UNIVERSIDAD AUTÓNOMA DE BAJA CALIFORNIA

PROGRAMA DE POSGRADO DE MAESTRÍA Y DOCTORADO EN CIENCIAS E INGENIERÍA

Implementación de modelos matemáticos presa-depredador en robots móviles

Tesis
para cubrir parcialmente los requisitos necesarios para obtener el grado de Maestro en Ingeniería

Presenta
Néstor Alejandro Mendoza Herrera

Ensenada, Baja California, México

Agosto 2023

UNIVERSIDAD AUTÓNOMA DE BAJA CALIFORNIA

FACULTAD DE INGENIERÍA, ARQUITECTURA Y DISEÑO

MAESTRÍA Y DOCTORADO EN CIENCIAS E INGENIERÍA

Implementación de modelos matemáticos presa-depredador en robots móviles

TESIS

Que para obtener el grado de Maestría en Ingeniería presenta:

Néstor Alejandro Mendoza Herrera

Aprobada por:

Dra. Rosa Martha López Gutiérrez
Director de tesis

Dr. César Cruz Hernández
Codirector de tesis

Dr. Adrian Arellano Delgado
Miembro del comité

Dr. Rigoberto Martínez Clark
Miembro del comité

Dra. Liliana Cardoza Avendaño
Miembro del comité

Ensenada Baja California, México. Agosto 2023

Resumen de la tesis de **Néstor Alejandro Mendoza Herrera**, presentada como requisito para obtener el grado de **Maestro en Ingeniería**, del programa de Maestría y Doctorado en Ciencias e Ingeniería de la Universidad Autónoma de Baja California. Ensenada, Baja California, México, Agosto 2023.

Implementación de modelos matemáticos presa-depredador en robots móviles

Resumen aprobado por:

Dra. Rosa Martha López Gutiérrez

Director de tesis

Dr. César Cruz Hernández

Codirector de tesis

El presente trabajo de investigación, reporta un estudio y la implementación de algunas relaciones biológicas de interés en modelos presa-depredador, mediante grupos de pequeños robots móviles. La metodología adoptada para alcanzar este objetivo, fue recurrir a la teoría de sistemas complejos y al diseño e implementación de algoritmos de control distribuido. Además reporta un análisis de la adaptación de los modelos matemáticos presa-depredador, los cuales, generan dinámicas equivalentes a los comportamientos dinámicos obtenidos, cuando se emplean los grupos de robots móviles. Los resultados se muestran mediante simulaciones numéricas, utilizando **MATLAB** y **CoppeliaSim**.

Palabras clave: Modelos presa-depredador, sistemas complejos, comportamientos dinámicos, robots móviles.

Thesis **Abstract** of **Néstor Alejandro Mendoza Herrera**, presented as a requirement to obtain the degree of **Master in Engineering**, from the program of Master and Doctorate in Sciences and Engineering of Autonomous University of Baja California. Ensenada, Baja California, Mexico, August 2023.

Implementation of prey-predator mathematical models in mobile robots

Abstract approved by:

Dra. Rosa Martha López Gutiérrez

Director of thesis

Dr. César Cruz Hernández

Codirector of thesis

The present research work, reports a study and the implementation of some biological relations of interest in prey-predator models, through groups of small mobile robots. The adopted methodology to reach this objective, was to resort to the theory of complex systems and the design and implementation of distributed control algorithms. It also reports an analysis of the adaptation of the prey-predator mathematical models, which generate equivalent dynamics to the dynamic behaviors obtained, when groups of mobile robots are used. The results were show through numerical simulations, using **MATLAB** and **CoppeliaSim**.

Keywords: Prey-predator models, complex systems, collective behaviors, mobile robots.

Dedicatoria

A mis padres Alejandro y Zulma, a mis hermanas Alejandra y Valentina, a mis abuelos maternos María Vallejan y César Herrera y a mis abuelos paternos María Meza (lita) y Juan Manuel Mendoza (lito) por forjarme como soy.

Agradecimientos

A la Facultad de Ingeniería, Arquitectura y Diseño, por la instrucción recibida y el uso de sus instalaciones en beneficio de mi formación académica.

Al Consejo Nacional de Humanidades, Ciencias y Tecnologías (CONAHCyT), por brindarme el apoyo económico para realizar mis estudios de maestría. No. de becario: 1068948

Al Consejo Nacional de Humanidades, Ciencias y Tecnologías (CONAHCyT) por el apoyo económico recibido a través del proyecto de investigación en ciencia básica entre instituciones, "Sincronización de Sistemas Complejos y Algunas Aplicaciones". Ref. 166654 (A1-S-31628).

A mis directores de tesis Dra. Rosa Martha López Gutiérrez y Dr. César Cruz Hernández por su instrucción recibida a lo largo de este trabajo de investigación, sobre todo por su paciencia y confianza que inspiraron a seguir adelante en este mundo de la ciencia.

Al Dr. Adrian Arellano Delgado, Dr. Rigoberto Martínez Clark y Dra. Liliana Cardoza Avendaño, por su fuerte instrucción en la formación personal recibida en licenciatura y maestría, también en cuanto a sus comentarios recibidos para enriquecer este trabajo de investigación.

A toda mi familia por su constante apoyo y por inspirarme a seguir adelante en cada día, y brindarme el valor de jamás rendirse.

A mis amigos de posgrado Caro Medina, Marco Guzmán, José David Arroyos y Carlos Villalobos por hacer este camino más placentero.

Índice

Índice de figuras

Índice de cuadros

Capítulo 1

1. Introducción

1.1. Motivación

A lo largo de los años muchas especies en la naturaleza han sido caracterizadas por la dinámica de comportamientos que realizan para cumplir con objetivos específicos que permitan el avance de su calidad de vida, entre ellas se encuentran, por ejemplo: los bancos de peces, bandadas de aves, los conjuntos de hormigas, conjuntos de luciérnagas, manadas de animales terrestres entre muchas otras. Estas especies son beneficiadas como resultado de las necesidades biológicas para permanecer juntos (Zarzosa Gómez, 2017) y prolongar su calidad de vida.

(Zarzosa Gómez, 2017) indica que los individuos que conforman un rebaño, un banco de peces, o una bandada de aves tienen una mayor probabilidad de mantenerse con vida. Eso se debe a la influencia del trabajo colaborativo, ya que los individuos de más alto nivel en la naturaleza (depredadores) suelen atacar a individuos que se mantienen solos o aislados. La organización de las especies cuando están en grupo es de mayor aprovechamiento, debido a que al estar en conjunto suelen ser un blanco (presas) más difícil de interceptar.

El comportamiento social entre las especies se refiere a una serie de comportamientos en la que los individuos y sus parejas interactúan entre sí para su propia supervivencia y reproducción, así como también la búsqueda de alimento, la migración, la recolección, el ataque, la defensa en contra de enemigos naturales, cuidado de la descendencia, entre otros (Robinson et al., 2005). Los comportamientos colectivos o sociales entre las especies naturales son de categoría compleja, eso se debe a las múltiples interacciones que ocurren dentro de las especies cuando están

realizando alguna tarea colectiva o propósito especifico. En la naturaleza hay muchas situaciones y organismos que generan comportamientos complejos, uno de los conjuntos más estudiados son los insectos sociales (Hualong et al., 2018). Los insectos sociales viven en comunidad, donde el nido (vivienda) es muy estructurado y la división de los individuos es clara y detallada (Hualong et al., 2018).

Los insectos sociales son caracterizados por la extrema división reproductiva del trabajo. Por lo regular existe una reina, la cual se especializa en la reproducción mientras que los trabajadores participan en actividades colaborativas, así como construir el nido (vivienda), recolectar comida, criar a los insectos jóvenes y la defensa de la colonia (Wilson, 1971). La organización dentro de los grupos de los insectos sociales es extremadamente estrecha donde el medio de comunicación entre los individuos es a través de feromonas (Karlson y Butenandt, 1959).

El comportamiento colectivo de grandes agregaciones de animales es un verdadero y fascinante fenómeno natural (Parrish y Edelstein-Keshet, 1999). Particularmente interesante es el caso cuando agregaciones se autoorganizan en patrones complejos sin necesidad de un estímulo externo (Krause y Ruxton, 2002). Además de su evidente relevancia en los campos de la etología y de la biología evolutiva, el comportamiento colectivo es un concepto clave en muchos otros campos de la ciencia, incluida la teoría de control (Jadbabaie et al., 2003), economía (Cont y Bouchaud, 2000) y ciencias sociales (Helbing et al., 2000).

Como se mencionó anteriormente, los comportamientos colectivos dentro de los grupos de especies es una serie de interacciones complejas para propósitos específicos. Sin embargo, el objetivo principal de las interacciones entre los individuos es mantener la cohesión del grupo. Esta cohesión es un requerimiento biológico muy fuerte, moldeado por la pre-

sión evolutiva para la supervivencia: Los rezagados y los grupos pequeños son significativamente más propensos a la depredación que los animales pertenecientes a grandes y altamente agregaciones cohesivas (Pitcher, 1993). Un ejemplo donde se puede visualizar el concepto anterior es el siguiente: Se considera una bandada de estorninos, la cual se encuentra bajo el ataque de un halcón peregrino, la bandada se contrae, se expande e incluso se divide, cambiando continuamente su densidad y su estructura. Ningún ave permanece aislada, posteriormente la bandada se reforma como un todo (Ballerini et al., 2007).

El ejemplo anterior sirve como referencia para conocer el comportamiento colectivo que realizan las aves ante el ataque de un depredador. La comunicación es un elemento clave durante el proceso de la dinámica que realizan para evadir el ataque del depredador, ya que permite que las aves puedan interactuar entre ellas mismas junto con sus vecinos para realizar una forma determinada, así como expandirla, reducirla (aumento y reducción de las distancias entre los individuos del grupo), dividirla, entre otras aptitudes, que les permitan ser una presa complicada para su depredador.

Comportamientos como los antes mencionados existen con gran abundancia en los diferentes campos de la ciencia, desde las visualizaciones en las ciencias naturales hasta las aplicaciones en las ciencias ingenieriles. Un aspecto muy importante que conecta a ambos conceptos es que en los comportamientos colectivos desde el punto de vista de las ciencias naturales son una referencia para poder reproducirlos en sistemas ingenieriles y realizar tareas colectivas que aporten un beneficio para la sociedad.

La reproducción de sistemas naturales en sistemas ingenieriles ha sido un tema muy abordado por especialistas en los temas de investigación, debido a que no solo se utiliza los conceptos de ciencias naturales pa-

ra el estudio de dicho trabajo, sino que también engloba herramientas físico-matemáticas que permiten la programación del trabajo, visualizadas desde alguna computadora o un equipo de ayuda que esté al alcance del usuario.

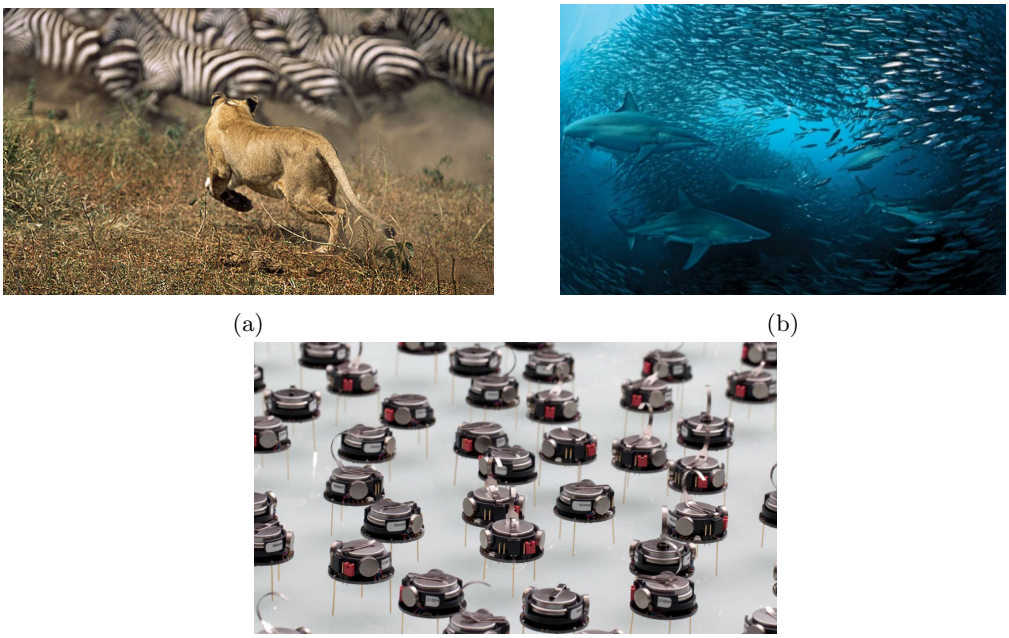
Así como hay comportamientos colectivos que se visualizan en los sistemas naturales, también hay los que son visualizados en la sociedad, y la relación que ambos mantienen es muy similar. La vida en la sociedad es a menudo altamente estructurada, con casi todas las actividades influenciadas por interacciones con otros miembros de la sociedad. La regulación social influye en cuándo, con qué frecuencia, con qué intensidad y con quienes realizan estas actividades (Robinson et al., 2005). Así como las especies sociales naturales, las personas que conforman la sociedad se rigen con los mismos principios complejos para realizar actividades que individualmente serían más complicadas que hacerlo de manera colectiva, y la manera de realizarlo es a partir de una serie de interacciones que se dan en los grupos de personas.

Las técnicas colectivas que utilizan los individuos en cualquier ámbito (natural, social, etc.) forman parte de una teoría que describe porque los individuos recurren a estas técnicas cuando es necesario, dejando completamente de lado que se trate de un asunto meramente aleatorio o de pura coincidencia, dicha teoría es conocida *teoría compleja*, o, dicho de otra forma, *teoría de los sistemas complejos*. (Parrish y Edelstein-Keshet, 1999) mencionan que dicha teoría indica que grandes poblaciones de unidades pueden autoorganizarse en agregaciones que generan patrones, almacenan información y participan en la toma colectiva de decisiones. El ejemplo mencionado por (Ballerini et al., 2008) donde ilustraba que una bandada de estorninos estaba bajo el ataque de un halcón peregrino, la cual se contrae, se expande, se divide, cambiando su forma, es un ejemplo claro de la *teoría compleja* ya que menciona a una gran agregación

donde los individuos se autoorganizan y toman las decisiones de cambiar la estructura de la bandada para evadir la depredación de un halcón peregrino. De esta forma se llegó a un beneficio para los estorninos, el cual fue la prolongación de su calidad de vida, resultado de que haya existido comunicación y organización entre los integrantes del grupo.

1.2. Planteamiento del problema

En el presente trabajo de investigación se aborda el problema de implementar algunos modelos biológicos presa-depredador en grupos de robots móviles llamados kilobots utilizando el esquema de máquinas de estado finito empleando algoritmos de control distribuido, es decir, que cada robot utilice la misma estructura de código (algoritmo).



(a) (b)

(c)

Figura 1: Problema de investigación, implementación de comportamientos presa-depredador basados en modelos matemáticos en grupos de kilobots. (a) Grupo de cebras escapando de un depredador. Fuente: https://noticiascuriosas.com/ las-rayas-de-los-animales-muestran-al-grupo-en-que-direccion-huir/. (b) Cardumen de sardinas ante el ataque de varios depredadores. Fuente: http://blogdefenomenosfisicos.blogspot.com/2019/01/cardumen-de-sardinas.html. (c) Grupo de kilobots, los cuales mostrarán dinamicas presa-depredador. Fuente: https://www.sheffield.ac.uk/sheffieldrobotics/projects/swarm-awareness.

En la figura 1 se observan algunos ejemplos de relaciones de presas y depredadores, en (a) y en (b) se observan relaciones naturales de este comportamiento complejo, mientras que en (c) se observan la unidades robóticas que adoptarán esas interacciones complejas.

1.3. Objetivos

Con la realización de esta investigación, se pretendió alcanzar el siguiente **objetivo general**:

1.3.1. Objetivo general

Implementar algunos modelos biológios presa-depredador en enjambres robóticos mediante sistemas complejos y algoritmos de control distribuido.

Mediante alcanzar los **objetivos particulares**:

1.3.2. Objetivos particulares

- *Objetivo particular 1*: Estudiar diferentes relaciones presa-depredador inspirados en la naturaleza.
- *Objetivo particular 2*: Analizar relaciones presa-depredador con dos y tres especies.
- *Objetivo particular 3*: Implementar en Matlab algunos modelos matemáticos de presa-depredador.
- *Objetivo particular 4*: Estudiar las características de operación del robot kilobot (comunicación, alcances, limitantes, movimiento, etc.) y su implementación en CoppeliaSim.
- *Objetivo particular 5*: Implementar los algoritmos de control distribuido en grupos de robots móviles kilobots.

1.4. Hipótesis

Es posible implementar algunas relaciones biológicas presa-depredador en enjambres de pequeños robots móviles, mediante la teoría de sistemas complejos y control distribuido.

1.5. Antecedentes

En esta sección se mencionarán algunos trabajos relacionados con el propósito principal de este trabajo de tesis, con la intención de dar a conocer parte del conocimiento existente que se ha realizado a lo largo de los años.

En el trabajo realizado por (Ordaz-Rivas et al., 2021) se define a un enjambre robótico como un gran grupo de agentes relativamente simples y autónomos que pueden mostrar comportamiento de inteligencia colectiva. A diferencia de sistemas multi-agente (Zhu et al., 2017), los enjambres robóticos están inspirados por el comportamiento observados en los organismos naturales, así como las hormigas, las aves, los peces, entre otros. Por otra parte (Zhu et al., 2017) menciona en su trabajo que un enjambre robótico se refiere a un grupo de agentes autoorganizados que intentan realizar tareas conjuntamente. Dicho concepto tiene su origen en muchos fenómenos biológicos observados.

