Simulación de crecimiento bacteriano en un entorno cerrado

José Antonio González González [*] aang@ciencias.unam.mx
Emilio Sánzhez Olivares [**] emilio_san_o@ciencias.unam.mx

Facultad de Ciencias
Universidad Nacional Autónoma de México

Fecha de entrega: 6 de junio de 2023

Resumen

En este proyecto se desarrolló una simulación para estudiar el crecimiento bacteriano en un entorno cerrado para después evaluar el efecto de un antibiótico en la población bacteriana. Para ello, se implementó un modelo en Python que representa un contenedor con dimensiones definidas, donde las bacterias pueden reproducirse. El objetivo principal fue analizar el crecimiento exponencial de las bacterias y el impacto de la acción del antibiótico en su población.

Se diseñó una clase llamada "bactery"para modelar las bacterias, donde se definieron atributos como la posición y el tiempo de reproducción. Las bacterias se mueven a su alrededor dentro del contenedor y se reproducen cuando alcanzan un tiempo de reproducción determinado.

1. Introducción

Las bacterias son microorganismos unicelulares presentes en nuestro entorno, estas desempeñan un papel fundamental en el mundo natural y en la salud humana. Estas diminutas criaturas tienen la capacidad de colonizar diversos hábitats, desde el suelo y el agua hasta nuestro propio cuerpo, donde juegan roles vitales en los ciclos biogeoquímicos y en la digestión. Sin embargo, el impacto de las bacterias no se limita solo a su función ecológica, ya que su capacidad para causar enfermedades ha sido reconocida desde tiempos inmemoriales.

En un sistema biológico, el crecimiento bacteriano se puede dividir en tres fases distintas:

Fase de Retraso (Lag Phase): Esta fase es el período de adaptación en el que las bacterias se están preparando para el crecimiento activo. Durante esta fase, las células bacterianas están sintetizando enzimas y metabolitos necesarios para el crecimiento. No hay un aumento significativo en el número de células ni en la masa celular durante esta fase.

Fase Logarítmica (Log Phase): También conocida como la fase de crecimiento exponencial, es cuando las bacterias están experimentando un crecimiento rápido y activo. Durante esta fase, las células se dividen por mitosis y el número de células se duplica en cada generación. Es en esta fase cuando se produce el aumento máximo en la masa celular y la población bacteriana se expande rápidamente.

Fase Estacionaria (Stationary Phase): En esta fase, el crecimiento bacteriano se ralentiza y se alcanza un equilibrio entre el crecimiento y la muerte de las células. La población bacteriana se estabiliza debido a factores como la disponibilidad limitada de nutrientes, la acumulación de productos metabólicos tóxicos y la competencia entre las células. Durante esta fase, el número de células que se dividen es igual al número de células que mueren, lo que da lugar a una población bacteriana estable.

Fase de muerte: En esta fase se da una muerte de las células que corrresponden a la población deter-

[*]
[**]

minado por un proceso conocito por apotosis , que intercambia proteínas en la mebrana celular para que estas sean fagocitas.

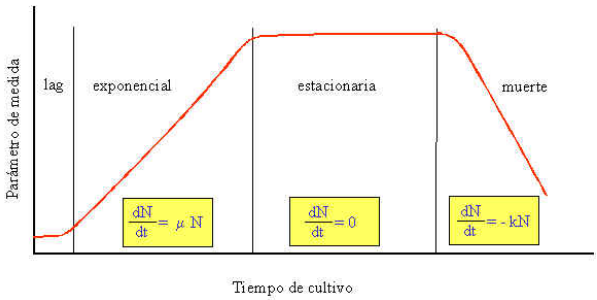

Figura 1. Fases del crecimiento bacteriano

El estudio del crecimiento y la reproducción bacteriana es de suma importancia, ya que nos permite comprender mejor los mecanismos de su expansión y evolución. Las bacterias se reproducen a través de mitosis, esto les permite aumentar su población de manera exponencial en condiciones favorables. Este crecimiento acelerado puede conducir a la formación de colonias bacterianas y, en algunos casos, a la aparición de infecciones con consecuencias graves para la salud humana.

