos bacterias adyacentes utilizaremos la **Fórmula de Herón** [15] que nos da el área de un triángulo conociendo las longitudes de sus lados.

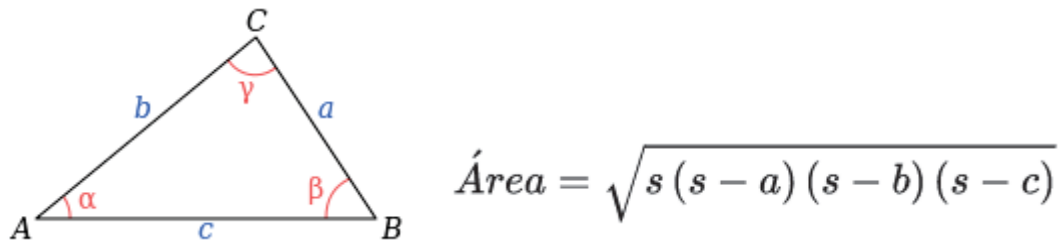


Figura 7: Fórmula de Herón.

Siendo  $s$  el semiperímetro:

$$s = \frac{a + b + c}{2}.$$

Figura 8: semiperímetro para formula de Herón.

Por lo tanto, el área de la intersección de las dos bacterias deberemos dividirlo en triángulos y mediante la fórmula de Herón calcular el área de intersección total.

Para **la resolución del movimiento** consideramos que solo se mueve una bacteria. Esta consideración la basamos en que en nuestro algoritmo iremos recorriendo todos los anillos y comprobando cada posición. Si en una posición determinada tenemos localizadas dos células, entonces, iteraremos sobre una de ellas para que se deshaga la superposición y seguiremos con el algoritmo. Al ir recorriendo cada posición de cada anillo, nos sirve con mover una de las dos bacterias que están colisionando, puesto que vamos a ir **resolviendo las superposiciones iterativamente**.

El **movimiento** lo consideramos como la **descomposición en dos partes** para simplificar el modelo. **Primero** un **desplazamiento** de la bacteria en la dirección que une el centro de gravedad de esta bacteria con el centro de gravedad de la bacteria que la empuja y **segundo**, un **giro** que se determina según convenga basándonos en la geometría del sistema. Esta aproximación la escogemos para simplificar el modelo en vez de considerar una representación explícita de fuerzas y momentos en nuestro programa.

A continuación, **detallamos la descomposición del movimiento** que hemos considerado como aproximación:

**D** es el **desplazamiento** y **G** es el **giro** que vamos a ejercer sobre la célula para deshacer su superposición.

$$D = k1 * A$$

$$G = k2 * A$$

Sabiendo que,

$$k1 = \frac{m2}{m1} * \beta_P$$

$$k2 = \frac{m2}{m1} * \frac{1}{s^2 + t^2} * \beta_G$$

*Figura 9: Desplazamiento y Giro.*

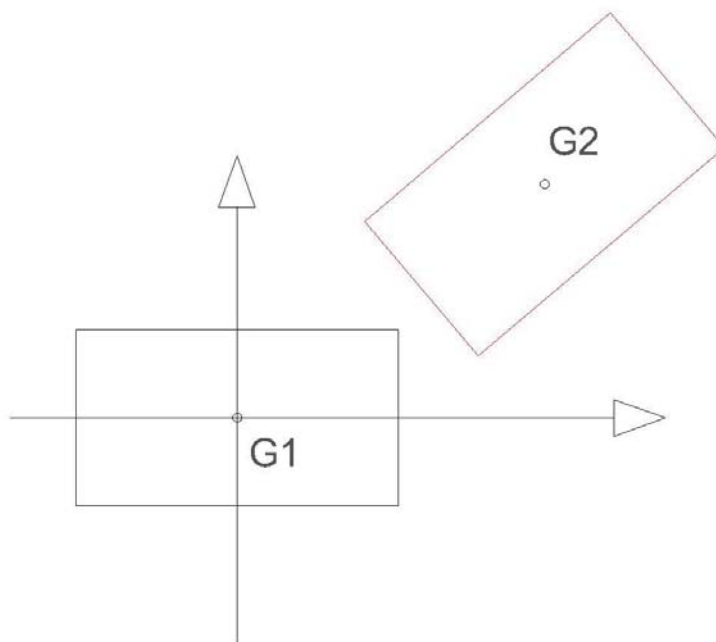
Siendo **A** el **área de la intersección de las dos células**, **k1** una **constante** utilizada para el **desplazamiento** que obtenemos de multiplicar la división de las **masas de ambos cuerpos (m1 y m2)** por una **constante de aproximación**. **k2** es otra **constante** que utilizamos para el **giro** y la **obtenemos de multiplicar las masas de las células por su momento de inercia**.