En el trabajo de (Ordaz-Rivas et al., 2021) se presenta un modelo de estudio para el comportamiento en enjambres al realizar una tarea colectiva, el comportamiento a estudiar fue relacionado a pruebas presa-depredador. En este mismo trabajo se hace mención que la relación presa-depredador es un comportamiento colectivo donde un grupo de robots (depredadores) intentan capturar a una presa , la cual es otro robot que tiene ese rol en ese momento. Es una prueba cooperativa porque uno, dos o incluso tres robots no pueden atrapar a las presas por sí solos, se requiere de cooperación para lograrlo. El modelo estudiado fue basado en el

trabajo de (Bara y Dale, 2009) el cual incluía la cinemática y la dinámica de cada robot en el enjambre. El algoritmo de trabajo solo depende de cuatro parámetros relacionados a la dinámica de los animales sociales, llamado repulsión, atracción, orientación e influencia (RAOI). El objetivo de este trabajo fue estimular las propiedades del enjambre mediante los parámetros de RAOI y cuantificar las características del comportamiento colaborativo, mostrando resultados a través de simulaciones por computadora e implementación experimental en robots móviles con ruedas.

Por otro lado se tiene el trabajo de (Dimidov et al., 2016) el cual consistió en realizar estrategias de búsquedas de objetivos partir de caminatas aleatorias implementadas en kilobots. En este trabajo se hace mención que las caminatas o paseos aleatorios representan estrategias de búsqueda fundamentales tanto en animales como en robots, sobre todo cuando no hay señales que puedan impulsar el movimiento o cuando las capacidades cognitivas del agente no permitan un comportamiento complejo.

El propósito de (Dimidov et al., 2016) fue analizar la eficacia de los patrones de una caminata aleatoria para un enjambre de kilobots que buscan un objetivo estático en dos condiciones ambientales diferentes, un espacio acotado o un espacio abierto. Para llevar acabo dicho propósito se hizo un estudio a cerca de la eficiencia de búsqueda y la capacidad de difundir información dentro del enjambre a través de simulaciones numéricas y experimentos con robots reales. Los experimentos realizados con los kilobots fueron hechos solamente en el escenario acotado en una arena de trabajo cuadrada de 90 cm de lado, donde los robots y el objetivo fueron distribuidos en la arena de manera uniforme. El objetivo a encontrar fue representado por un kilobot inmóvil, programado de tal forma que emitiera de forma continua un código de identificación.

Además, los kilobots se comunican entre sí y comparten información sobre el descubrimiento del objetivo emitiendo un código de identificación, ya sea cuando un robot ha descubierto al objetivo por su cuenta o cuando un robot ha recibido la información de otro robot.

En el trabajo realizado por (Carrasco Gutiérrez et al., 2020) se estudió la relación presa-depredador a través del modelo dinámico de Lotka-Volterra. Sin embargo, en este trabajo se consideró una variación del modelo antes mencionado para el diseño de un algoritmo de máquina de estado finito que pudiera representar el fenómeno presa-depredador en un grupo de robots móviles de bajo costo llamados kilobots, incluyendo efectos de desplazamiento entre los agentes y competencia intraespecífica entre los depredadores con el fin de aumentar el realismo en el algortimo de implementación en el grupo de robots móviles. Para demostrar los resultados obtenidos se utilizó el software V-REP, ahora conocido como CoppeliaSim, con la intención de evaluar el algoritmo implementado a través de simulaciones numéricas.

Por otro lado en (Pelkonen, 2018) se realizó una representación del comportamiento del sistema inmunólogico humano utilizando kilobots en el simulador V-REP, ahora más conocido como CoppeliaSim. En su trabajo (Pelkonen, 2018) consideró a los kilobots como células inmunes y al ambiente como una concentración antígena, con alta concentración de patógenos, donde el objetivo principal es que los kilobots realicen la búsqueda y destrucción de los patógenos en el área. El autor utilizó algoritmos de control descentralizado para hacer de los kilobots un sistema robusto, flexible y escalable. El autor también utilizó sincronización de comunicación entre MATLAB y V-REP (CoppeliaSim) para hacer funcionar los kilobots y el medio ambiente al mismo tiempo.

Después se tiene el trabajo de (Sidón Ayala, 2017) quien menciona que en su trabajo que la *robótica en enjambre* ha abierto un nuevo campo de investigación dentro de robótica en sí, la cual se encuentra inspirada principalmente en el comportamiento observable de los insectos sociales. El autor menciona que diferentes robots son capaces de interactuar entre sí, generando comportamientos emergentes visualizados en poblaciones de insectos sociales (abejas, hormigas, aves, entre otros). En el trabajo realizado por (Sidón Ayala, 2017) se utilizaron los kilobots para conformar enjambres robóticos que permitan el estudio de diversos comportamientos emergentes, algunos ejemplos de estos comportamientos fueron sincronización mediante destellos de luz, formación de robots y elección de líder. Cada comportamiento fue realizado utilizando esquemas de máquinas de estado finito para que los kilobots pudieran determinar las acciones a ejecutar y pudieran actuar de acuerdo a la situación de cada caso.

Las diferencias entre los trabajos previos y el presente trabajo de investigación radican en que se hicieron estudios sobre algunas relaciones biológicas, representadas por modelos matemáticos, para posteriormente adaptarlos en algoritmos compuestos por máquinas de estado finito e implementarlos en robots móviles llamados kilobots, con el propósito de visualizar las interacciones (comportamientos emergentes) en un entorno simulado.

Capítulo 2

2. Sistemas complejos

Antes de explicar algunas de las características que describen a los sistemas complejos es de vital importancia, explicar algunos aspectos previos que conectan con la ciencia de este tipo de sistemas, con la intención de tener una comprensión más detallada, uno de dichos aspectos es el término *sistema*.

El término *sistema* usualmente se refiere a objetos, cosas o entidades (tanto concretas como abstractas) para lo cual es posible identificar algún tipo de limite para que el sistema pueda ser distinguido por su exterior. (Por su puesto, esta es una definición genérica), por lo tanto, una vez que se ha identificado el sistema es posible decir que pertenece a él y que no, es decir, lo que esta fuera de su limite (entorno) (Roli, 2015).

Por otro lado (Ponce Muñoz, 2009) explica que un *sistema* es un conjunto de elementos interactuando entre sí con un propósito, ello implica que las partes componentes ejercen una influencia sobre los demás elementos repercutiendo en ellos inevitablemente. Un *sistema*, es entonces más que la simple suma de sus componentes; lo anterior genera dos efectos: por una parte la interacción de las partes generará nuevas propiedades en el sistema, diferentes a las de sus componentes, situación que se le denomina propiedades emergentes. Un ejemplo sobre lo anterior lo constituye un equipo de sonido, para lograr el sonido se requiere la conjunción de sus componentes, pero éstos por separado no son capaces de generar el sonido.

Después de haber hecho un análisis sobre lo que es un sistema, es el momento profundizar un poco más en el tema, explicando uno de los términos más populares que conectan con este trabajo de investigación, dicho término son los *sistemas complejos*.

La ciencia de los *sistemas complejos* es un campo científico bastante reciente que estudia como las partes de un sistema dan lugar a los comportamientos colectivos del sistema, y como éste interactúa con su entorno. Se enfoca en ciertas preguntas sobre partes, totalidades y relaciones. La ciencia de los sistemas complejos es el corpus de teorías y métodos que permiten ayudar a lidiar con los sistemas complejos. Algunos ejemplos de este tipo de sistemas son el cerebro, la sociedad, el ecosistema, la célula, las colonias de hormigas, la bolsa de valores, entre otros. No existe una definición formal de un sistema complejo pero se puede proporcionar una definición informal al decir que los sistemas complejos son caracterizados por algunas de las siguientes propiedades:

- Compuesto por muchos elementos.
- Interacciones no lineales.
- Topología de red.
- Realimentaciones positivas y negativas.
- Adaptativo y evolutivo.
- Robusto.
- Niveles de organización.

La ciencia de los sistemas complejos tiene el objetivo del estudio y modelado, y control de los sistemas del mismo tipo. Es interdisciplinaria y envuelve muchas disciplinas, como las matemáticas, física, ciencias

computacionales, biología, economía, filosofía, neurología y más.

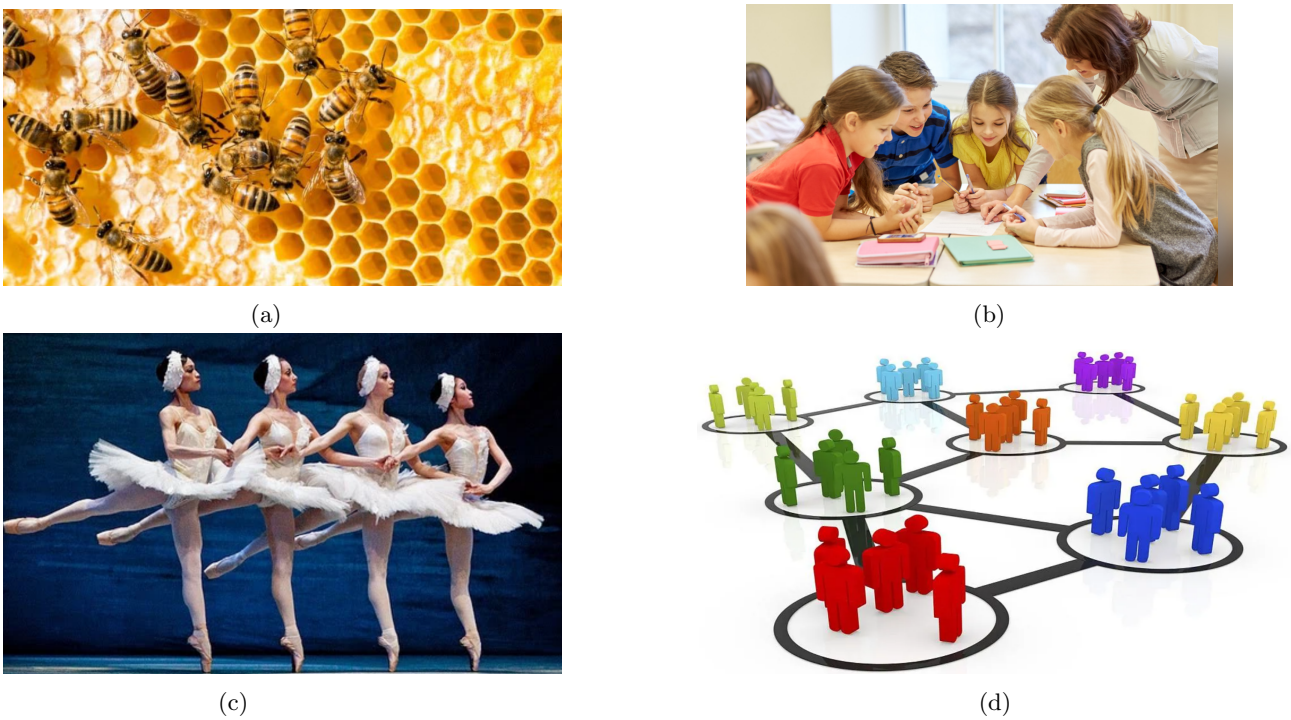
Una peculiaridad de los sistemas complejos es que las reglas locales a menudo propagan información de tal manera que todo el sistema está sujeto a dinámicas que no es posible comprender a partir de los constituyentes del sistema. De echo a menudo se dice que el comportamiento global emerge en un sistema complejo de las interacciones de sus partes, gobernado por normas locales. (Roli, 2015).

Las ciencias de la complejidad estudian fenómenos, sistemas o comportamientos de complejidad *creciente*; esto es, fenómenos y sistemas que aprenden y se adaptan, y que, en el filo del caos o bien, lo que es equivalente, lejos del equilibrio, responden a la flecha del tiempo de la termodinámica del no-equilibrio (Nicholis y Prigogine, 1994). Por consiguiente, *à la lettre*, no se ocupan de todos y cada uno de los fenómenos y sistemas del mundo, puesto que no todas las cosas son complejas y en numerosas ocasiones es incluso deseable que no lo sean o que no se vuelvan o se hagan complejas. Vale decir, que en el más riguroso de los sentidos de lo mejor de la filosofía de la ciencia que una teoría que lo explica todo no explica nada, y así, afirmar que "todo es complejo" equivale a una idea trivial, análoga acaso a la numerología o astrología (Maldonado, 2014).

En una forma compacta, los sistemas complejos pueden describirse como una clase de problemas donde:

- La cantidad de variables en interacción es muy grande.
- La interacción de variables sea poca, pero, desde el punto de vista matemático o físico, el tipo de interacciones sean no lineales.
- El conocimiento de las partes de un fenómeno no sea suficiente para

conocer y explicar su comportamiento al integrarse como un todo.



(a) (b) (c) (d)

Figura 2: Ejemplos de sistemas complejos en la vida cotidiana. (a) Un grupo de abejas construyendo su hogar. Fuente: https://concepto.de/abejas/. (b) Un grupo de niños trabajando en equipo en una tarea escolar. Fuente: https://www.educalinkapp.com/blog/trabajo-en-equipo/. (c) Baile en sincronía por un grupo de bailarinas de ballet. Fuente: https://www.balaio.com.ar/ballet-clasico/. (d) Ejemplo de varios grupos de personas interactuando en una sociedad. Fuente: https://libremercado-medioambiente.ufm.edu/2017/12/13/la-sociedad-es-un-sistema-dinamico-y-complejo/.

La figura 2 muestra algunos ejemplos de sistemas complejos que pueden observarse en la vida cotidiana. Por ejemplo, en (a) se observa un grupo de abejas haciendo trabajo colectivo para construir su hogar, en (b) se observa un grupo de niños trabajando en equipo para hacer una tarea escolar, en (c) se observa un grupo de bailarinas haciendo una coreografía de un baile en un tiempo simultáneo, mostrando un comportamiento de sincronización, en (d) se observa una sociedad donde diferentes grupos de personas interactúan entre ellas para promover dinámicas y relaciones complejas, donde a su vez esos grupos de personas pueden llegar a afectar a otros grupos, es decir, que existe una interconexión entre los grupos de una sociedad.

Nótese que la similitud que existe entre los sistemas mostrados en la figura 2 es la presencia de grupos de agentes, ya sean de animales, como se observa en (a) o de personas, como se observa en (b), (c) y (d), y como en conjunto generan interacciones locales para lograr un propósito específico.

Capítulo 3

3. Modelos de relaciones presa-depredador

Un aspecto importante a considerar en este trabajo de investigación es que se utilizará la teoría de los sistemas complejos para conocer de una forma científica una de las relaciones de la ciencia básica que ayudará a contribuir para el análisis de los comportamientos que se buscan obtener, dicha relación es conocida como *presa-depredador*.

La relación *presa-depredador* consiste en la interacción de dos o más especies donde una de ellas posee nivel trófico más alto (depredador) que la otra (presa) en un ambiente. En dicha relación se observa un fenómeno conocido como *depredación*, el cual actúa relativamente de manera aislada con otras especies, haciendo que sus poblaciones conserven un ciclo regular (Khalaf, 2016). La forma de un ciclo regular puede observarse en la dinámica de un modelo que involucra la relación presa-depredador, dicho modelo es conocido como modelo de *Lotka-Volterra*.

3.1. Modelo de *Lotka-Volterra*

El modelo básico de *Lotka-Volterra* fue propuesto de manera independiente por el matemático americano Alfred Lotka y el matemático italiano Vito Volterra en 1925 y 1926, de manera respectiva. El modelo incluye una serie de suposiciones que lo simplifican. Primeramente, el modelo asume que la población de la presa tiene suministro alimenticio ilimitado. La segunda suposición es que el depredador es la única amenaza de la presa, y por lo tanto cualquier disminución en la población de la presa esta relacionada con la depredación. La siguiente suposición es que la presa es el único suministro alimenticio del depredador, y que el crecimiento del depredador depende enteramente de la cantidad presas capturadas. Por lo tanto, cualquier incremento en la población del de-

predador esta relacionada con la depredación. Adicionalmente se asume que la tasa de encuentros entre los depredadores y las presas es conjuntamente proporcional al tamaño de las dos poblaciones. Esta suposición de unión es representada por los términos pxy y dxy en el sistema de ecuaciones diferenciales. Finalmente se asume una proporción constante de encuentros entre depredadores y presas que conduce a la muerte de las presas. Con todas esas suposiciones se simplifica el modelo de *Lotka-Volterra*, el cual puede ser reconstruido como un sistema de ecuaciones diferenciales reducido (Von Arb, 2013). A continuación, se mostrará el modelo respectivo de *Lotka-Volterra*:

$$\frac{dx}{dt} = bx - pxy \tag{1}$$

$$\frac{dy}{dt} = dxy - ry \tag{2}$$

(Von Arb, 2013) indica que la ecuación (1) representa a la población de las presas, en el tiempo t, la cual se denota como $x(t)$, y la ecuación (2) representa a la población de los depredadores, en el tiempo t, la cual se denota como $y(t)$. Por lo tanto $\frac{dx}{dt}$ representa el cambio en la población de las presas, x, a medida que avanza el tiempo, y $\frac{dy}{dt}$ representa el cambio en la población de los depredadores, y, a medida que avanza el tiempo.

Otro dato importante que es necesario definir es el significado de cada una de las variables que se encuentran en el modelo de *Lotka-Volterra*. (Von Arb, 2013) menciona el significado de cada variable, las cuales son b y p en la ecuación (1) y d y r en la ecuación (2). En este sistema de ecuaciones diferenciales, el parámetro b representa la tasa de crecimiento

de las presas (especie x) en la ausencia de la interacción con el depredador (especie y). Por otro lado p y d son los parámetros de los dos términos de interacción, donde p representa el efecto de la depredación de la especie y sobre la especie x, mientras que d es la tasa de crecimiento de las especies y en condiciones perfectas: presas abundantes sin impacto negativo del medio ambiente. Finalmente, r es la tasa de muerte de las especies y por causas naturales.