La comprensión de los procesos de crecimiento bacteriano y el impacto sobre la vida de las personas en fundamental para la medicina. Mediante la simulación computacional, es posible explorar y analizar cómo las bacterias se reproducen y propagan en entornos cerrados. Esto proporciona una plataforma para investigar estrategias para el control de infecciones y explorar enfoques para su tratamiento.

Programación Orientada a Objetos (OOP)

La programación orientada a objetos (POO) es un paradigma de programación que se basa en la idea de organizar el código en entidades llamadas objetos, los cuales representan entidades del mundo real y tienen su propio estado interno y comportamiento esto se ve representado mediante los atributos (características que poseen) y métodos (acciones que realizan). En la POO, los programas se construyen a partir de la interacción entre estos objetos, los cuales se comunican entre sí enviándose mensajes.

Algunos conceptos clave en la programación orientada a objetos son: Clase: Es una plantilla o modelo que define la estructura y el comportamiento de un tipo de objeto. Una clase define los atributos (variables) que tiene un objeto y los métodos (funciones) que puede realizar. Objeto: Es una instancia de una clase. Los objetos son entidades concretas que pueden tener su propio estado interno y realizar acciones específicas. Encapsulación: Es el concepto de agrupar datos (atributos) y comportamiento (métodos) dentro de un objeto. Los objetos encapsulan su estado interno y proporcionan interfaces para interactuar con ellos. Herencia: Es un mecanismo que permite que una clase herede atributos y métodos de otra clase. La herencia permite crear jerarquías de clases y reutilizar código.

Polimorfismo: Es la capacidad de un objeto de tomar diferentes formas y comportarse de manera diferente en función del contexto. El polimorfismo permite tratar diferentes objetos de manera uniforme a través de interfaces comunes. Un ejemplo de la aplicación de la OPP es es siguiente.

```
class Perro:
    #método constructor de atributos
    def ..init..(self,raza,edad,color):
        self.raza = raza #raza del perro
        self edad = edad
        self.color = color
    #métodos
    def correr(self):
        if self.raza == umbral:
            print('el perro corre')
    def avanzar_Edad(self):
```

Se define una clase perro que tiene como atributos una raza, una edad y un color; y como métodos tiene correr y avanzar edad.

2. Metodología

A través de la programación orientada a objetos (OOP) se creó un programa en Python que simulara una bacteria y un espacio cerrado en el cuál se pudiera reproducir.

Se agregaron las bibliotecas <matplotlib y <numpy> forma :

```
#import matplotlib.pyplot as plt
#import numpy as np
```

2.1. Creación de la bacteria

Con la programación orientada a objetos se creó una clase bacteria "bactery"que tiene como atributos una posición, un tiempo y diferentes métodos para simular la reproducción bacteriana.

```
class bactery :
    def __init__(self, position, time):
        self.position = position
        self.time = time

    def actualize_time(self):
        self.time += 1

    def chechktime(self):
        if self.time_reprocutcion()== True:
            self.move()
    def time_reprocutcion(self):
        if self.time %2  == 0:
            return True
        else:
            return False

    def death(self):
        if self.time == 15:
            self.position = ()
            print('the bactery is death')
            return True
        else:
            return False

    def move(self):
        self.position[0] = self.position[0]
        + np.random.choice([0, 1,-1])
        self.position[1] = self.position[1]
        + np.random.choice([0, 1,-1])

        newposition = tuple(self.position[0]
        + np.random.choice([0, 1,-1]),
        self.position[2] +
        np.random.choice([0, 1,-1]))

class newbactery(bactery):
    def __init__(self,position,time,strength):
        super().__init__(position, time)  # Llama al constructor de la clase base
        self.time = time +1
```

El método *actualize.time*$(self)$ incrementa en uno el tiempo de la bacteria.
El método chechktime(self) verifica si la bacteria puede reproducirse según su tiempo.
El método *time.reprocutcion*$(self)$ verifica si el tiempo de la bacteria es par y, por lo tanto, puede reproducirse.
El método *death*$(self)$ verifica si el tiempo de la bacteria es igual a 15 y, en ese caso, la elimina y muestra un mensaje de que la bacteria ha muerto.
El método *move*$(self)$ mueve la bacteria a una nueva posición dentro de un rango específico y la muestra en un gráfico junto con el contenedor.

El método *..str..*$(self)$ devuelve una representación en cadena de la bacteria.