Para realizar el **giro de la bacteria** tendremos en cuenta la posición inicial de las células por lo que comprobaremos si hemos hecho alguna simetría para deshacerla y calcular el giro a partir de las posiciones iniciales. Estos son los casos posibles teniendo en cuenta el cambio de ejes que hemos implementado en el que el centro de gravedad de la célula 1 sería el centro de coordenadas.

#### CASO 1:

Coordenadas de G2 > 0.

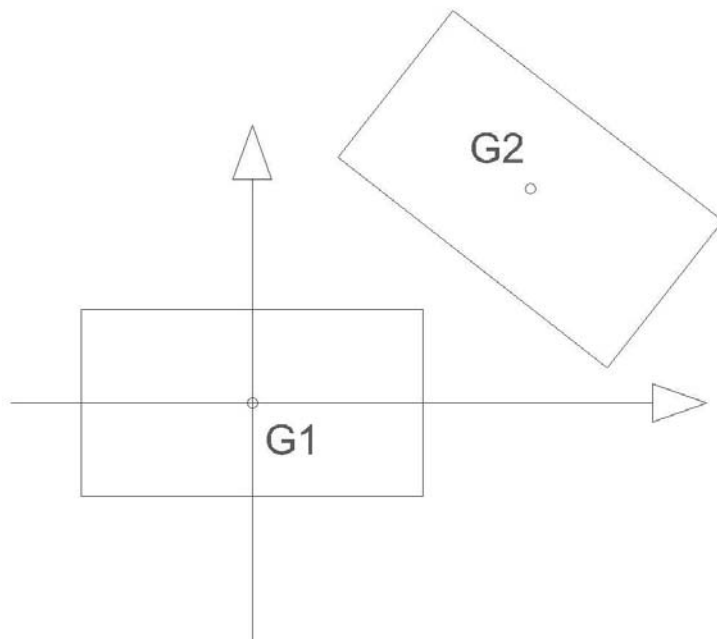
$(\alpha_2 - \alpha_1) > 0$ , siendo  $\alpha_1$  el ángulo de la célula 1 y  $\alpha_2$  el ángulo de la célula 2.



**CASO 2:**

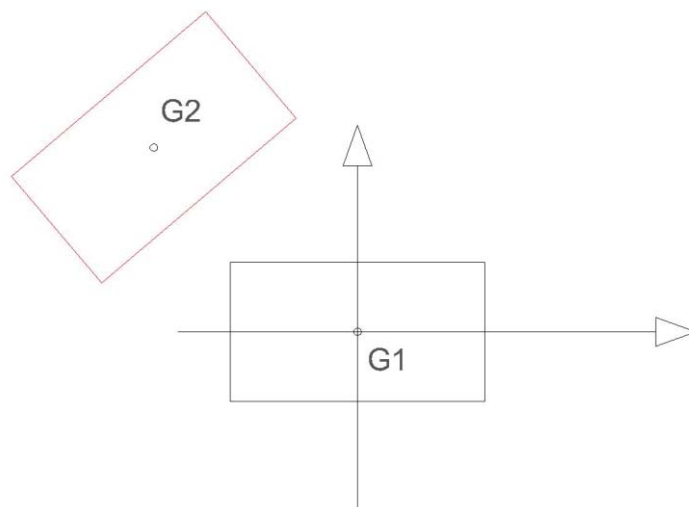
Coordenadas de  $G2 > 0$ .

$(\alpha_2 - \alpha_1) < 0$ , siendo  $\alpha_1$  el ángulo de la célula 1 y  $\alpha_2$  el ángulo de la célula 2.

**CASO 3:**

Coordenadas de  $G2 = (xG2, yG2)$ ,  $xG2 < 0$ ,  $yG2 > 0$ .

$(\alpha_2 - \alpha_1) > 0$ , siendo  $\alpha_1$  el ángulo de la célula 1 y  $\alpha_2$  el ángulo de la célula 2.