Una vez que se ha comprendido el concepto de las variables es posible obtener la respuesta de la dinámica temporal del modelo, junto con el atractor que genera mediante la asignación de los datos que se le agreguen a cada variable y a sus condiciones iniciales. Para ello se hará uso de los datos del trabajo de (Kumar, 2019) y también se hará uso del software *MATLAB* para darle solución al sistemas de ecuaciones diferenciales de *Lotka-Volterra* con el propósito de poder visualizar los gráficos referentes a la dinámica temporal y al atractor en su plano de fase.

(Kumar, 2019) establece en su trabajo los parámetros de entrada, para ingresarlos de manera correcta en el modelo de *Lotka-Volterra*. Los valores seleccionados por el autor fueron los siguientes: $b = 1.1$, $p = 0.4$, $d = 0.1$ y $r = 0.4$, mientras que los valores de las condiciones iniciales fueron $x(0) = 10$ y $y(0) = 10$, también el vector del tiempo abarca desde un valor inicial igual a cero ($t_0 = 0$) hasta un valor final igual a 100 ($t_f = 100$). (Ver anexo 9.1 Sistema de Lotka-Volterra). Para obtener el gráfico correspondiente a la dinámica temporal del modelo de *Lotka-Volterra* se utilizó un software llamado *MATLAB*, en el cual se pudo ingresar cada parámetro junto con el modelo para obtener su respuesta. A continuación se mostrará el gráfico resultante:

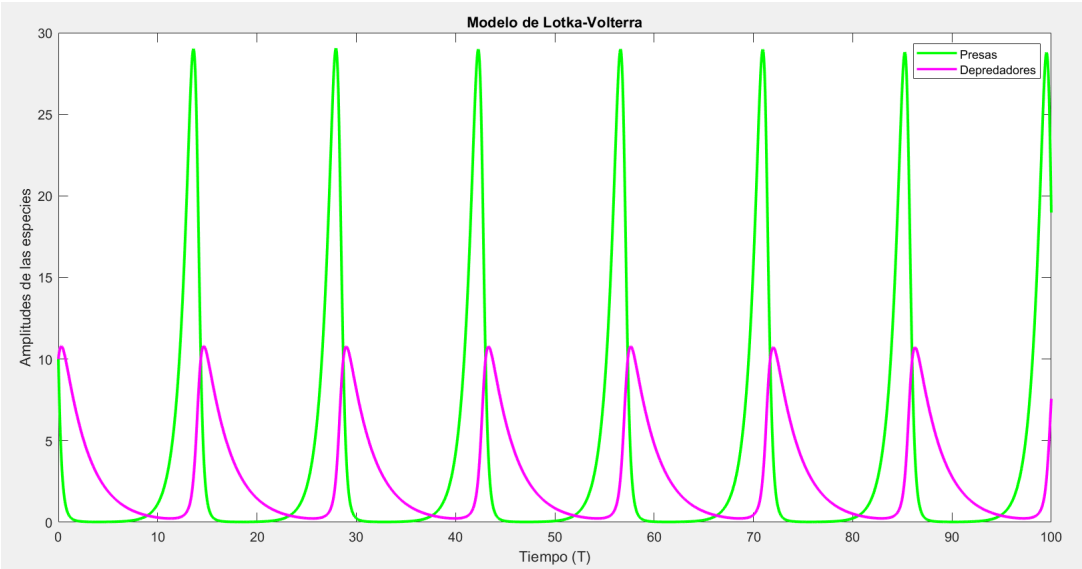


Figura 3: Modelo de interacción presa-depredador de *Lotka-Volterra*, mediante los datos de (Kumar, 2019). El código puede encontrarse en el anexo a este trabajo de tesis. Anexo 9.1 Sistema de Lotka-Volterra.

En la figura 3 se puede observar la dinámica de población de las especies del modelo de *Lotka-Volterra*, donde se puede notar que un aumento lo suficientemente grande en el número de los depredadores conduce a una disminución en el número de las presas. Esto es lógico desde un punto de vista biológico, ya que una población mayor de depredadores conduce a un aumento en las interacciones entre los depredadores y las presas, y por lo tanto aumenta la muerte de las presas. Esto también es lógico basado desde el punto de vista matemático, buscando de nuevo en el sistema de ecuaciones diferenciales, se puede ver que un incremento en el número de los depredadores conducirá a una disminución en $\frac{dx}{dt}$, el cambio en la población de las presas a medida que el tiempo avanza. Como la población de presas disminuye, la población de los depredadores también comienza a disminuir, ya que el aumento en la población de los depredadores no se puede sostener debido a la disminución del número de presas. A medida que la población de los depredadores disminuye, la población de las presas comienza a recuperarse. Una vez que la población de las presas se ha recuperado lo suficiente, la población de los depredadores vuelve a aumentar. Esto lleva al punto de partida una vez más,

haciéndolo un patrón periódico, el cual es común en todas las relaciones presa-depredador (Von Arb, 2013).

Otro gráfico importante, ademas de la serie temporal, es el plano de fase, en el cual se puede reflejar los tamaños de la población de las presas, respecto a la población de los depredadores, sin la inclusión del tiempo. A continuación, se mostrará el plano de fase correspondiente a la serie temporal obtenida:

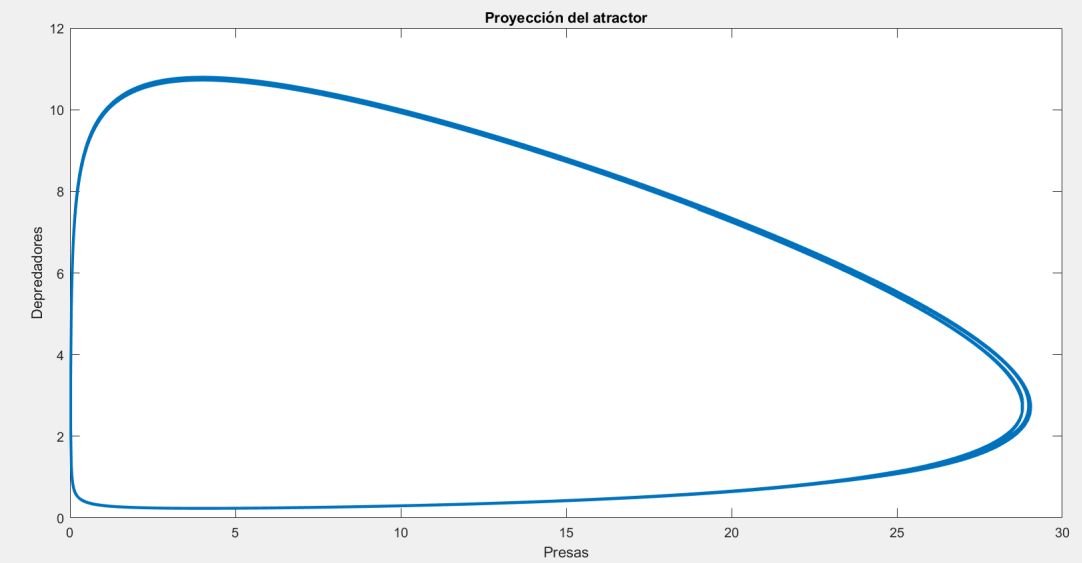


Figura 4: Atractor del modelo de *Lotka-Volterra*, basado en los datos de (Kumar, 2019). El código puede encontrarse en el anexo a este trabajo de tesis. Anexo 9.1 Sistema de Lotka-Volterra.

La figura 4 hace referencia al plano de fase del modelo de *Lotka-Volterra* utilizando el mismo conjunto de parametros con los cuales se obtuvo la dinámica temporal. (Von Arb, 2013) explica que el plano de fase adquiere una forma redonda, debido a que la población de los depredadores incrementa poco después de la población de las presas, causando que la población de las presas disminuya y rápidamente causando que la población de los depredadores disminuya también. En este punto la forma redonda regresa a donde comenzó.

3.2. Modelo de dos especies de depredadores y una especie de presas

Después de haber conocido la dinámica de las especies para el caso de interacción con una presa y un depredador, es el momento de realizar un análisis similar al modelo anterior, con la diferencia que en este caso se incluirá a una nueva especie, es decir, un modelo de tres especies. El caso especifico es de *dos especies de depredadores y una especie de presas*. La razón de emplear un sistema de tres especies es porque representa a una de las relaciones biológicas tanto populares como complejas, debido a que una nueva especie conlleva a el análisis de una nueva variable dependiente y su manifestación bajo la influencia de interacciones de dos espacies más.

El modelo de dos especies de depredadores y una especie de presas es una variante del modelo de *Lotka-Volterra*, en el cual se presenta una situación en la cual dos poblaciones de depredadores están presentes y ambas atacan a una sola especie de presas como fuente de alimento primaria. Este sistema se compone de tres ecuaciones diferenciales. Una ecuación diferencial representa el cambio en la población de cada una de las poblaciones presentes en el sistema a medida que avanza el tiempo. En este caso se define la población de presas en el tiempo t como $x(t)$, la población del primer depredador en el mismo instante de tiempo se define como $y(t)$ y la población del segundo depredador en el mismo instante de tiempo se define como $z(t)$. La forma más compacta de poder escribir el sistema es modelar las ecuaciones de la siguiente manera:

$$\frac{dx}{dt} = ax - bxy - cxz \tag{3}$$

$$\frac{dy}{dt} = dxy - ey \tag{4}$$

$$\frac{dz}{dt} = fxz - gz \tag{5}$$

En la ecuación (3) del modelo se encuentran diversas variables, donde a representa la tasa de incremento de las presas, b representa la tasa de disminución de las presas en respuesta al encuentro con el depredador 1 y c representa la tasa de disminución de las presas en respuesta al encuentro con el depredador 2. En la ecuación (4) del modelo se encuentran las variables d que representa el crecimiento del depredador 1 en respuesta al encuentro con las presas, y e que representa la tasa de mortalidad del depredador 1 en respuesta a la ausencia de las presas. En la ecuación (5) del modelo se encuentran las variables f que representa la tasa de crecimiento del depredador 2 en respuesta al encuentro con las presas, y g que representa la tasa de mortalidad del depredador 2 en respuesta a la ausencia de las presas (Von Arb, 2013).

Para conocer la respuesta del sistema de tres especies se utilizará nuevamente el software llamado $MATLAB$, para observar su comportamiento temporal y su respuesta en el espacio de fase, para seleccionar los valores de los parámetros de entrada y de las condiciones iniciales se elegirán de manera aleatoria. A continuación, se mostrarán los comportamientos correspondientes al sistema de tres especies:

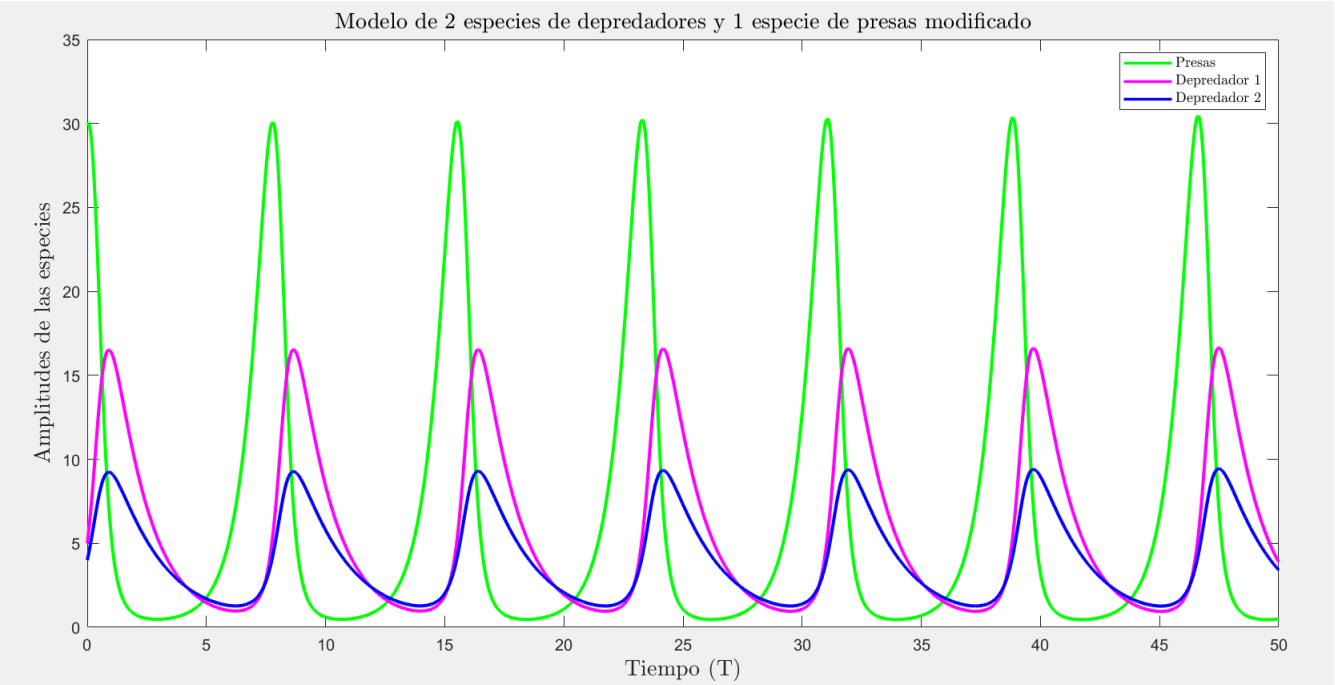


Figura 5: Dinámica temporal del modelo de tres especies, para el caso de dos especies de depredadores y una especie de presas. El código puede encontrarse en el anexo a este trabajo de tesis. Anexo 9.2 Sistema de dos especies de depredadores y una especie de presas.

En la figura 5 puede observarse que la respuesta temporal es muy similar a la del modelo básico de *Lotka-Volterra*, debido a que el movimiento del comportamiento es oscilatorio para cada especie y mientras el tiempo avanza seguirán conservando el mismo comportamiento. Una de las características por las que eso pasa es debido a la selección de los parámetros de entrada y a las condiciones iniciales, los valores seleccionados fueron los siguientes: $a = 1.5$, $b = 0.2$, $c = 0.1$, $d = 0.1$, $e = 0.715$, $f = 0.07$, $g = 0.5$, $x(0) = 30$, $y(0) = 5$, $z(0) = 4$, $t_0 = 0$ y $t_f = 50$. (Ver anexo 9.2 Sistema de dos especies de depredadores y una especie de presas). Lo que está ocurriendo en el gráfico es que la presa esta bajo la influencia de dos depredadores, por esa razón no alcanza a tener un crecimiento exponencial, también se observa que la población del depredador 1 esta ligeramente por encima que la población del depredador 2, eso es debido a que su tasa de encuentros con la presa es ligeramente superior a la del depredador 2 ($d = 0.1$ y $f = 0.07$, respectivamente), a pesar de que el depredador 1 tiene una tasa de mortalidad más elevada

que el depredador 2 ($e = 0.715$ y $g = 0.5$, respectivamente) no es lo suficientemente grande para que alcance su descenso.

La dinámica anterior también puede representarse mediante su atractor correspondiente en el espacio de fase para los datos seleccionados. A continuación, se mostrará que forma fue la que generó:

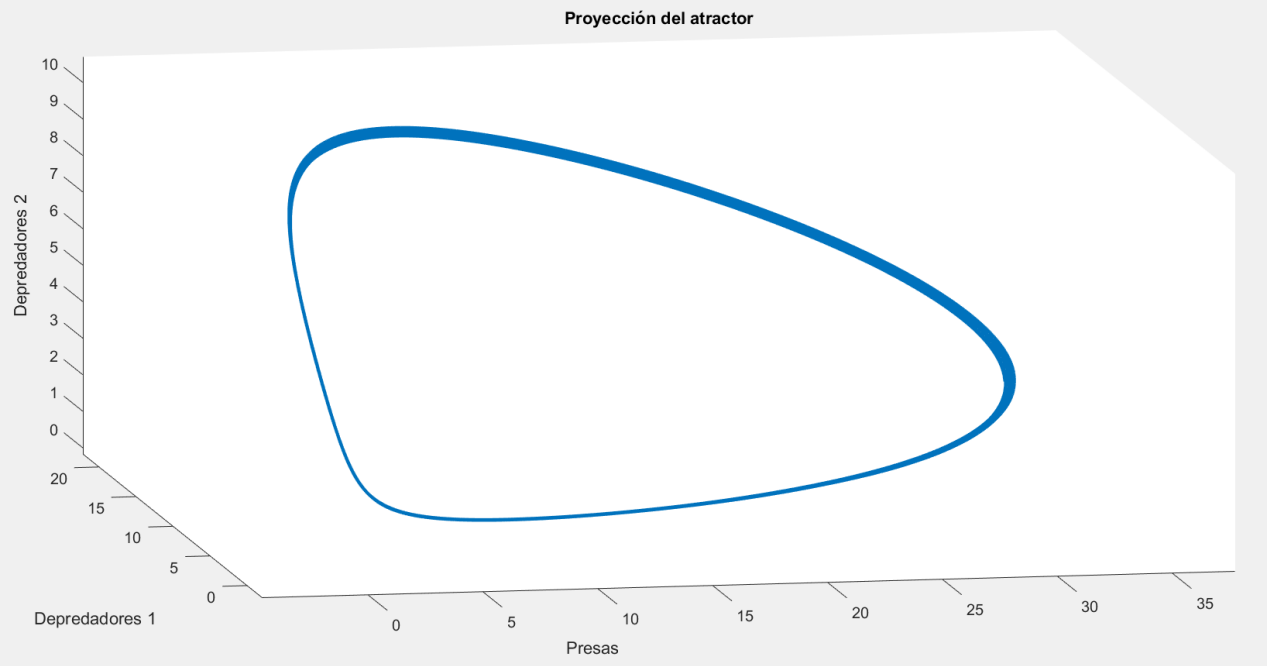


Figura 6: Atractor referente al modelo de dos especies de depredadores y una especie de presas. El código puede encontrarse en el anexo a este trabajo de tesis. (Anexo 9.2 Sistema de dos especies de depredadores y una especie de presas).