2.2. Creación del contenedor

Una vez lista la bacteria y sus métodos era necesario colocarla en un espacio confinado para su reproducción, para esto se creó una clase llamada Çontenedorçon atributos de un tiempo global y los tiempos, posiciones y número de bacterias que se encuentran dentro de él en un determinado periodo de tiempo.

```
class Contenedor:
    def __init__(self,timeglobal,
    times,positions,bacterias):

        self.timeglobal = timeglobal
        self.times = times
        self.positions = positions
        self.bacterias = bacterias

    def lista_timepos(self):
        tiempos_bacteria = []
        for element in tiempos_bacteria:
            print(element)

    def lista_posiciones(self):
        posicion_bacteria = self.positions
        posicion_bacteria = []

    def check_position(self):
        return True
    def limites(self):
```

El método lista.timepos(self) muestra los tiempos de las bacterias.
El método lista.posiciones(self) muestra las posiciones de las bacterias.
El método check.position(self) verifica si la posición de una bacteria está dentro del contenedor.

2.3. Reproducción

Para la reproducción de las bacterias y la visualización del crecimiento se creó el siguiente código.

```
t = 0  # tiempo
lista_bacterias = []
time_lis = []
bacteria = bactery([0,0],0)
lista_bacterias.append(bacteria)

while t < t_f  :
    for bacterias in lista_bacterias:
        bacterias.actualize_time()
        if bacterias.time_reprocutcion():
```

```
            opciones = [-5,-4,-3,-2,-1,0, 1, 2, 3, 4, 5]
            x = np.random.choice(opciones)
            y = np.random.choice(opciones)
            nueva_bacteria = bactery([x,y],0)
            lista_bacterias.append(nueva_bacte:
            #print(bacterias.position)

x0 = [bacterias.position[0] for bacterias :
y0 = [bacterias.position[1] for bacterias :
tamaño_puntos = 80
color_puntos = '#58D68D'

# Crear una figura y ejes
fig, ax = plt.subplots()

# Graficar los puntos como círculos
ax.scatter(x0, y0, s=tamaño_puntos, c=colo:

# Establecer límites del gráfico
rango = 12
ax.set_xlim(-rango, rango)
ax.set_ylim(-rango, rango)
```

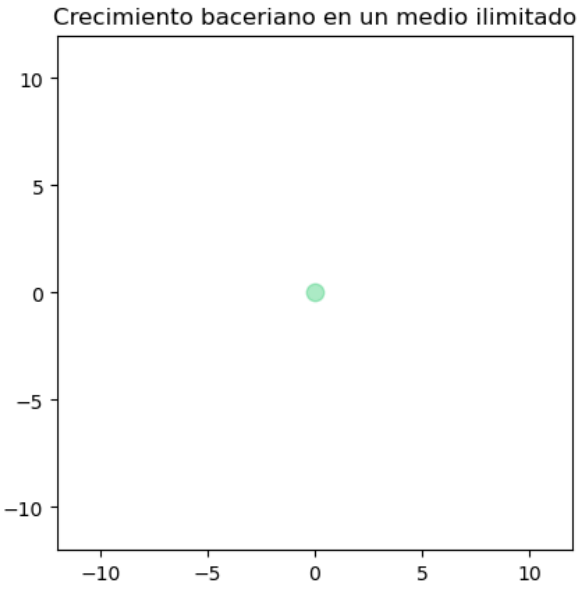

Figura 2. Bacteria inicial

```
# Configurar aspecto del gráfico
ax.set_aspect('equal')
ax.set_title('Crecimiento baceriano en un medio ilimitado')

# Mostrar el
plt.show()


l = len(lista_bacterias)
time_lis.append(l)
t = t+1
```