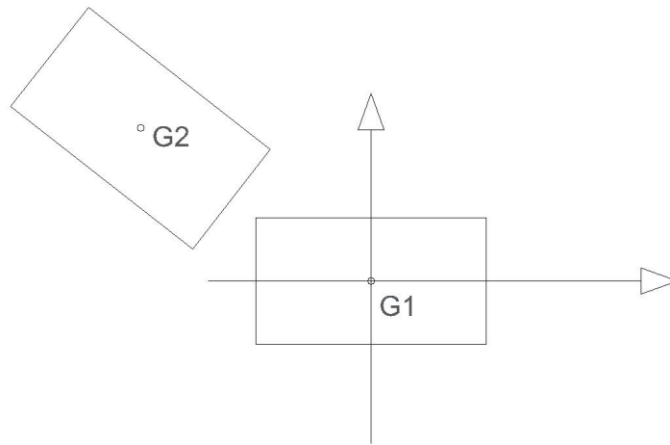




**CASO 4:**

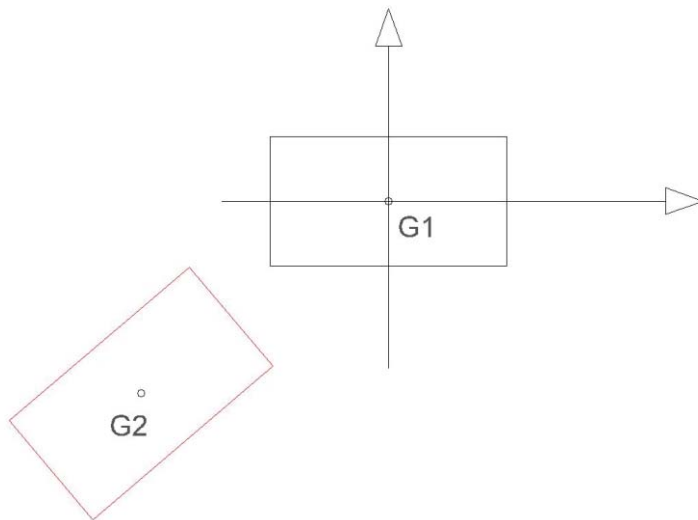
Coordenadas de  $G2 = (x_{G2}, y_{G2})$ ,  $x_{G2} < 0$ ,  $y_{G2} > 0$ .

$(\alpha_2 - \alpha_1) < 0$ , siendo  $\alpha_1$  el ángulo de la célula 1 y  $\alpha_2$  el ángulo de la célula 2.

**CASO 5:**

Coordenadas de  $G2 = (x_{G2}, y_{G2})$ ,  $x_{G2} < 0$ ,  $y_{G2} < 0$ .

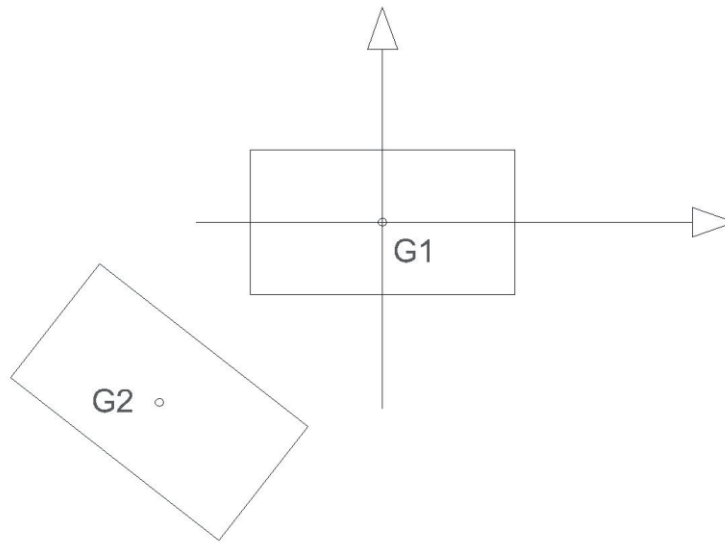
$(\alpha_2 - \alpha_1) > 0$ , siendo  $\alpha_1$  el ángulo de la célula 1 y  $\alpha_2$  el ángulo de la célula 2.



**CASO 6:**

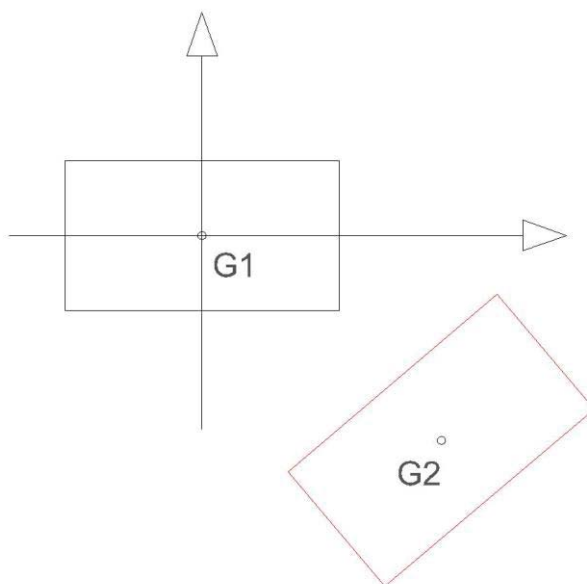
Coordenadas de  $G2 = (x_{G2}, y_{G2})$ ,  $x_{G2} < 0$ ,  $y_{G2} > 0$ .