En la figura 6 se puede observar que el atractor generado es muy similar al del atractor del modelo de *Lotka-Volterra* por la razón que el movimiento de su serie temporal es oscilatorio a medida que el tiempo avanza, por lo que se espera que adquiera una forma redonda, debido que se repite el patrón de la depredación.

3.3. Sistema de dos especies de presas y una especie de depredadores

En este momento ya se tiene una noción sobre lo que sucede con los sistemas de tres especies y se ha visto como se pueden comportar sus poblaciones a medida que el tiempo transcurre. El procedimiento que se

realizará ahora será muy similar al modelo del caso anterior, sin embargo el caso a presentar es nuevo. El análisis que se planteará será para un modelo de tres especies, al igual que el caso anterior, con la variante de que ahora se obtendrán los comportamientos para el caso de *dos especies de presas y una especie de depredadores* (Khalaf, 2016).

En base a la metodología descrita en el trabajo de (Khalaf, 2016), a la metodología descrita en el trabajo de (Von Arb, 2013) y a la forma en la que ella modeló el sistema de *dos especies de depredadores y una especie de presas* es posible reescribir el sistema de *dos especies de presas y una especie de depredadores* con una estructura equivalente a la mostrada en (Von Arb, 2013). Dicha estructura se puede representar de la siguiente manera:

$$\frac{dx}{dt} = e_1xy + e_2xz - dx \tag{6}$$

$$\frac{dy}{dt} = k_1y - c_1xy \tag{7}$$

$$\frac{dz}{dt} = k_2z - c_2xz \tag{8}$$

En la estructura del modelo anterior, x representa la población de los depredadores en el tiempo t, y representa la población de la primer especie de presas en el mismo instante de tiempo t y z representa la población de la segunda especie de presas en el mismo instante de tiempo t. En la ecuación (6) que muestra el cambio de los depredadores mientras el tiempo avanza, la formación de la ecuación se conforma mediante la diferencia entre la tasa de natalidad y la tasa de mortalidad, la cual incrementa a e_1y+e_2z-d, explicado de otra forma, los parámetros e_1 y e_2 corresponden a las tasas de aumento en la población de los depredadores debido a que hacen referencia a las interacciones entre el depredador y la

presa 1, y el depredador y la presa 2, respectivamente, y d corresponde a la tasa de disminución en la población de los depredadores debido a la falta de interacciones con las presas por su ausencia. La ecuación (7) que muestra el cambio en la primer población de presas mientras el tiempo avanza, se conforma de igual manera que la ecuación (6), mediante la tasa de natalidad menos la tasa de mortalidad, donde el parámetro k_1 corresponde a la tasa de aumento de las presas en ausencia de la población de depredadores y el parámetro c_1 corresponde al descenso en la primer población de presas a causa de los encuentros con los depredadores. Para la ecuación (8) su forma es equivalente a la ecuación (7) debido que se trata también de una población de presas, en ese sentido, el parámetro k_2 corresponde al aumento en la segunda población de presas debido a la ausencia de interacciones con la población de depredadores y el parámetro c_2 corresponde a la tasa de disminución de la segunda población de presas por la presencia de interacción con los depredadores.

Para obtener la dinámica de los comportamientos del modelo anterior se utilizará *MATLAB*, como en los ejemplos previos, y además se utilizará una propuesta de datos aleatorios para obtener la respuesta del sistema de la relación *dos especies de presas y una especie de depredadores*. Los datos utilizados fueron los siguientes: $e_1 = 0.008$, $e_2 = 0.009$, $d = 0.7$, $k_1 = 0.9$, $c_1 = 0.009$, $k_2 = 1$, $c_2 = 0.01$, $t_0 = 0$ (tiempo inicial), $t_f = 50$ (tiempo final), los valores de las condiciones iniciales son $x(0) = 30$, $y(0) = 20$ y $z(0) = 5$. (Ver anexo 9.3 Sistema de dos especies de presas y una especie de depredadores).

A continuación, se mostrará primeramente la serie temporal del modelo correspondiente al caso de *dos especies de presas y una especie de depredadores*:

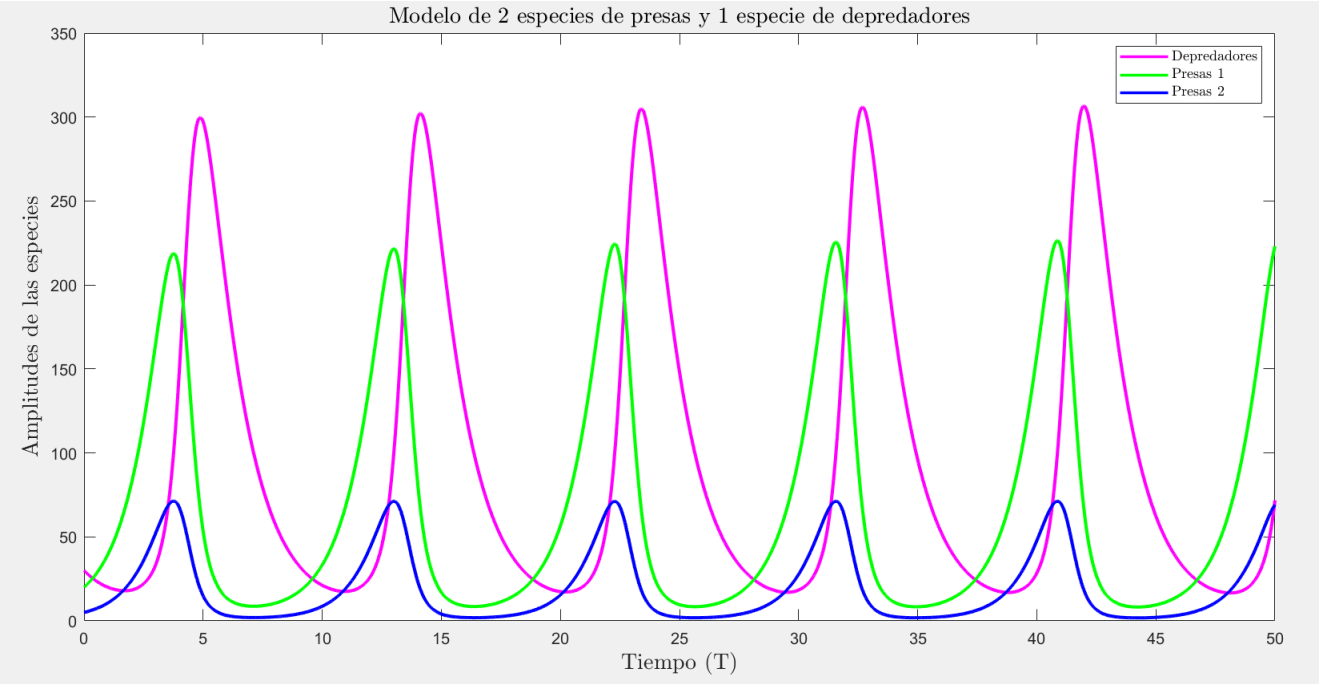


Figura 7: Dinámica temporal del modelo de tres especies, para el caso de dos especies de presas y una especie de depredadores. El código puede encontrarse en el anexo a este trabajo de tesis. Anexo 9.3 Sistema de dos especies de presas y una especie de depredadores.

En la figura 7 puede observarse que la gráfica es muy similar en cuanto a la de los modelos de *Lotka-Volterra* y de *dos especies de depredadores y una especie de presas*, ya que se muestran que los comportamientos de las especies son periódicos, al igual que los casos anteriores. A diferencia de los casos anteriores se observa que la población de los depredadores asciende más en relación al lapso temporal, eso se debe a que ahora hay dos suministros de fuente de alimento primaria para ellos, posterior a ese ascenso las dos poblaciones de presas descienden debido a que la cantidad de depredadores es muy grande, lo que afecta de forma negativa al depredador y provoca su descenso, posteriormente al descenso de los depredadores las poblaciones de presas se recuperan a como estaban anteriormente y a partir de ese momento en adelante los comportamientos se repiten.

Después de mostrar la dinámica temporal del sistema anterior se procederá a mostrar el atractor del mismo sistema para comprobar como se

manifiesta respecto a los sistemas anteriores:

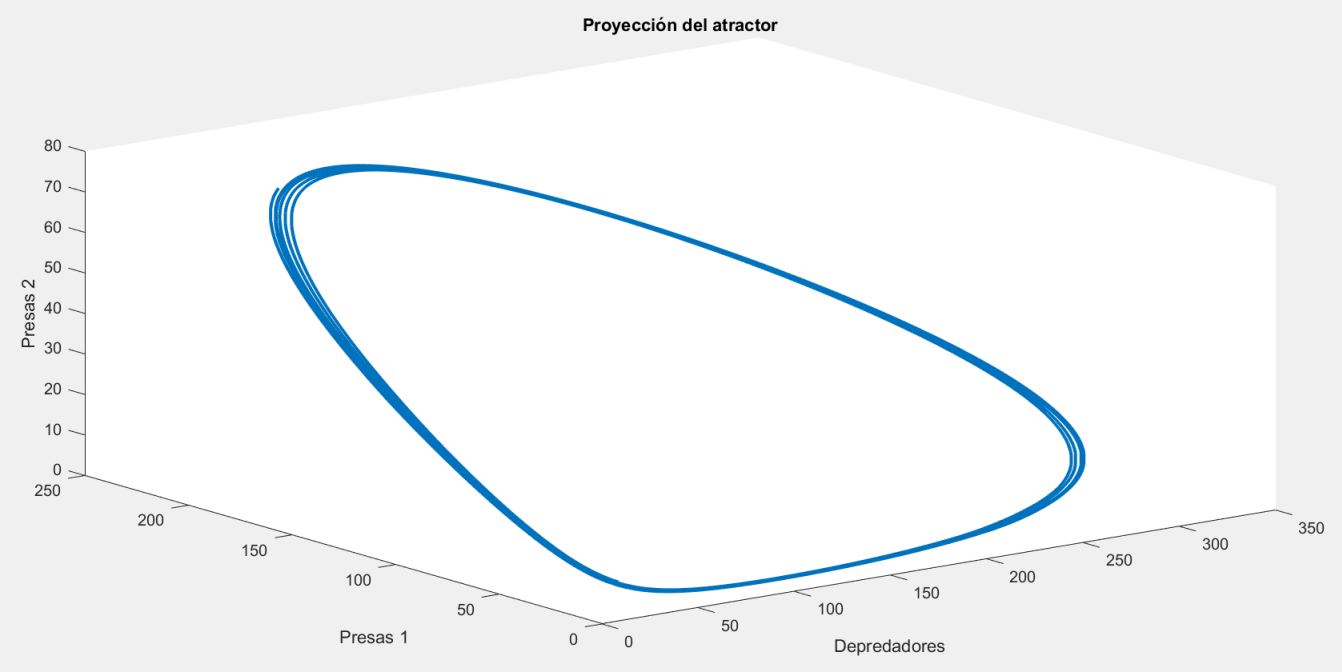


Figura 8: Atractor referente al modelo de dos especies de presas y una especie de depredadores. Anexo 9.3 Sistema de dos especies de presas y una especie de depredadores.

En la figura 8 se puede observar que el atractor tiene una forma redonda, eso se debe a la misma razón de los sistemas anteriores, cada comportamiento dinámico de los sistemas tienen características periódicas, haciendo alusión a ciclos periódicos, donde después de un periodo de tiempo el comportamiento se repite, por esa razón la proyección del atractor tiene forma redonda, aunque no representa un circulo perfecto por la razón de que la dinámica de las especies tienen diferentes frecuencias.

3.4. Modelos matemáticos modificados

Con la intención de agregar más realismo a los modelos matemáticos es necesario contemplar ciertas interacciones que se puede dar en las relaciones biológicas observadas en la naturaleza, dichas interacciones corresponden al nombre de "competencia intraespecífica" y "competencia interespecífica" (Oganician, 2017), quien define a este par de interacciones cuando los individuos compiten por un recurso escaso, que a su vez puede ser compartido por dos o más especies. Otra estrategia por la cual fueron agregadas dichas interacciones fue porque permiten regular el tamaño de las poblaciones depredadoras.

3.4.1. Modelo de Lotka-Volterra modificado

A diferencia del modelo original, que se había visto con anterioridad, el modelo de *Lotka-Volterra* puede ser modificado al agregar términos de competencia entre depredadores, esto puede verse reflejado con robots móviles, cuando dos o más robots interactúan entre depredadores, ocasionando una disminución en su población en el transcurso del tiempo. El modelo modificado puede observarse de la siguiente manera:

$$\frac{dx}{dt} = bx - pxy \tag{9}$$

$$\frac{dy}{dt} = dxy - ry - ky^2 \tag{10}$$

Anteriormente se había explicado el modelo original, sin embargo se observa que la ecuación (10) tiene una ligera diferencia en comparación con el modelo original, el termino ky^2 hace referencia a la competencia entre depredadores, el parámetro k hace referencia a la intensidad de la competencia entre los depredadores sobre la dinámica temporal mientras que la variable y^2 hace referencia a que hubo un encuentro entre depredadores.

El modelo de *Lotka-Volterra* modificado es el que será implementado en el grupo de robots móviles, debido a que posee más realismo que el original, además que permite regular la población depredadora, permitiendo asemejar las dinámicas al modelo original.

A continuación se mostrará una representación de la dinámica de poblaciones cuando éstas son afectadas por un factor de competencia en la población de los depredadores, obteniéndose el siguiente gráfico:

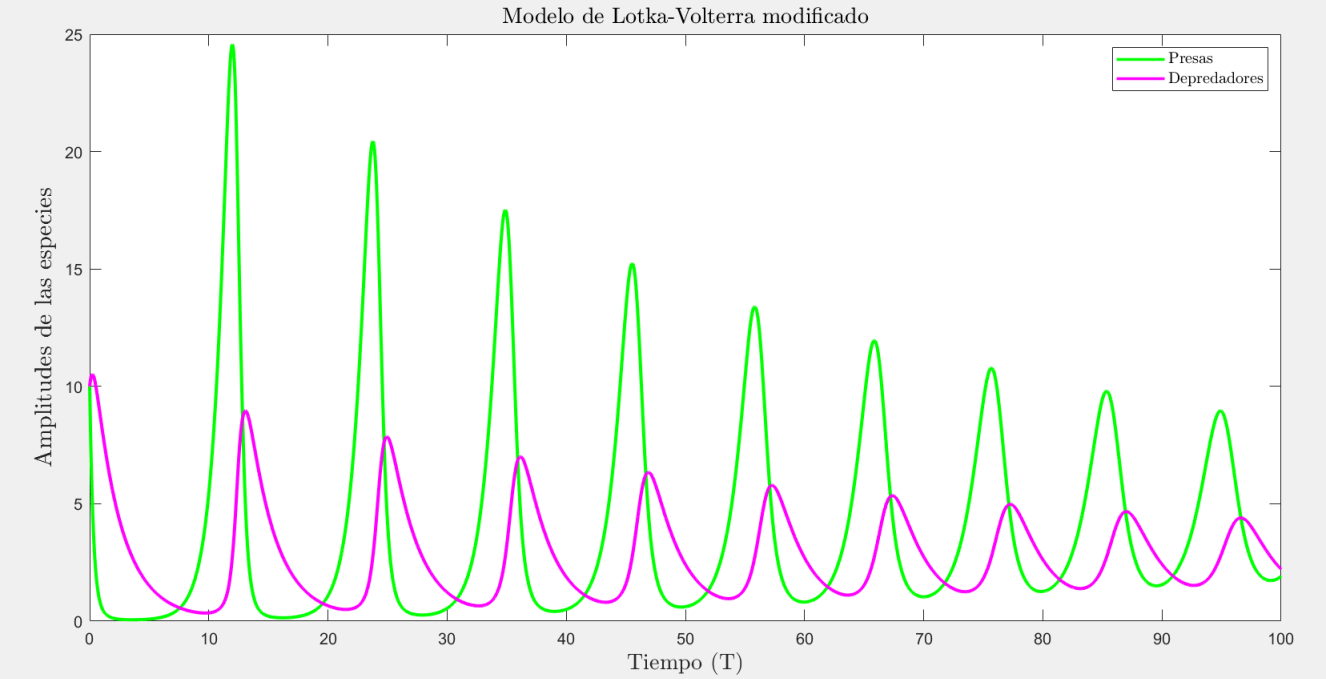


Figura 9: Comportamiento del modelo de *Lotka-Volterra* con competencia intraespecífica entre depredadores. Anexo 9.4 Sistema de Lotka-Volterra modificado.