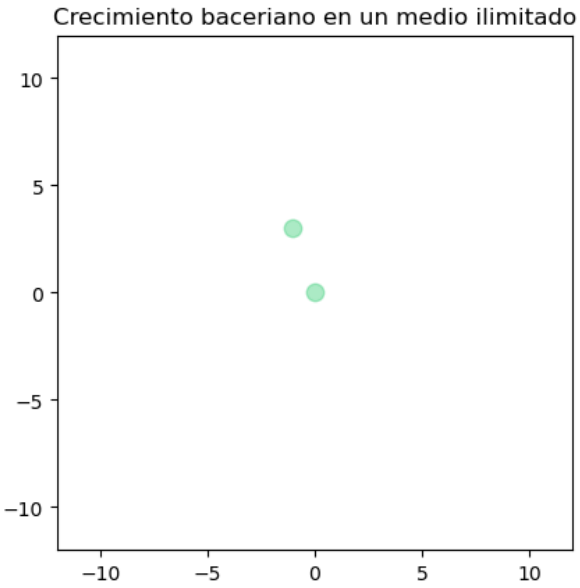

Figura 3. Bacterias en el contenedor luego de 1 ciclo de reproducción

3. Resultados

Con lo anterior se graficó el crecimeinto de las bacterias dentro de un contenedor dado de dimensiones finitas; las bacterias se ven representadas como puntos, cada ciclo de reproducción se creó una nueva visualización representando en su conjunto el crecimiento bacteriano. Además de esto, con los datos del número de bacterias a lo largo del tiempo se creó una gráfica que representa el crecimiento bacteriano de manera exponencial de la simulación.