$(\alpha_2 - \alpha_1) < 0$ , siendo  $\alpha_1$  el ángulo de la célula 1 y  $\alpha_2$  el ángulo de la célula 2.

**CASO 7:**

Coordenadas de  $G2 = (x_{G2}, y_{G2})$ ,  $x_{G2} > 0$ ,  $y_{G2} < 0$ .

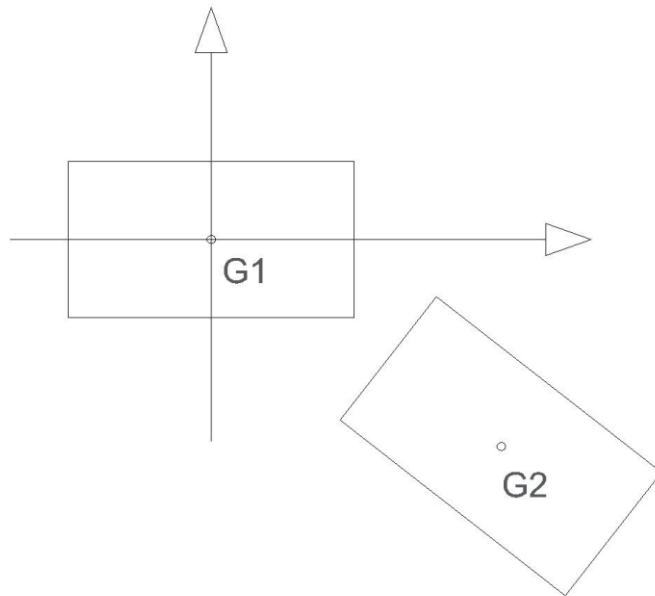
$(\alpha_2 - \alpha_1) > 0$ , siendo  $\alpha_1$  el ángulo de la célula 1 y  $\alpha_2$  el ángulo de la célula 2.



**CASO 8:**

Coordenadas de  $G2 = (x_{G2}, y_{G2})$ ,  $x_{G2} > 0$ ,  $y_{G2} < 0$ .

$(\alpha_2 - \alpha_1) < 0$ , siendo  $\alpha_1$  el ángulo de la célula 1 y  $\alpha_2$  el ángulo de la célula 2.



En nuestro modelo creado para el estudio de las superposiciones de dos células, se cubren los casos 1, 3 y 7 representados anteriormente, por lo que, el resto de los casos habrá que aplicarles simetrías sobre la bacteria 2 para que se posicionen como deseamos y así poder estudiarlos más fácilmente.

Para el caso 2, haremos una simetría respecto al eje X de coordenadas, haciendo una aproximación con el caso 7. Para el caso 4 haremos simetría respecto al eje Y de coordenadas, aproximándolo al caso 1. En el caso 5, haremos primero una simetría respecto del eje X y después una simetría respecto del eje Y, aproximando así el caso 5 con el caso 1. Para el caso 6, se ha de hacer una simetría respecto al eje X de coordenadas para obtener una aproximación con el caso 3. Finalmente, para el caso 8, haremos una simetría respecto al eje X aproximándolo con el caso 1.

Una vez que hemos estudiado las simetrías y hemos hecho los cálculos anteriores, para estudiar hacia donde se va a producir el giro, **calculamos el baricentro del área de intersección** obtenido anteriormente **para definir si el giro se producirá en sentido horario o antihorario**.

Para el cálculo de las coordenadas del baricentro de la intersección de las dos bacterias. Utilizamos esta fórmula [16]:

$$\overrightarrow{OG} = \frac{\sum m_i \overrightarrow{OA_i}}{\sum m_i} = \frac{m_1 \overrightarrow{OA_1} + \dots + m_n \overrightarrow{OA_n}}{m_1 + \dots + m_n}, \quad \text{con} \quad \sum m_i \neq 0$$

Figura 10: Fórmula para encontrar el baricentro.

Siendo  $A_1 \dots A_n$  n puntos y  $m_1 \dots m_n$  n números (m como masa). Entonces **el baricentro** de los  $(A_i, m_i)$  en el punto **G** se calcula de la manera definida.

Una vez obtenido el baricentro, estudiaremos el giro que se producirá en cada caso. Partiendo de que **si el producto de las coordenadas del baricentro es positivo** entonces el **giro** se hará en **sentido horario** y si el **producto es negativo** el **giro** se realizará en **sentido antihorario**.