La figura 9 muestra como se observa el comportamiento del modelo de *Lotka-Volterra* agregando el efecto de competencia entre depredadores. En este caso, a diferencia del modelo original, se observa que los comportamientos de las especies están disminuyendo su amplitud en cada ciclo que avanza el tiempo, parecido al efecto de una cascada, eso se debe a la reducción de la eficiencia en la caza del depredador, es decir, que entre más ineficiente sea la especie depredadora (mayor cantidad de la tasa de competencia) las interacciones con las presas serán menores

en los siguientes instantes de tiempo, lo que a su vez también limita el crecimiento de la población depredadora. Para alcanzar los datos de la simulación anterior se hizo uso de los mismos datos que en el modelo original, a continuación se mencionarán: b= 1.1, $p = 0.4$, $d = 0.1$, $r = 0.4$ y $k = 0.01$ (k es el parámetro de competencia), mientras que las condiciones iniciales también fueron las mismas, $x(0) = 10$ y $y(0) = 10$, de igual manera el vector de tiempo fue desde 0 hasta 100 unidades de tiempo, es decir $t_0 = 0$ y $t_f = 100$. (Ver anexo 9.4 Sistema de Lotka-Volterra modificado).

Posteriormente se anexará el plano fase (atractor) de la dinámica temporal anterior correspondiente a la figura 9 para observar el comportamiento que adquiere:

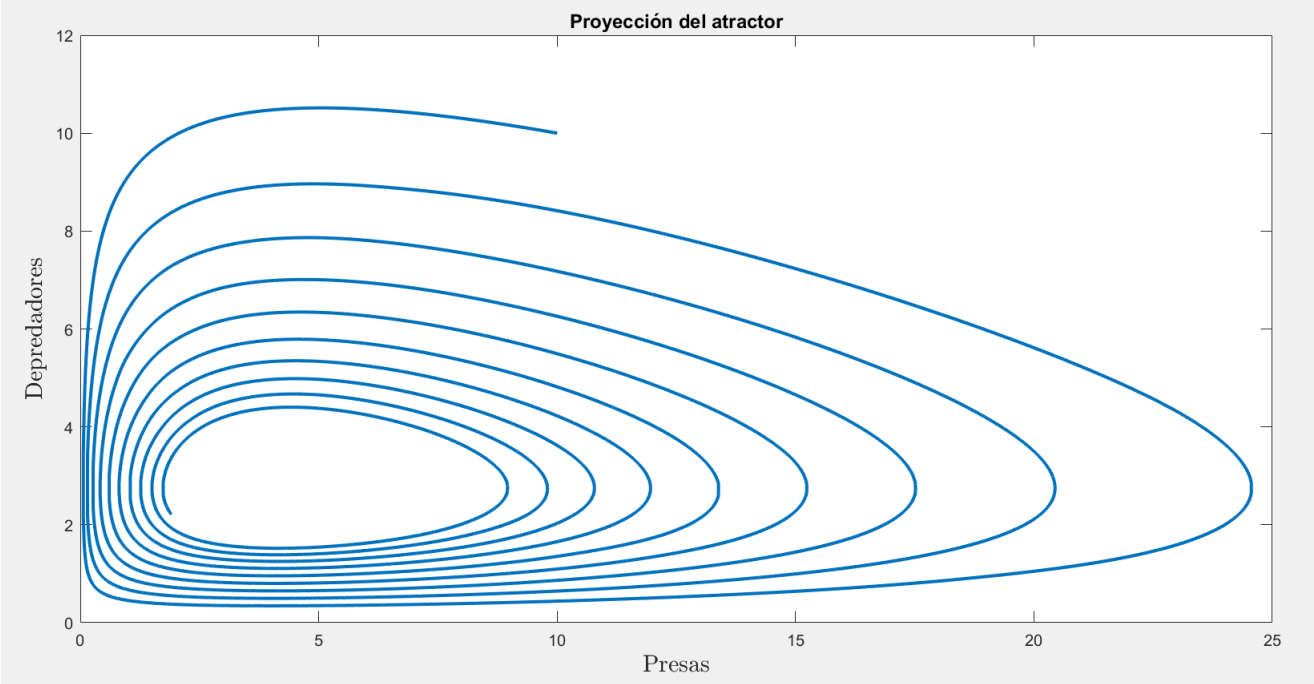


Figura 10: Atractor del modelo de *Lotka-Volterra* con competencia intraespecífica entre depredadores. Anexo 9.4 Sistema de Lotka-Volterra modificado.

La figura 10 muestra como se comporta el atractor en relación a la dinámica temporal del modelo de *Lotka-Volterra* modificado, se observa como una especie de espiral que gira de afuera hacia adentro tratando de

llegar a un cierto punto, lo que esta sucediendo es que el atractor muestra que las poblaciones de ambas especies están siendo reducidas en cada ciclo que ocurre, similar a lo que sucede en la dinámica temporal, lo cual es debido a la modificación del modelo explicada anteriormente, la cual ocasiona que la población depredadora sea más ineficiente, ocasionando que tengan encuentros con menores cantidades de presas.

En la visualización con robots móviles (kilobots) el factor de competencia anterior es contempla debido a que funciona como agente regulador de los depredadores, permitiendo la reproducción de las presas y generando la estabilización a través de ciclos oscilatorios en condiciones no ideales.

3.4.2. Modelo de 2 especies de depredadores y 1 especie de presas modificado

De la misma manera como se explicó el modelo anterior, en la aplicación del algoritmo de control en robots móviles de este modelo matemático se hará una modificación para la situación en la que se presente interacción por parte de la población depredadora. Obteniendo la siguiente forma matemática:

$$\frac{dx}{dt} = ax - bxy - cxz \tag{11}$$

$$\frac{dy}{dt} = dxy - ey - k_1 y^2 - k_2 yz \tag{12}$$

$$\frac{dz}{dt} = fxz - gz - k_3 z^2 - k_4 zy \tag{13}$$

El conjunto de ecuaciones anteriores corresponde al modelo matemático modificado del escenario *2 poblaciones de depredadores y 1 población de presas*, esta es la forma en la que se implementará en los grupos de robots móviles con el motivo de aumentar el realismo en los conjuntos

de interacciones que puedan presentar los kilobots, y regular las poblaciones de depredadores cuando hayan muchos integrantes de ellas.

La ecuación (11) no presentó ningún cambio respecto al modelo original, pero las ecuaciones (12) y (13) si presentaron cambios, se presentan dos términos nuevos en cada ecuación. En (12) se observa el término k_1y^2 el cual hace referencia a la tasa de competencia que ocurre entre depredadores del grupo y, el término k_2yz hace referencia a la interacción entre depredadores de la población y y depredadores de la población z. Por otro lado en (13) se encuentran los términos k_3z^2 que hace referencia a las interacciones entre depredadores del grupo z y el término k_4zy el cual hace referencia a que hay interacciones entre las dos poblaciones de depredadores z y y a una taza de proporción k_4. A continuación, se anexará una figura en la cual se podrá observar la dinámica temporal con los cambios recibidos de los efectos de competencia entre depredadores:

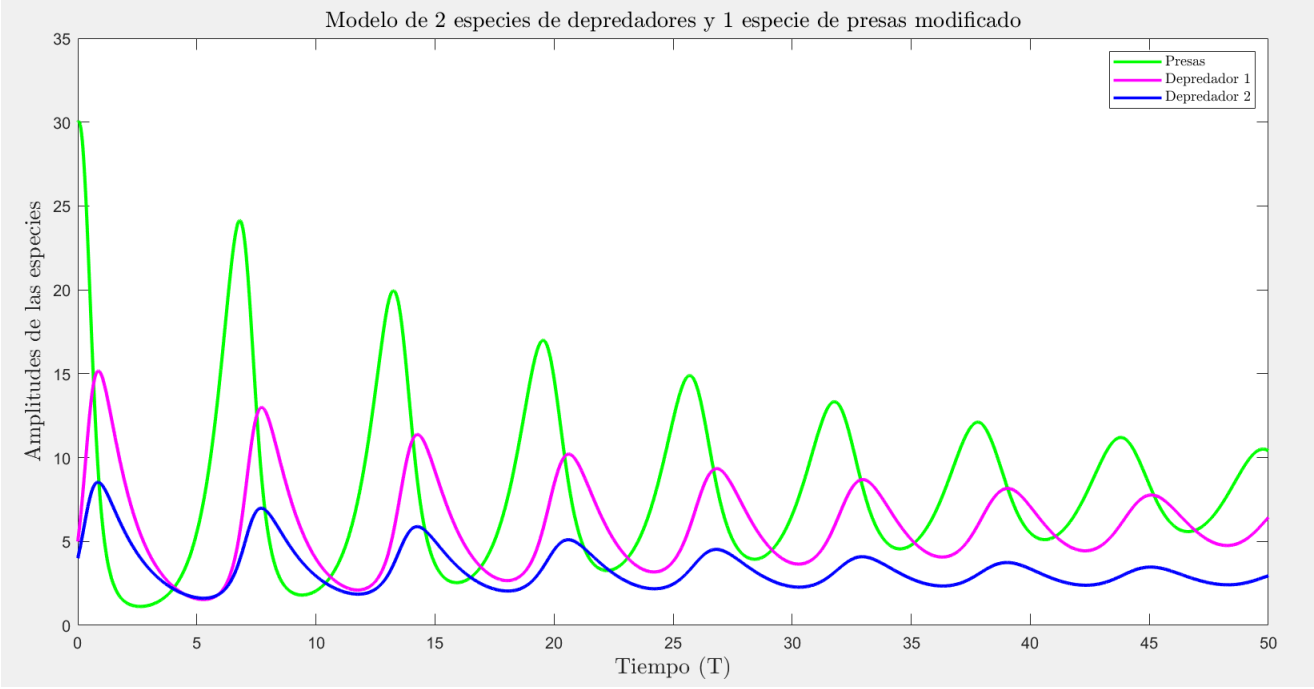


Figura 11: Dinámica temporal del modelo de dos especies de depredadores y una especie de presas modificado. Anexo 9.5 Sistema de dos especies de depredadores y una especie de presas modificado.

En la figura 11 se observa un comportamiento similar con el modelo de *Lotka-Volterra* modificado, en relación a que las tres especies sufren un descenso en las amplitudes en su dinámica temporal, el efecto que genera tal suceso en la dinámica anterior son los términos de competencia entre depredadores que fueron agregados al modelo modificado, entre más grande sean los efectos de competencia más se limitará el crecimiento de los depredadores y permitirá una regulación más rápida del sistema. Los parámetros agregados al sistema fueron los siguientes: $a = 1.5$, $b = 0.2$, $c = 0.1$, $d = 0.1$, $e = 0.715$, $f = 0.07$, $g = 0.5$, los factores de competencia fueron $k_1 = 0.01$, $k_2 = 0.005$, $k_3 = 0.01$ y $k_4 = 0.005$, las condiciones iniciales fueron $x(0) = 30$, $y(0) = 5$ y $z(0) = 4$, y el vector de tiempo fue desde $t_0 = 0$ hasta $t_f = 50$ unidades de tiempo. (Ver anexo 9.5 Sistema de dos especies de depredadores y una especie de presas modificado.)

Aunado a lo anterior se anexará el atractor del modelo modificado anterior:

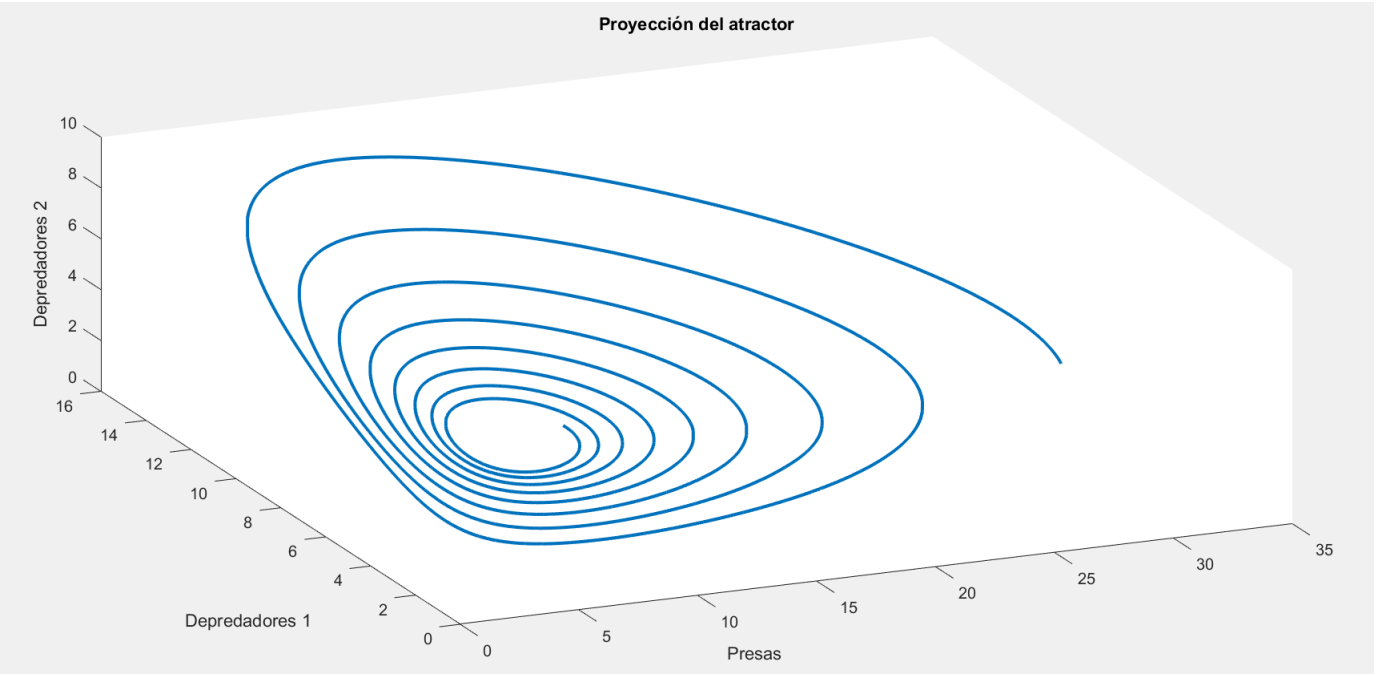


Figura 12: Atractor del modelo de dos especies de depredadores y una especie de presas modificado. Anexo 9.5 Sistema de dos especies de depredadores y una especie de presas modificado.

La figura 12 muestra como se comporta la proyección del atractor del modelo explicado anteriormente, en el cual se observa un comportamiento similar al atractor de *Lotka-Volterra* modificado, pareciendo que converge a un punto contenido en el espacio, dicho suceso ocurre debido a que mientras avanza el tiempo las amplitudes de las especies están siendo menores con cada ciclo, por la razón que se explicaba anteriormente, entre más competencia exista más ineficiente son los depredadores, por lo que las interacciones con las presas son menores, las cuales son reflejadas en el atractor.

Este modelo modificado es el que será implementado en los kilobots, ya que los efectos de competencia permiten regular la población de los depredadores cuando hay muchos de éstos, lo cual generará una mejor respuesta a la del modelo original utilizando los grupos de robots móviles.

3.4.3. Modelo de 2 especies de presas y 1 especie de depredadores modificado

Por otra parte en este modelo se presentará una situación equivalente a las anteriores, añadiendo los términos de competencia al modelo original, en esta ocasión la configuración modificada del modelo matemático de *2 poblaciones de presas y 1 población de depredadores* quedaría de la siguiente manera:

$$\frac{dx}{dt} = e_1xy + e_2xz - dx - Tx^2 \tag{14}$$

$$\frac{dy}{dt} = k_1y - c_1xy \tag{15}$$

$$\frac{dz}{dt} = k_2z - c_2xz \tag{16}$$

La ecuación (14) es la única que tuvo una modificación, añadiendo el término de competencia entre depredadores, en esta ocasión el término

Tx^2 hace referencia a la competencia o interacción entre depredadores del grupo x, perjudicando a esta población a medida que el valor de T aumente, mientras que las ecuaciones (15) y (16) permanecen iguales a las del modelo original. A continuación se mostrará la dinámica de poblaciones que el efecto de competencia entre depredadores genera sobre el modelo modificado:

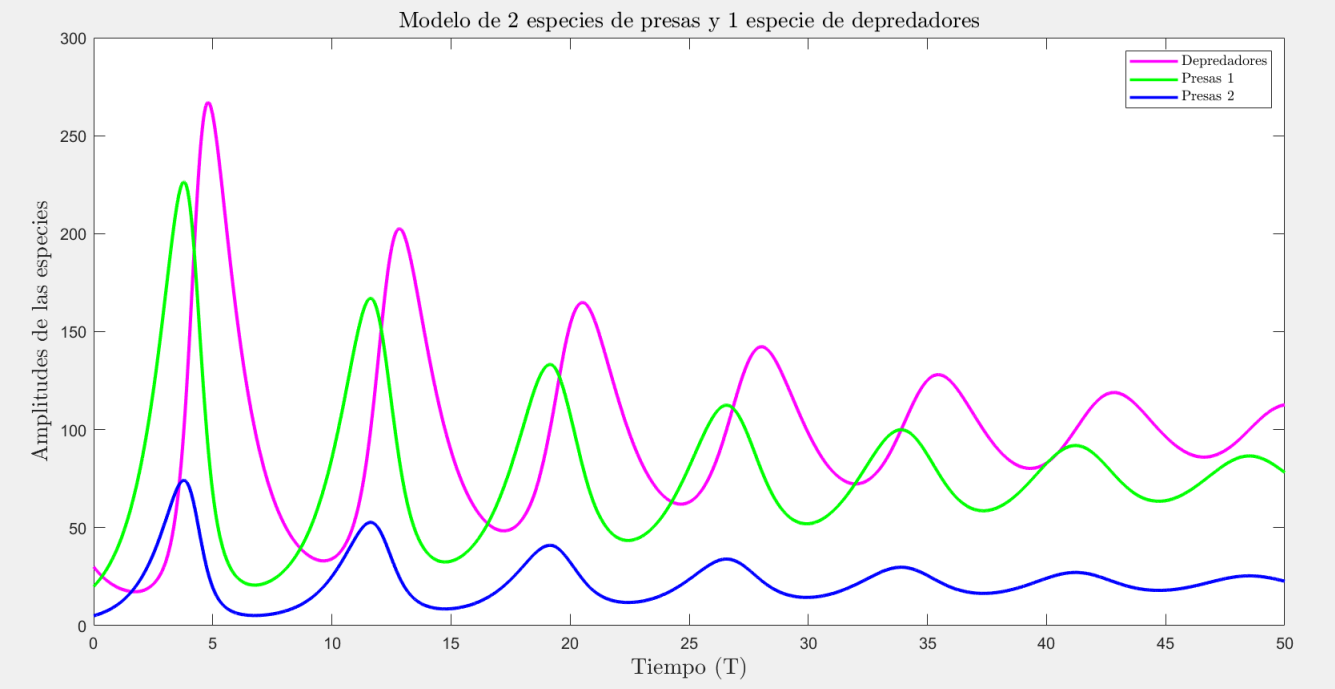


Figura 13: Comportamiento de la dinámica de poblaciones del modelo de 2 especies de presas y una especie de depredadores modificado. Anexo 10 Sistema de dos especies de presas y una especie de depredadores modificado.