4. Conclusiones

1. Se logró una pequeña representación del crecimiento bacteriano durante la realización de este proyecto.

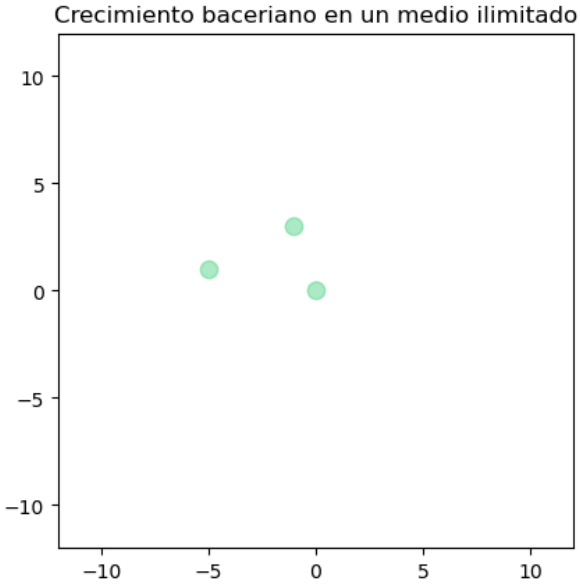

Figura 4. Bacterias en el contenedor luego de 2 ciclos de reproducción

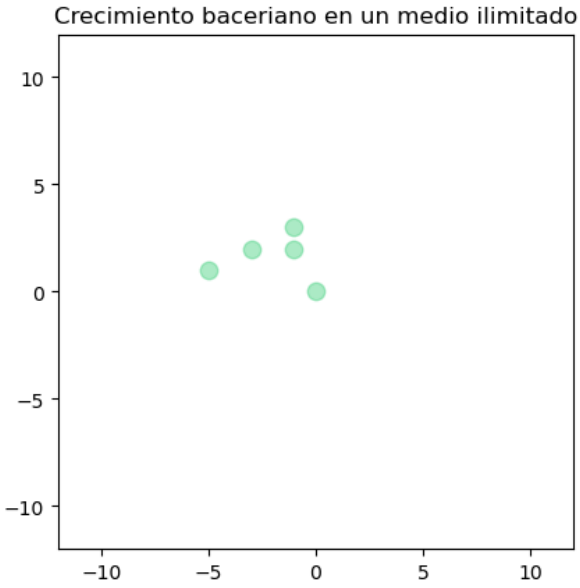

Figura 5. Bacterias en el contenedor luego de 3 ciclos de reproducción

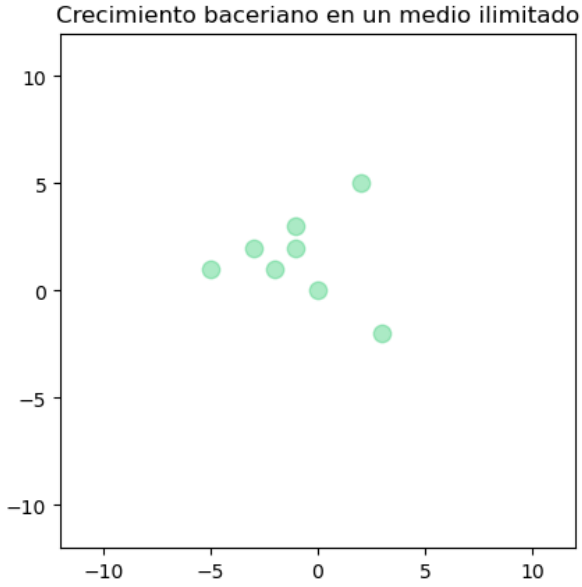

Figura 6. Bacterias en el contenedor luego de 4 ciclos de reproducción

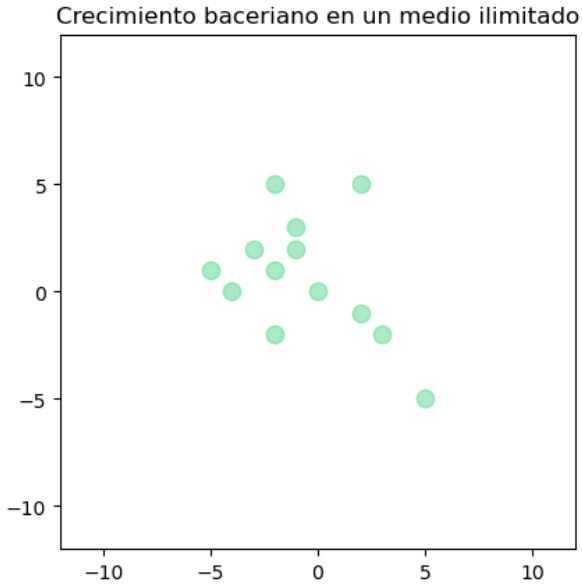

Figura 7. Bacterias en el contenedor luego de 5 ciclos de reproducción

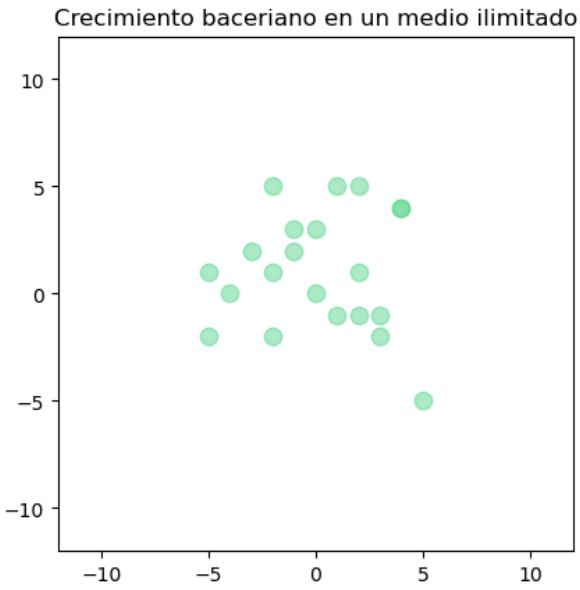

Figura 8. Bacterias en el contenedor luego de 6 ciclos de reproducción

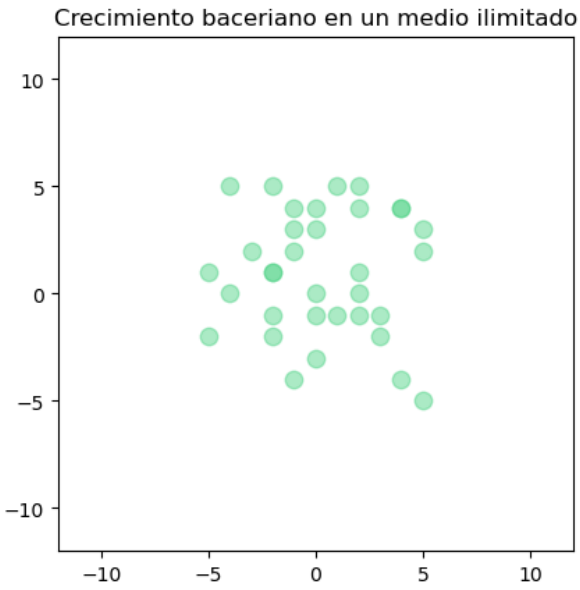

Figura 9. Bacterias en el contenedor luego de 7 ciclos de reproducción

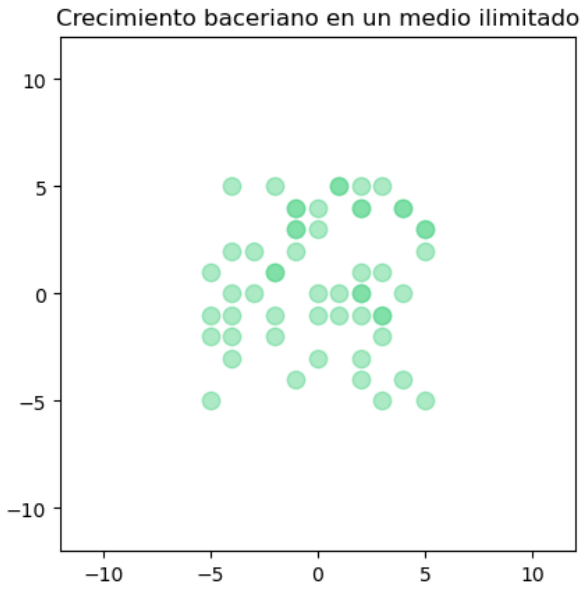

Figura 10. Bacterias en el contenedor luego de 8 ciclos de reproducción

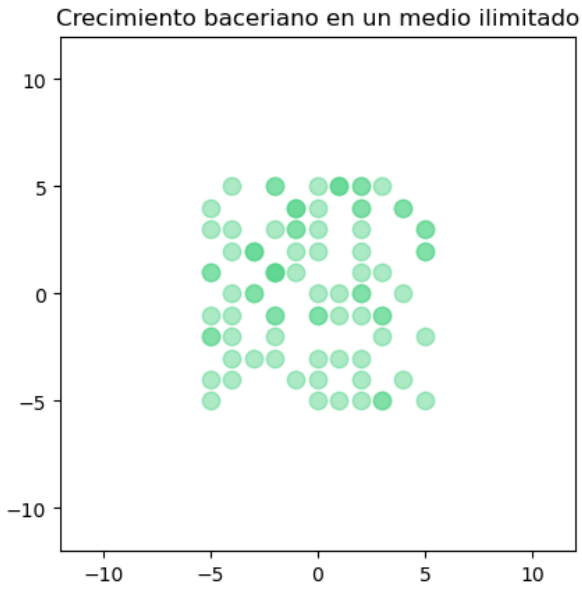

Figura 11. Bacterias en el contenedor luego de 9 ciclos de reproducción