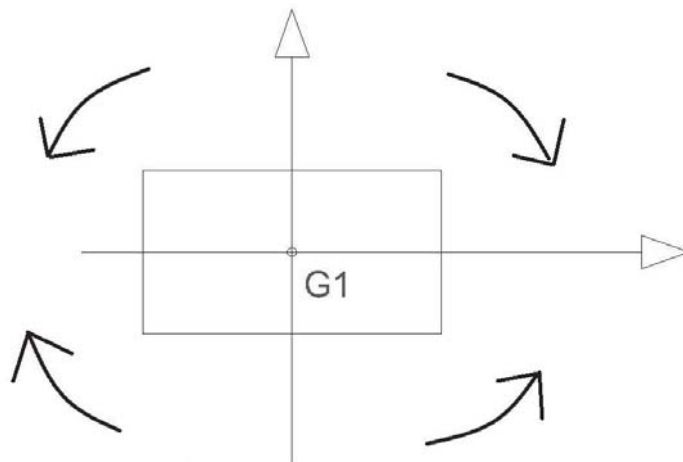


Figura 11: Cálculo de Giro.

A continuación, presentamos detalladamente el **cálculo del giro** para cada caso:

- **Caso 1:**
  - Si  $x * y > 0 \rightarrow \alpha_{nuevo} = \alpha_{inicial} - G$
  - Si  $x * y < 0 \rightarrow \alpha_{nuevo} = \alpha_{inicial} + G$
- **Caso 2:**
  - Si  $x * y > 0 \rightarrow \alpha_{nuevo} = \alpha_{inicial} - G$
- **Caso 3:**
  - Si  $x * y < 0 \rightarrow \alpha_{nuevo} = \alpha_{inicial} + G$
- **Caso 4:**
  - Si  $x * y > 0 \rightarrow \alpha_{nuevo} = \alpha_{inicial} + G$
  - Si  $x * y < 0 \rightarrow \alpha_{nuevo} = \alpha_{inicial} - G$
- **Caso 5:**
  - Si  $x * y > 0 \rightarrow \alpha_{nuevo} = \alpha_{inicial} - G$
  - Si  $x * y < 0 \rightarrow \alpha_{nuevo} = \alpha_{inicial} + G$
- **Caso 6:**
  - Si  $x * y > 0 \rightarrow \alpha_{nuevo} = \alpha_{inicial} - G$
- **Caso 7:**
  - Si  $x * y < 0 \rightarrow \alpha_{nuevo} = \alpha_{inicial} - G$
- **Caso 8:**
  - Si  $x * y > 0 \rightarrow \alpha_{nuevo} = \alpha_{inicial} - G$
  - Si  $x * y > 0 \rightarrow \alpha_{nuevo} = \alpha_{inicial} + G$

Siendo  $(x, y)$  **coordenadas del baricentro del área de intersección de las dos células** y  $G = k2 * A$ , definido anteriormente, con  $k2 = \frac{m2}{m1} * \frac{1}{s^2+t^2} * \beta_G$  y  $A$  el **área de intersección** de ambas células.

## 5. RESULTADOS Y VALIDACIÓN

En este punto se van a tratar los resultados de este proyecto. Vamos a realizar una comparativa y vamos a estudiar los resultados obtenidos con la implementación del motor físico en el simulador BactoSim.

Con la implantación del nuevo motor explicado en el apartado anterior, hemos conseguido una mejora en el tiempo de propagación de la colonia de bacterias con respecto a la anterior implementación.

En el siguiente gráfico se puede observar como el crecimiento de los agentes con el nuevo motor es más eficiente que el anteriormente implantado.