La figura 13 muestra como se comporta la dinámica de poblaciones del modelo de dos especies de presas y una especie de depredadores cuando sufre una modificación en la ecuación del depredador, añadiendo el efecto de competencia intraespecífica. Al igual que en los modelos anteriores, la modificación de éste ocasiona que la dinámica temporal alcance una reducción en las amplitudes de cada especie, logrando que las cantidades de presas puedan sostener a las cantidades de los depredadores sin alterar su comportamiento. En este caso la selección de parámetros fue la siguiente: $e_1 = 0.008$, $e_2 = 0.009$, $d = 0.7$, $k_1 = 0.9$, $c_1 = 0.009$, $k_2 = 1$, $c_2 = 0.01$, $T = 0.001$ (T es el parámetro de competencia entre

depredadores), las condiciones iniciales fueron $x(0) = 30$, $y(0) = 20$ y $z(0) = 5$, el vector de tiempo abarca desde $t_0 = 0$ hasta $t_f = 50$ unidades de tiempo. (Ver anexo 10 Sistema de dos especies de presas y una especie de depredadores modificado). A continuación se mostrará la proyección del atractor del modelo modificado:

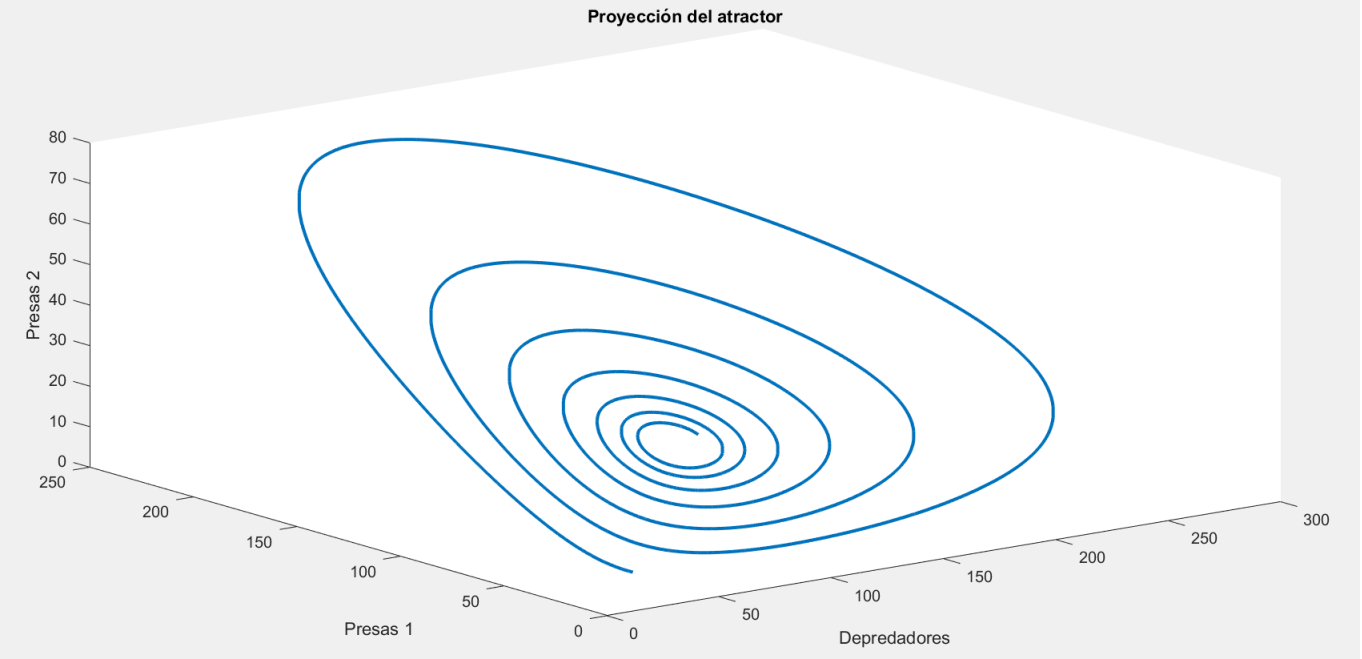


Figura 14: Comportamiento del atractor del modelo de 2 especies de presas y una especie de depredadores modificado. Anexo 10 Sistema de dos especies de presas y una especie de depredadores modificado.

La figura 14 muestra el comportamiento del atractor en relación a la modificación del modelo matemático antes explicado, justo como se vió con los análisis de los modelos anteriores, todas las dinámicas tienden a una reducción en las especies, pareciendo que el sistema converge a un punto en el espacio debido al efecto de competencia entre depredadores, por la razón que se había explicado anteriormente, como la población depredadores es más ineficiente no permiten que la población de presas pueda recuperarse, lo que provoca que las interacciones sean menores hasta llegar a un punto donde éstas puedan sostener a la población de depredadores.

A pesar de que esta forma es la que se usará para realizar la implementación en los grupos de robots móviles las dinámicas no tienden a estabilizarse en un punto fijo ya que se usan agentes reales con desplazamiento, los efectos de competencia sirven como herramienta de regulación de la población de depredadores cuando ésta tiene un número grande de integrantes, lo cual ocasiona que su población disminuya y permita la reproducción de las poblaciones de presas, permitiendo un equilibrio en forma de ciclos no regulares, debido a que las condiciones no son ideales y continuas cuando se usan agentes móviles (Von Arb, 2013).

La conclusión que se puede obtener de este capítulo es que al analizar el sistema de *Lotka-Volterra* se sabe que éste es la base de los demás modelos propuestos en este trabajo de investigación, ya que los sistemas de *dos depredadores y una presa* y *dos presas y un depredador* corresponden a modificaciones del sistema de *Lotka-Volterra*. Por ejemplo la ecuación (1), correspondiente a la población de presas de *Lotka-Volterra* es muy parecida a la ecuación (3) correspondiente a la población de presas del sistema *dos depredadores y una presa*, y a las ecuaciones (7) y (8) correspondientes a las poblaciones de presas del sistema *dos presas y un depredador*, ya que mantienen la misma estructura, es decir, tienen una componente malthusiana (término positivo de crecimiento exponencial de las presas) y los términos de regulación (términos negativos donde tienen interacción con los depredadores para regular la población de presas). La única diferencia es que en la ecuación (3) hay dos términos de regulación de presas, sencillamente porque están bajo la influencia de dos poblaciones de depredadores. Por otro lado con el caso de los depredadores es equivalente al caso de las presas. Por ejemplo, la ecuación (2), ecuación de la población de los depredadores de *Lotka-Volterra*, es muy similar a las ecuaciones (4) y (5) del sistema *dos depredadores y una presa* y a la ecuación (6) del sistema *dos presas y un depredador*, debido a que tienen la componente malthusiana (termino negativo que

aumenta los decesos de los depredadores por ausencia de presas) y el termino de regulación (término positivo que aumenta la población de depredadores por la presencia de las presas), la única diferencia radica en la ecuación (6) con dos términos de regulación, por la razón de que en ese caso el depredador esta bajo la influencia de dos poblaciones de presas. El hecho de que en cada sistema se hayan obtenido dinámicas temporales oscilatorias se sustenta en el comentario de (Von Arb, 2013) quien explica que esos comportamientos son comunes para todas las relaciones presa-depredador debido a que todas las dinámicas vuelven al punto donde empezaron.

Lo anterior es basado en los modelos matemáticos originales, sin embargo, los modelos matemáticos modificados sirven para conocer que sucede en las dinámicas de poblaciones cuando la influencia de competencia participa como efecto de regulación en las especies depredadoras, esta influencia será efectuada en los algoritmos de control que serán implementados en los grupos de kilobots, con la intención de mantener un equilibrio entre las especies y asemejar sus respuestas a los modelos matemáticos originales (comportamientos periódicos).

Otra situación que se debe explicar es que en los modelos matemáticos modificados a medida que los valores de competencia entre depredadores crezcan aumenta la ineficiencia de éstas, ocasionando que las especies interactúen de forma más negativa. Esto se debe a que entre más crezca la competencia entre depredadores más se limita su crecimiento de población, pero al haber presas suficientes aún funcionan como suministro para la población depredadora, evitando que desaparezcan.

Capítulo 4

4. Robot móvil kilobot

Un kilobot es considerado como un robot de código abierto con operaciones totalmente escalables. Este robot fue diseñado para hacer que las pruebas de algoritmos colectivos de cientos o miles de robots sean accesibles para investigadores del tema de *robótica* (Rubenstein et al., 2013).

El diseño del hardware de un robot kilobot permite que se puedan producir en grandes cantidades de óptima manera, ademas de que un robot de este tipo posee habilidades similares a las de otros robots colectivos, algunas de ellas son las siguientes:

- Locomoción de accionamiento diferencial.
- Poder de computo abordo.
- Comunicación de vecino a vecino.
- Detección de luz ambiental.
- Detección de distancia de vecino a vecino.

Por otra parte en (Carrasco Gutiérrez et al., 2020) se describe al kilobot como: "un robot de bajo costo, fácil de usar para el desarrollo avanzado de enjambres de robots que pueden ser programados para realizar funciones útiles, coordinando interacciones entre sus muchos individuos". Estos enjambres pueden inspirar su comportamiento en los insectos sociales, tales como las colonias de hormigas, que pueden eficientemente buscar y encontrar fuentes de comida en ambientes complejos y extensos, para de forma colectiva, transportar objetos grandes y coordinar la construcción de nidos y puentes en tales ambientes.

En (Carrasco Gutiérrez et al., 2020) se mencionan algunas ventajas del kilobot descritas de K-TEAM (2011), las cuales son:

- Son de bajo costo.
- Son pequeños (33 mm de diámetro) y 34 mm de altura (incluyendo las patas del robot).
- Pueden comunicar cadenas de 3 bytes con sus vecinos hasta 7 cm de distancia. (Pueden medir la distancia de vecino a vecino).
- Cuentan con un sensor de luz ambiental.
- Tiene un LED RGB con tres niveles de intensidad por color.
- Cuenta con una batería recargable y removible.
- Es fácil de manipular con el controlador, con el cual se puede interactuar con cientos de kilobots a la vez.

4.1. Estructura del robot

Se deben considerar dos factores competitivos al diseñar el robot kilobot: *costo y funcionalidad*. El robot necesita suficiente funcionalidad para permitirle realizar una amplia variedad de comportamientos colectivos, y al mismo tiempo debe ser lo suficientemente simple con el fin de mantener el costo bajo.

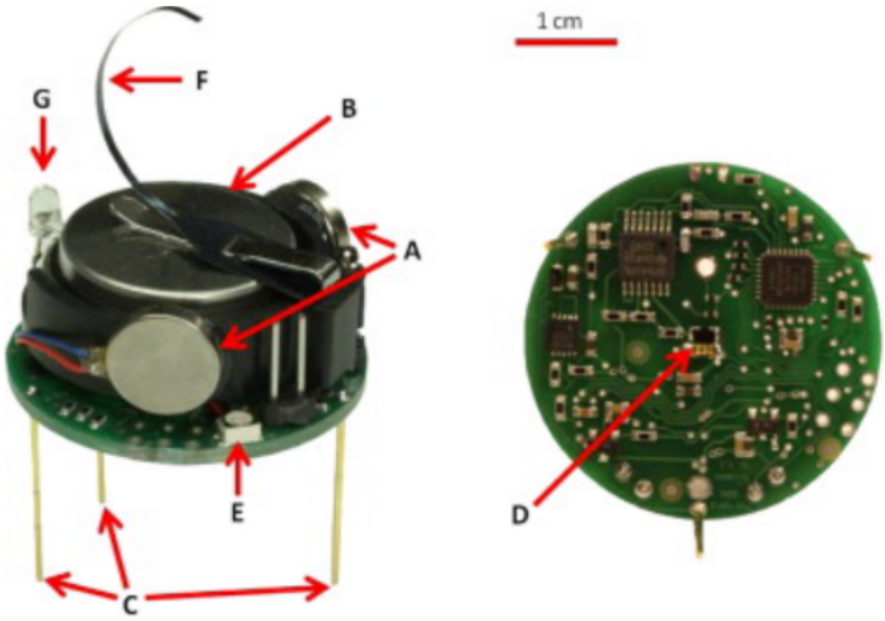


Figura 15: Vistas isométrica (izquierda) e inferior (derecha) de un kilobot, imagen obtenida del trabajo de (Rubenstein et al., 2013).

La figura 15 muestra la composición de los elementos del kilobot, los cuales lo hacen un robot funcional para le generación de comportamientos con robots del mismo tipo. Cada una de las indicaciones de la figura anterior se mencionarán a continuación:

- A: Motores de vibración.
- B: Batería del robot.
- C: Soportes de apoyo rígidos.
- D: Transmisor/Receptor de infrarrojos.
- E: Led de tres colores (RGB).
- F: Pestaña de carga.
- G: Sensor de luz ambiental.

4.2. Locomoción del robot

La estrategia de locomoción más común para los robots de enjambre es utilizar un accionamiento diferencial de dos ruedas, donde cada rueda es impulsada por un motorreductor eléctrico. Sin embargo, este tipo de locomoción suele ser muy costoso. Para mantener el bajo costo accesible del kilobot se utiliza dos motores de vibración para la locomoción. Cuando uno de estos motores es activado, las fuerzas centripetas generadas por el motor vibratorio se convierten en una fuerza de avance para el kilobot. Debido al montaje descentrado de los motores la vibración de un motor provocará una rotación del kilobot sobre su eje vertical, mientras que la vibración del otro motor provocará una rotación opuesta.

Al controlar la magnitud de la vibración de los motores de forma independiente de manera diferencial, el robot puede moverse en un rango continuo desde la rotación en el sentido de las agujas del reloj hasta el sentido contrario. Esto permite que el kilobot se mueva aproximadamente a 1 cm/s y gire $45^o/s$.

4.3. Comunicación y detección de kilobots

La gran mayoría de los algoritmos de robots colectivos utilizan la comunicación y detección de robot a robot, como la distancia y el rumbo a los vecinos, esa es la información principal para impulsar la colectividad en robots individuales.

Para la comunicación con los robots vecinos, cada kilobot tiene un transmisor de LED infrarrojo y un receptor de foto-diodo infrarrojo, ambos ubicados en la parte inferior del robot, que se encuentran apuntando a la superficie sobre la que se encuentran. Los kilobot tienen la cualidad de que pueden recibir mensajes de información de sus vecinos desde cualquier dirección, sin embargo, la distancia de separación entre ellos debe ser al rededor de 7 cm para la comunicación pueda ser

cumplida (Rubenstein et al., 2013).

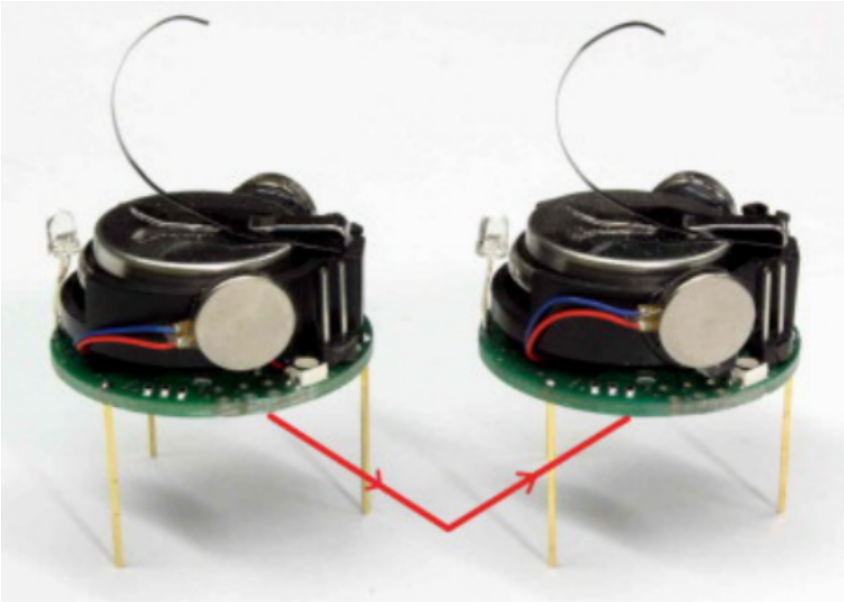


Figura 16: Ilusración de la comunicación entre kilobots, imagen obtenida del trabajo de (Rubenstein et al., 2013).

La figura 16 muestra la forma en la que se realiza la comunicación entre kilobots, como se observa, el kilobot de la izquierda envía una señal de luz a través de un transmisor infrarrojo y el kilobot de la derecha se encarga de recibir dicha señal mediante el foto-diodo infrarrojo, el requisito para que la comunicación se cumpla es que la distancia sea cercana (menor o igual a 7 cm) entre ellos.