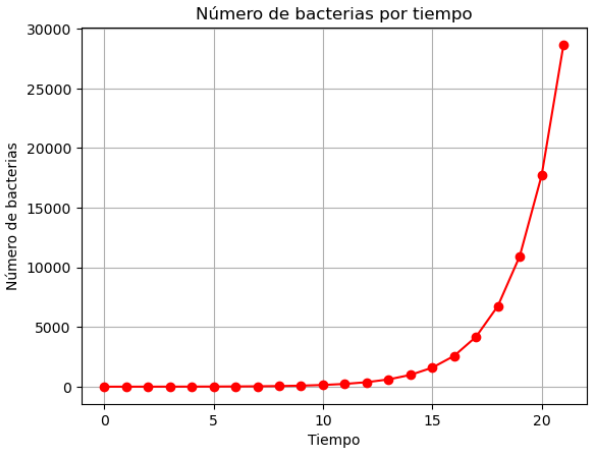

Figura 12. Gráfica crecimiento exponencial de la simulación

2. La representación gráfica del número de bacterias y la división dentro de un contenedor a lo largo del tiempo proporciona una visualización clara de los patrones de crecimiento y la dinámica poblacional.

3. El proyecto sirve como base para futuras investigaciones en el campo de la modelización y simulación del crecimiento bacteriano.

5. Trabajo a futuro

1. Crear un modelo de antibiótico que al contacto con las bacterias las elimine, revisar el comportamiento de la gráfica poblacional.

2. Al igual que con el antibiótico, agregar un modelo que simule el sistema inmune, crear interacciones correspondientes con las bacterias y ver el comportamiento de la gráfica poblacional.

3. Agregar otros factores que puedan matar a las bacterias.

4. Agregar diferentes variantes de bacterias; bacteria dañina/no dañina.

6. Referencias

Link del Repositorio de Github:
`https://github.com/Aangntonio/Bacterygrown/tree/main`