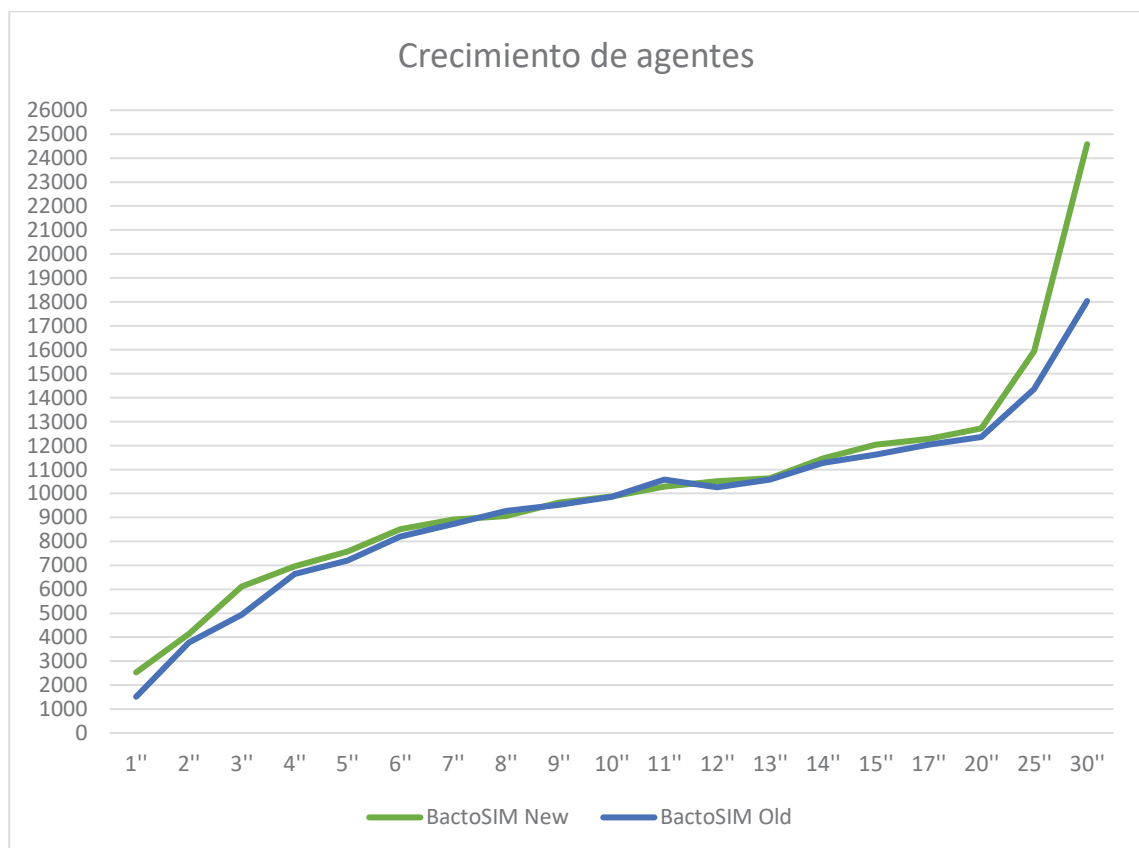
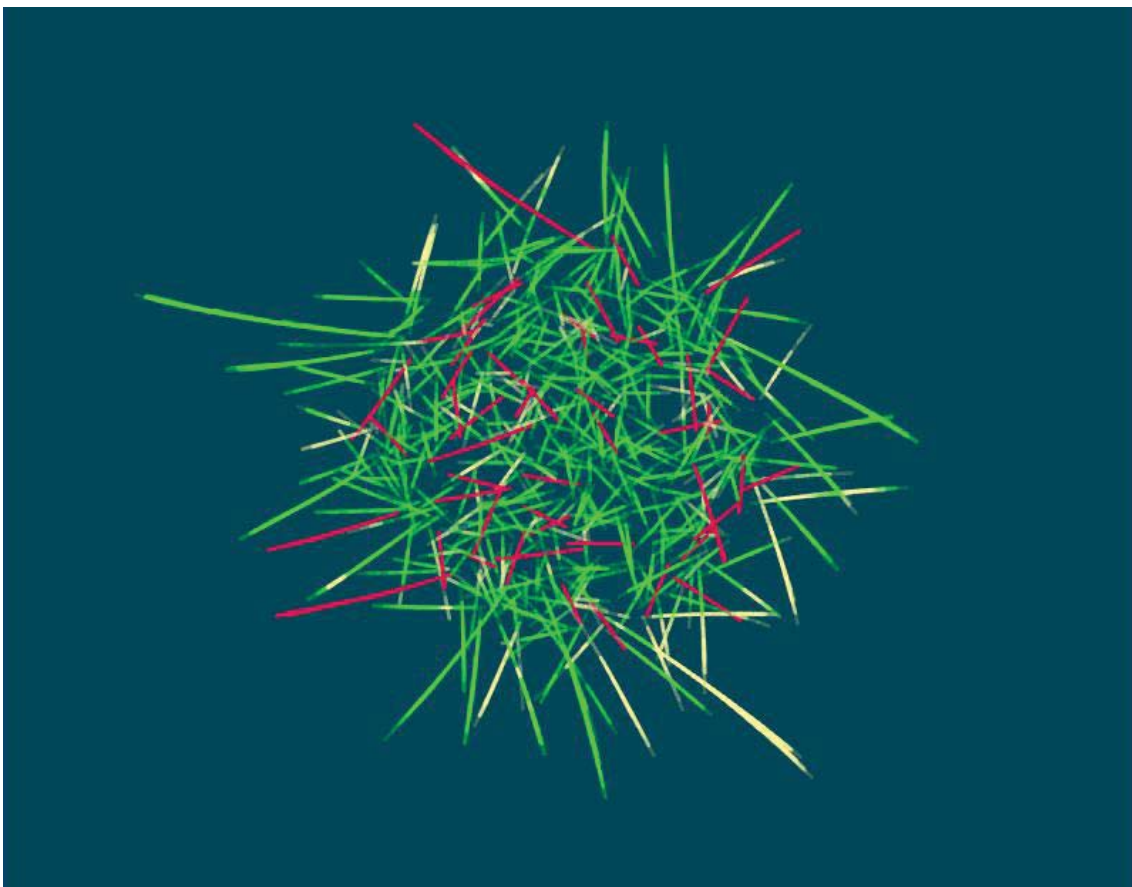