El propósito de este trabajo de tesis consiste precisamente en usar la comunicación entre los kilobots implementando modelos matemáticos basados en algunas relaciones *presa-depredador*, para reflejar sus dinámicas en ellos y verificar si la respuesta por parte de los grupos de robots tienen semejanza con las respuestas de los modelos matemáticos.

Capítulo 5

5. Programa de aplicación y algoritmos

Es importante tener a disposición una forma de poder proyectar los comportamientos que surgen debido a la relación presa-depredador mediante agentes móviles que reflejan la dinámica de los individuos naturales sociales, como se ha mencionado anteriormente, los agentes móviles que serán utilizados en este trabajo de investigación son los *kilobots*. Una de las cualidades fundamentales sobre estos robots móviles es la función de poder comunicarse en enjambre a corta distancia, es decir, la cualidad de interacción de vecino a vecino, donde la operación de movimiento debe ser necesariamente en una superficie lisa. La forma en la que se va a proyectar la dinámica de los comportamientos colectivos mediante los kilobots será con la aplicación de un programa llamado *CoppeliaSim*.

Antes de seleccionar esta forma de proyectar las dinámicas de los kilobots se contempló otra posible alternativa que también serviría para el mismo objetivo, se trata de la forma experimental. Una de las razones principales por la que se decidió tomar el camino de aplicar el programa *CoppeliaSim* era que proporciona ventajas de poder mostrar evidencias de manera gráfica, mientras que de la forma experimental no se cuenta con esa ventaja, otra ventaja es que la ejecución dentro del programa CoppeliaSim es en tiempo real, eso quiere decir que las operaciones de los kilobots son muy similares, aproximándose a lo que ocurre en la realidad (forma experimental), otra ventaja adicional del programa CoppeliaSim es que cuenta con un sistema de verificación del prototipo, esto quiere decir que al seleccionar al individuo robótico que se utilizará el programa despliega algunos elementos principales de como este individuo se compone. Por lo que la aplicación del presente programa es útil para la familiarización con sistemas robóticos.

5.1. CoppeliaSim

El entorno de *CoppeliaSim* utiliza una plataforma de experimentación con robots virtuales, el cual representa el resultado de los esfuerzos de tratar de cumplir con todos los requisitos en un marco de simulación versátil y escalable (Rohmer et al., 2013).

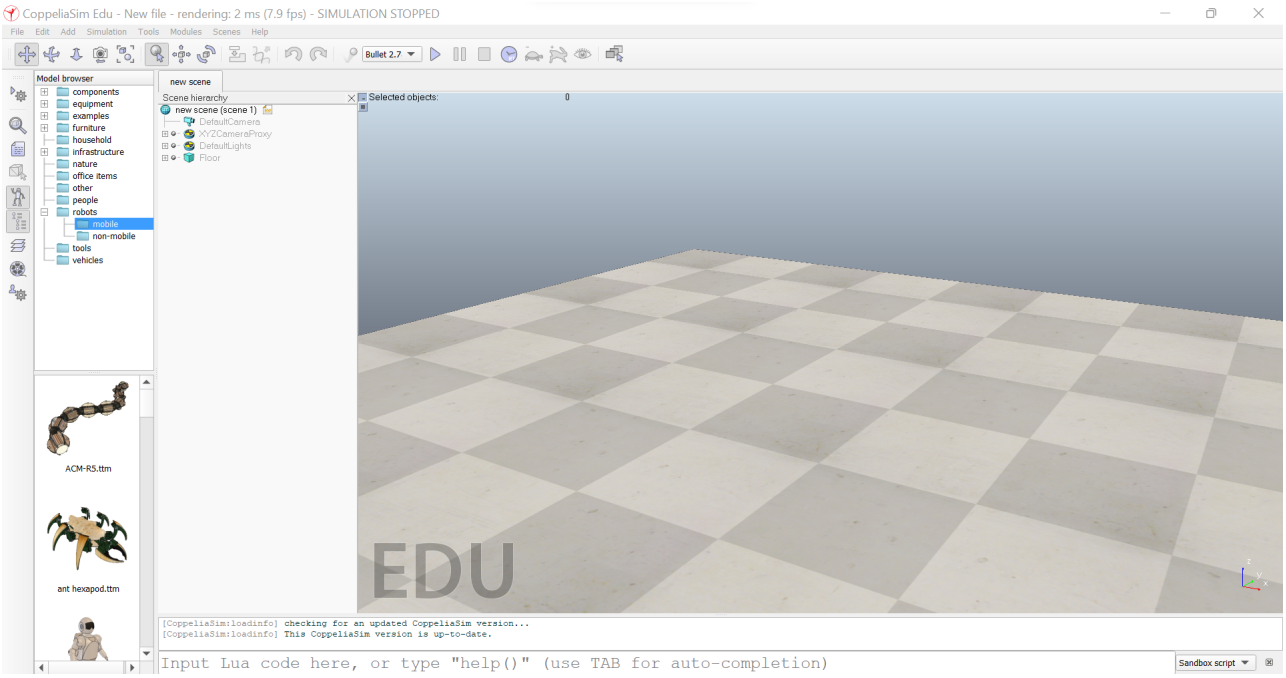


Figura 17: Ejemplo de la interfaz principal del programa *CoppeliaSim*.

La figura 17 representa el lugar de trabajo donde diferentes individuos robóticos (tanto móviles como no móviles) pueden operar de una forma similar a comparación de la realidad, además de brindar información sobresaliente del individuo robótico seleccionado por parte de algún usuario.

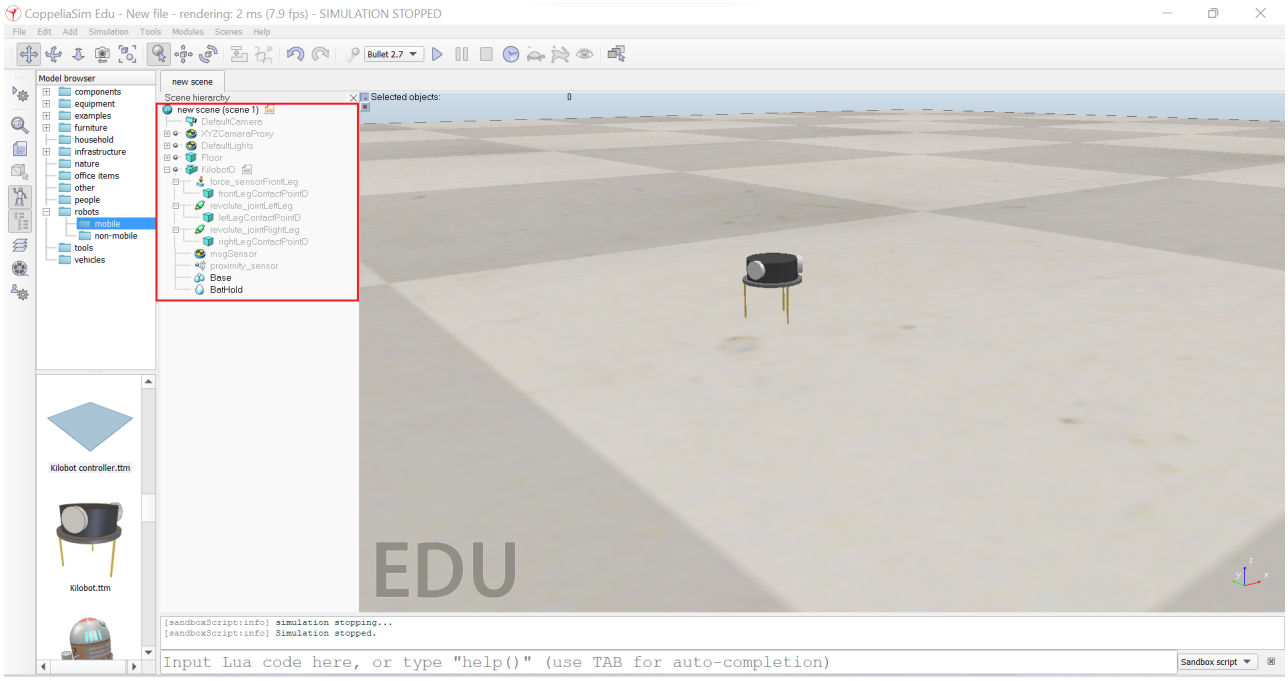


Figura 18: Ejemplo de los elementos que componen al kilobot en el programa *CoppeliaSim*.

La figura 18 muestra la interfaz del programa CoppeliaSim, al igual que la figura 17, con la diferencia de que en el escenario de la interfaz se encuentra un ejemplo del robot kilobot. En la parte izquierda del escenario se observa un recuadro en color rojo, dentro de ese cuadro se muestran algunos elementos utilizados para la configuración del kilobot dentro del entorno de CoppeliaSim, como lo son el sensor de fuerza de la pierna frontal, las articulaciones de las piernas izquierda y derecha, el sensor de mensajes (útil para la comunicación con vecinos cercanos a un rango de distancia de 7 cm), el sensor de proximidad (útil para la detección de objetos en el entorno del kilobot), la batería del robot, su base, entre otros posibles elementos.

Es importante resaltar estos elementos ya que se puede dar una idea del funcionamiento del kilobot, sin embargo falta explicar otros elementos importantes, como lo son el lugar para agregar el código del robot (necesario para indicarle las ordenes a seguir) y el controlador del kilobot (necesario para inicializar el movimiento del robot), a continuación, se

mostrarán estos elementos:

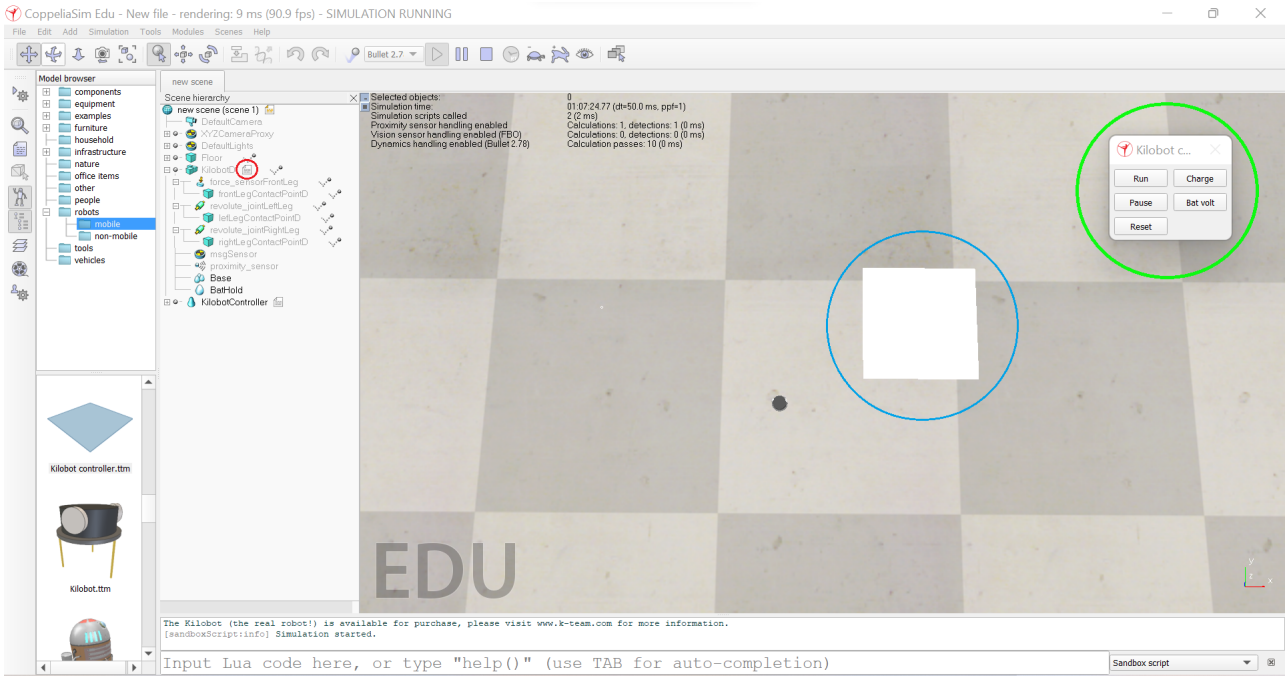


Figura 19: Apartado del código y controlador del kilobot en el programa *CoppeliaSim*.

Como se observa en la figura 19, en el lado izquierdo en un pequeño círculo en rojo se encuentra el scrip para poder añadir el código necesario del kilobot para darle la dinámica correspondiente a su movimiento, en cambio, en la interfaz del programa se encuentra un círculo más grande en color azul, haciendo referencia al controlador del kilobot, el cual se encarga de inicializar el movimiento del kilobot, ligado al controlador se encuentran las opciones de él, mostradas en el círculo verde, las cuales son: Run (correr), Pause (pausa), Reset (reinicio), Charge (carga) y Bat volt (voltaje de la bateria).

Una vez conociendo al individuo robótico de trabajo, el kilobot en este caso, sus características y opciones relacionadas a las simulaciones de movimiento, ya es posible conocer las instrucciones que se le deben asignar al kilobot para seguir la relación presa-depredador mediante un enjambre (grupo compuesto por varios robots). Dichas instrucciones son en base a condicionales que le permiten al kilobot los pasos a seguir

cuando tienen a otros kilobots cercanos, por poner un ejemplo, cuando hay una presa cercana a otra presa, cuando una presa esta cerca a un depredador o cuando un depredador esta cerca a un depredador. Lo anterior son las diferentes combinaciones que se dan en la realidad y es posible reflejar esos escenarios utilizando grupos de kilobots, mediante la aplicación del software CoppeliaSim.

5.2. Algoritmo presa-depredador

Es importante mencionar que hay ciertas diferencias en cuanto a la forma de representar el algoritmo presa-depredador en los programas de MATLAB y CoppeliaSim, anteriormente se ha explicado como obtener la dinámica del modelo presa-depredador referente al sistema de Lotka-Volterra a través de MATLAB, sin embargo, para poder representar el algoritmo presa-depredador a través de grupos de robots móviles se utilizará CoppeliaSim para llevar a cabo esa actividad. Aquí hay una cuestión importante a considerar y es que la implementación del algoritmo es de forma diferente pero se basa en el mismo principio. Para poder representar el algoritmo presa-depredador es necesario primeramente definir las características de cada papel (presa y depredador) y cuales serán las interacciones que podrían tener con otros individuos en el entorno (interacción de vecino a vecino). Dichas características e interacciones se encuentran en el trabajo realizado por (Oganician, 2017), las cuales se representan por (+) cuando se tratan de interacciones positivas y por (-) cuando se tratan de interacciones negativas. Primeramente se representa la interacción de la depredación, la cual se define en ese trabajo como una relación en la que los individuos de una especie (depredadora) cazan a individuos de la otra especie (presa) para subsistir, y siempre tiene un efecto negativo sobre el individuo, por lo que es posible categorizar a la interacción anterior de la forma (+ -). Es importante categorizar otra interacción ademas de la depredación, la cual es la interacción de la competencia, en el trabajo realizado por (Oganician, 2017) se define

que la competencia ocurre cuando la relacion de ambas especies va en detrimento, debido a esa razón se puede representar como (- -), ya que ninguna de las especies se vería beneficiada.

Una vez explicado lo anterior referente a lo que ocurre cuando dos especies (una de presas y otra de depredadores) interactúan entre sí es posible relacionarlo a lo que ocurre en un escenario real. En este caso cuando en un ambiente se encuentran un grupo de presas y un grupo de depredadores e interactúen entre ellos lo principal que ocurrirá sería el efecto de la depredación, y cuando los depredadores interactúen entre ellos por el recurso de las presas se presentará el efecto de la competencia intraespecífica.

Las interacciones explicadas anteriormente son posibles representarlas mediante una esquematización popular denominada máquina de estados finitos. Se optará este camino porque esta es la metodología que los kilobots pueden seguir, para cumplir las interacciones que tendrán con otros kilobots vecinos y cumplir las características que tendrán cuando adquieran tanto el estado de presa como el estado de depredador. A continuación, se mostrará la esquematización de la máquina de estados finitos para la implementación del algoritmo presa-depredador:

Figura 20: Algoritmo presa-depredador representado por una maquina de estados finitos.

La figura 20 es una representación que explica de que forma se van a comportar los agentes móviles cuando adquieran sus estados respectivos y por consecuencia las características que éstos conllevan. El rol de presa o depredador se le asigna al kilobot de forma en la que el usuario desee, sin embargo las características de cada rol se implementan por separado para cada comportamiento. En la siguiente sección se explicará cada algoritmo por separado para saber de que forma se implementan las características respectivas de cada comportamiento.