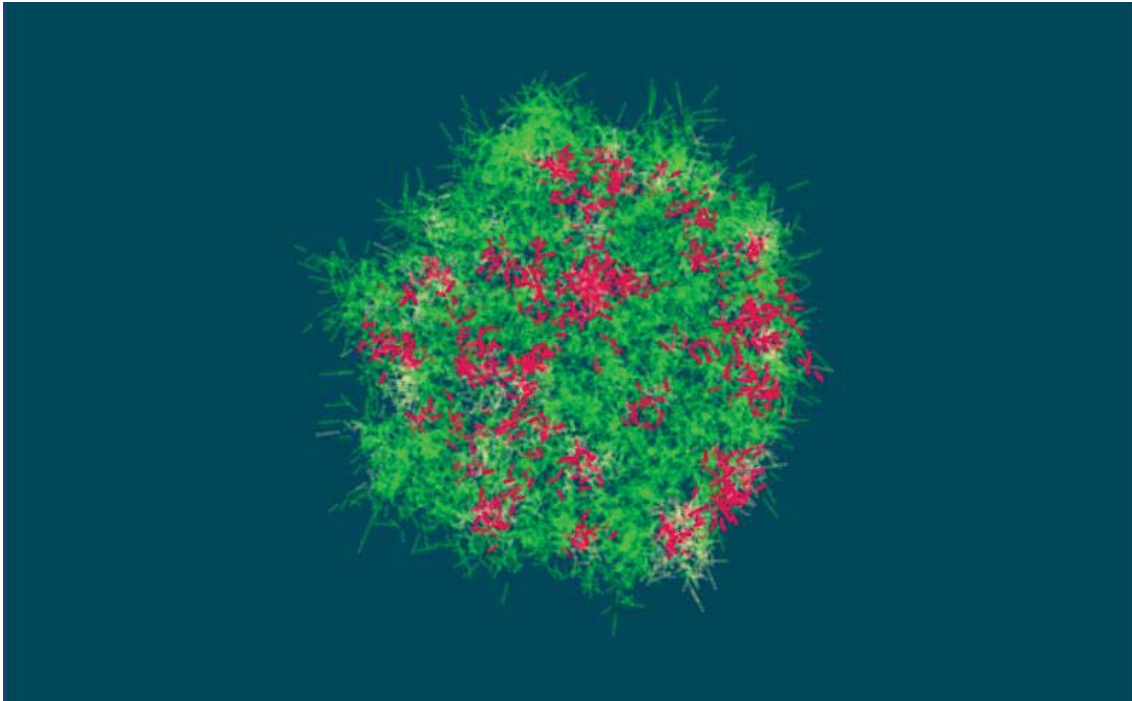
Figura 12: Gráfico comparativo.

Tanto al inicio como al final del crecimiento se puede ver una clara diferencia en cuanto a resultados. Mejorando por tanto el modelo que había implantado anteriormente.

Con respecto a la manera en que los agentes van creciendo también se ve una clara diferencia entre el modelo antiguo y el nuevo. Anteriormente las bacterias crecían de una forma menos regular que ahora.



*Figura 13: BactoSim versión antigua.*



*Figura 14: BactoSim con el nuevo motor implantado.*

Se puede observar en las anteriores ilustraciones como con el nuevo motor el crecimiento es más regular y las bacterias crecen con una forma más parecida a una colonia real.



## 6. CONCLUSIONES Y FUTURAS LINEAS DE TRABAJO.

### 6.1. Conclusiones

Los simuladores de colonias bacterianas son los encargados de simular la iteración que se produce tanto entre las propias bacterias como entre las bacterias y las señales del medio. Es además el encargado de resolver las colisiones a nivel de individuos y de la correcta colocación de estos en la colonia mediante un procedimiento de empuje. Este empuje consiste en desplazar el solapamiento que se va produciendo desde el interior hacia el exterior de la colonia, con el fin de relajar todas las fuerzas que están actuando en ella. Este procedimiento es clave en la simulación de una colonia y es el proceso que más cómputo se lleva de toda la simulación. Utilizando estos simuladores, los biólogos pueden realizar experimentos orientativos acerca de cómo afecta una sustancia a un conjunto de bacterias o como se propaga cierta señal entre ellas.

BactoSim es una herramienta de simulación de colonias bacterianas que está sobre todo enfocado, a la conjugación y a la propagación del código genético a través de la colonia. Las células inicialmente se modelaban como círculos sobre una malla discreta con el consiguiente ahorro de cálculo, permitiendo observar la propagación del código genético a través de la colonia. BactoSim no posee un motor físico eficiente por lo que surge la necesidad de implementarlo de manera que sean más eficientes los tiempos de ejecución. Para la implementación del nuevo motor nos hemos basado en el simulador GRO, el cual posee un motor llamado CellEngine.

El nuevo motor para BactoSim utiliza un método iterativo que se compone de cuatro fases, fase de detección de bordes, fase de formación de anillo, fase de relajación y fase de reubicación. Las tres primeras fases seguimos la idea general que tiene CellEngine, pero es en la cuarta fase donde se implementa un nuevo algoritmo para la reubicación de las bacterias que están colisionando de tal manera que se vayan reubicando en una nueva posición.

Con el nuevo motor implementado obtenemos mejoras en los tiempos de ejecución de BactoSim, lo cual es buena señal ya que nos acerca más a una simulación real que en un futuro permitirá a los investigadores ser mas eficientes en sus trabajos de investigación acotando los ciertos casos que pueden ver claramente negativos sin la necesidad de realizar todas las simulaciones en las placas de Petri lo cuál les lleva mucho tiempo.

## 6.2. Futuras líneas de trabajo

Para continuar con este proyecto en un futuro existen varias líneas de trabajo. La primera se podría mejorar el algoritmo implementado para que fuera más eficiente haciendo que la comprobación de la superposición de las bacterias fuera más dinámica y no tuviera que comprobar todas las bacterias en cada pasada si no que siguiera un patrón que nos ayudase a mover desde dentro hacia afuera las bacterias en una pasada al igual que tiene implementado CellEngine.

Otra posible línea de trabajo futura podría ser la visualización de los agentes en forma de bacilo en vez de rectángulos como está actualmente.

## Anexos

Código del proyecto: <https://github.com/carolalonsome/BactoSIM>

## Bibliografía

- [1] <http://conceptodefinicion.de/biologia-sintetica/>
- [2] Wikipedia. Synthetic Biology.
- [3] Diccionario Google.
- [4] Wikipedia. Simulador.
- [5] Andrianantoandro E, Basu S, Karig DK, Weiss R (2006) Synthetic biology: new engineering rules for an emerging discipline. *Mol Syst Biol* 2: 2006.0028.
- [6] Heinemann M, Panke S (2006) Synthetic biology, putting engineering into biology. *Bioinformatics* 22: 2790–2799.
- [7] O'Toole G, Kaplan HB, Kolter R (2000) Biofilm formation as microbial development. *Annu Rev Microbiol* 54: 49–79.
- [8] 4. A bunch of tiny individuals-Individual-based modeling for bacteria IBM Helweber Bucci 2009 FUND
- [9] Tesis Máster Antonio García conjugación agosto 2011 FI UPM
- [10] Real-Time Collision Detection-Christer Ericson
- [11] Jang et al. - Specification and Simulation of Synthetic Multicelled Behaviors - ACS Synthetic Biology - 20128
- [12] D5.2: Reports on experimental validation and theoretical analysis: Bactosim I: Individual Based Model (IBM) for theoretical analysis of conjugative plasmid propagation
- [13] (23) Cullum, J., and Vicente, M. (1978) Cell growth and length distribution in *Escherichia coli*. *J. Bacteriol.* 134, 330–337
- [14] Research Article: A New Improved and Extended Version of the Multicell Bacterial Simulator gro.
- [15] [https://es.wikipedia.org/wiki/F%C3%B3rmula\\_de\\_Her%C3%B3n](https://es.wikipedia.org/wiki/F%C3%B3rmula_de_Her%C3%B3n)
- [16] <https://es.wikipedia.org/wiki/Baricentro>

Este documento esta firmado por



<b>Firmante</b>	CN=tfgm.fi.upm.es, OU=CCFI, O=Facultad de Informatica - UPM, C=ES
<b>Fecha/Hora</b>	Tue Jul 10 15:36:50 CEST 2018
<b>Emisor del Certificado</b>	EMAILADDRESS=camanager@fi.upm.es, CN=CA Facultad de Informatica, O=Facultad de Informatica - UPM, C=ES
<b>Numero de Serie</b>	630
<b>Metodo</b>	urn:adobe.com:Adobe.PPKLite:adbe.pkcs7.sha1 (Adobe Signature)