5.3. Algoritmo de la presa

Para poder explicar de manera detallada el algoritmo de la presa se representarán las características que éste conlleva a través de un diagrama de flujo, en el cual se incluirán las mismas características e interacciones que se encuentran en la figura 20, las cuales explican la metodología para poder implementar el algoritmo en el programa de CoppeliaSim. A continuación, se mostrará el algoritmo de la presa:

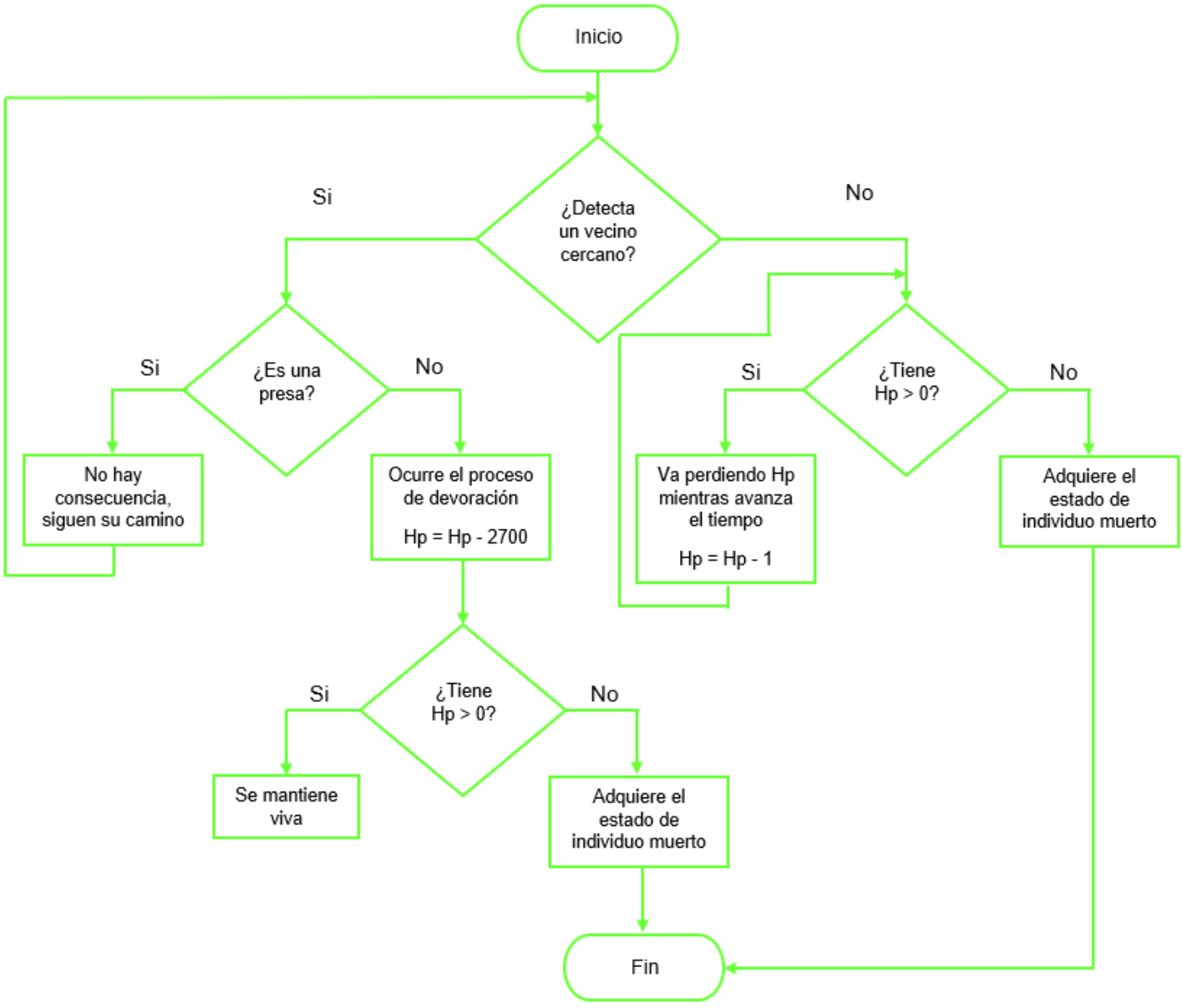


Figura 21: Algoritmo del comportamiento de la presa.

En la figura 21 se explican las interacciones que tendrá un kilobot cuando adquiera el estado o el rol de la presa, al dar inicio lo que hará el kilobot será empezar a moverse por el área, en caso de detectar a un vecino cercano se hará la pregunta ¿Es una presa? En caso de serlo el kilobot seguirá desplazándose por el área sin riesgo alguno o consecuencia. Por otro lado en caso de no ser una presa por consecuencia será un depredador y lo que sucedería ahí sería el proceso de devoración por parte del depredador a la presa. En el diagrama de flujo se observa una ecuación denotada como $Hp = Hp - 2700$, donde Hp hace referencia a los *Health Points* o *Puntos de Vida* de la presa. La ecuación anterior hace mención a que los Hp de la presa disminuirán siempre y cuando esté dentro del rango del depredador (dicho rango puede ser menor o igual a 7 cm), el valor de 2700 puede ser modificable, sin embargo, al hacer las pruebas pertinentes dicho valor reflejó mejor la dinámica de las presas. Otra cuestión a considerar es que en el momento que $Hp = 0$ la presa adquirirá el estado de individuo muerto al instante.

Por otro lado, cuando no hay interacción por parte de la presa con algún otro vecino de forma automática comienza el descenso de Hp de la presa, denotada por la ecuación $Hp = Hp - 1$, dicha ecuación hace mención a que desde el inicio del programa mientras avanza el tiempo los Hp de la presa irán disminuyendo de uno por uno, eso es con el fin de reflejar lo que sucede en un escenario real, ya que todos los seres vivos presentan disminución en su calidad de vida mientras el tiempo avanza. Al igual que la condición anterior, en el momento en el que sus Hp sean equivalentes a 0 la presa adquirirá el estado de individuo muerto al instante, concluyendo de esa forma su algoritmo.

5.4. Algoritmo del depredador

De manera similar a la explicación del algoritmo anterior se realizará ahora la explicación del algoritmo del depredador, para ello se anexarán las características que este estado conlleva, así como sus interacciones al momento de encontrarse con un kilobot vecino, equivalentes a la figura 20, representados en forma de un diagrama de flujo, con la intención de visualizar de una mejor manera dichas interacciones con sus vecinos. A continuación, se mostrará el algoritmo del depredador:

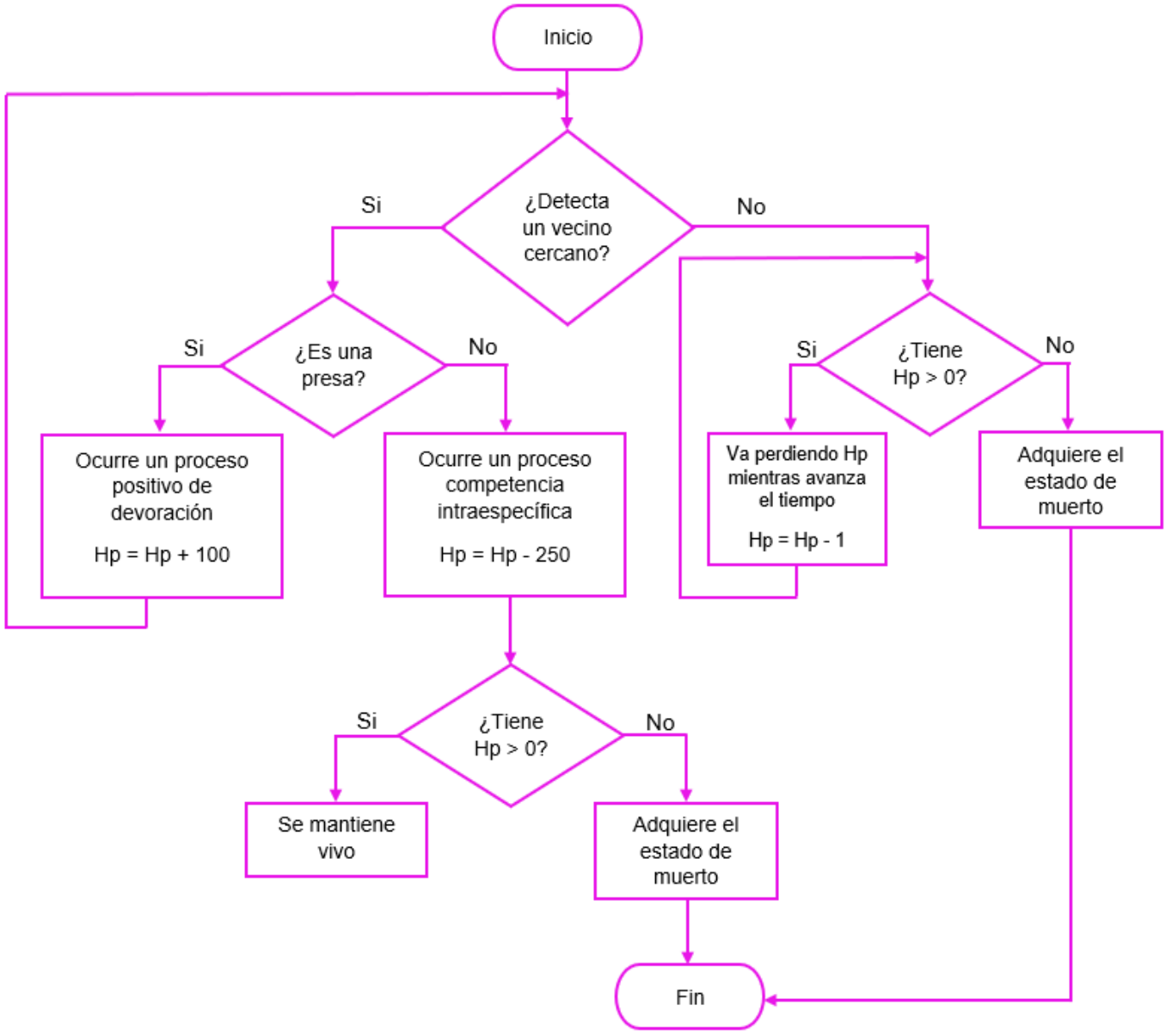


Figura 22: Algoritmo del comportamiento del depredador.

La figura 22 muestra el algoritmo del depredador, el cual es bastante similar al de la presa, sin embargo, tiene algunas modificaciones que a continuación se explicarán. Al iniciar el programa el kilobot que tenga el papel del depredador comenzará a moverse por el área, al momento de encontrarse con un kilobot vecino se hará la pregunta ¿Es una presa? En caso de que si lo sea ocurrirá el proceso de devoración por parte del depredador hacia la presa, dicho proceso es positivo para el depredador, debido a que las presas son el sustento de ellos, al ser positivo ese proceso se refleja como un aumento en sus Hp, esa es la razón de la ecuación $Hp = Hp + 100$ que se encuentra en el diagrama de flujo. También se debe considerar la alternativa cuando la interacción del depredador no es con una presa, al no ser con una presa por consecuencia será con un depredador, y al haber la interacción de un depredador con un depredador ocurre un proceso denominado *competencia intraespecífica*, dicho proceso afecta de forma negativa al depredador influyendo en sus Hp, esa es la razón de la ecuación matemática $Hp = Hp - 250$, es decir, que cada vez que un depredador interactúe con otro depredador sus puntos de vida irán en disminución. En el momento en el que los puntos de vida del depredador vayan en disminución si llegan al valor de 0 por consecuencia el depredador adquiriría el estado de individuo muerto.

En este momento se ha explicado que ocurre con el depredador cuando ha tenido interacción con otros vecinos, tanto presa como depredador, por el contrario, cuando no hay ninguna interacción por parte del depredador con algún vecino lo que ocurrirá es el descenso de sus puntos de vida de forma inmediata, de uno por uno, es decir, $Hp = Hp - 1$, en el momento que no se cumpla la condición $Hp > 0$ el depredador adquirirá el estado de individuo muerto al instante, concluyendo de esa forma su algoritmo.

5.5. Algoritmo del individuo muerto

Una vez explicados los algoritmos tanto de la presa como del depredador faltaría por representar el algoritmo del individuo muerto, es decir, lo que sucede con una presa cuando es devorada o en el caso de los depredadores cuando resultan perjudicados por el efecto de la competencia intraespecífica o por la escasez de las presas. Así como en los casos anteriores se anexará un diagrama de flujo correspondiente a este algoritmo para poder visualizarlo de una manera más sencilla. A continuación se mostrará dicho algoritmo:

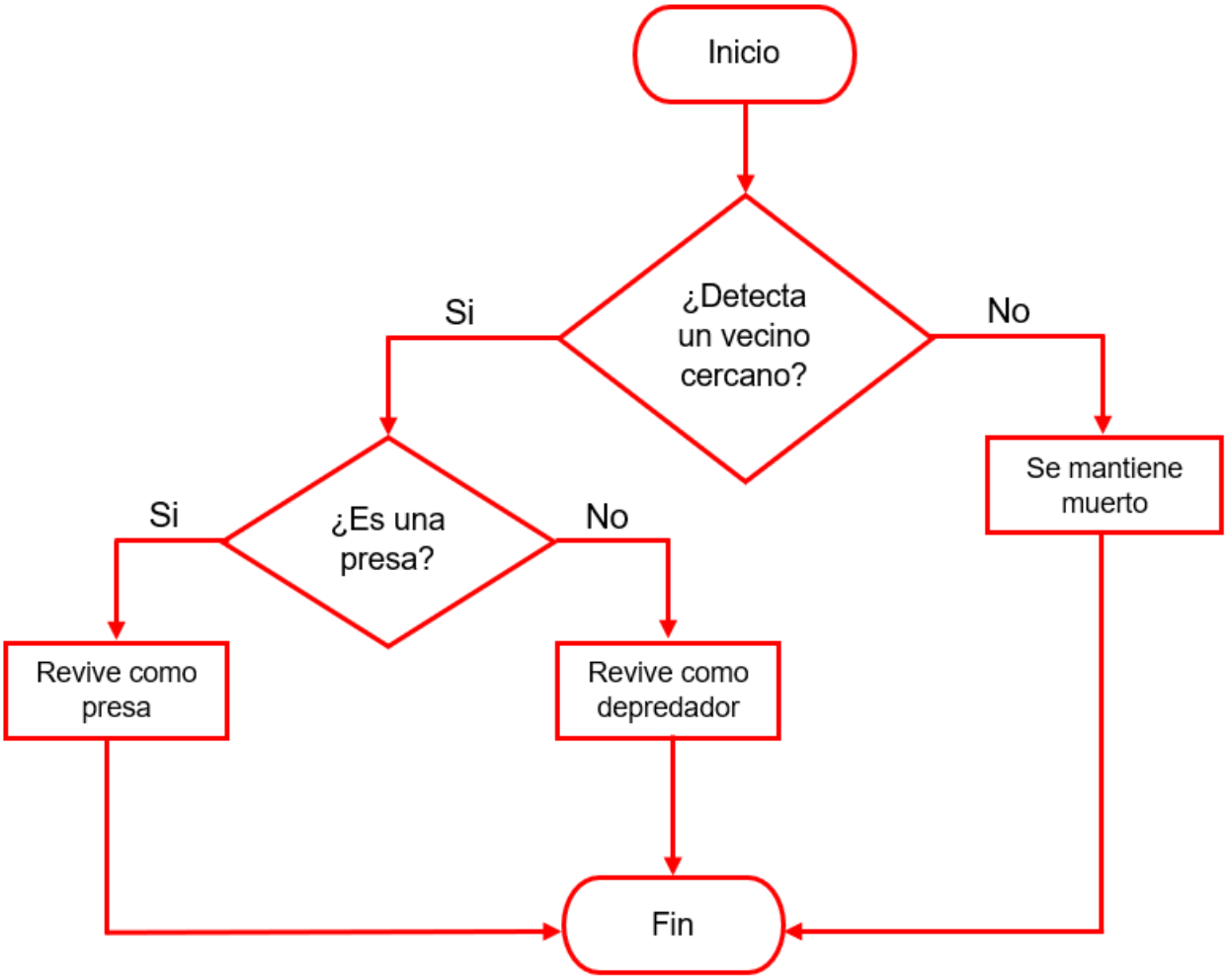


Figura 23: Algoritmo del comportamiento de un individuo muerto.

La explicación del algoritmo de la figura 23 consiste en que cuando un kilobot adquiere el estado de individuo muerto (completamente inmóvil) éste todavía puede interactuar con las presas y depredadores que se encuentren por el área, de modo que cuando una presa se encuentre a una cierta distancia de un kilobot que tenga este estado puede volver a operar como presa, esa condición aplica del mismo modo para los depredadores, es decir, que cuando un depredador se encuentre lo suficientemente cerca de un kilobot con estado de individuo muerto éste puede volver a operar como un depredador.

La condición anterior de que los kilobots puedan volver a operar una vez que hayan hecho la transición del estado de individuo muerto al estado de presa o depredador según sea el caso hace una alusión a la reproducción de las especies, es decir, con que frecuencia se producen los nuevos individuos a causa de los anteriores.

Otro dato importante por el cual se optó por realizar que los kilobots revivan como presa o depredador según sea el caso es que es posible volver a reusar a los kilobots que se tenían en un inicio, de tal manera que reflejen la reproducción de las especies sin la necesidad de agregar a nuevos agentes móviles.

Capítulo 6

6. Resultados

Después de haber realizado la explicación de los algoritmos correspondientes a la forma en como se desenvuelven las presas y los depredadores en un entorno cerrado, y las interacciones que éstos tienen con sus vecinos, se procederá a realizar la implementación de cada algoritmo en diferentes enjambres de robots móviles para observar que comportamientos generan, y validar si lo que generan es equivalente o similar a la respuesta que generan los modelos matemáticos estudiados en la literatura.

6.1. Escenario de 1 población de presas y 1 población de depredadores

El primer escenario a modelar corresponde al escenario compuesto por únicamente dos poblaciones, una de presas y otra de depredadores, el modelo matemático relacionado a este escenario es el modelo básico de Lotka-Volterra, ya que las condiciones de este modelo son únicamente para dos especies. Con el propósito de poder representar este modelo matemático en grupos de pequeños robots móviles se seguirá la metodología representada por la figura 20 y los algoritmos representados por las figuras 21, 22 y 23.

Lo primero que se debe hacer para poder realizar la implementación de los algoritmos correspondientes en los robots móviles de este escenario es recurrir al programa CoppeliaSim y acondicionar un área de trabajo para la operación de los kilobots. A continuación, se mostrará el área de trabajo creada para representar el escenario de 1 población de presas y 1 población de depredadores (la cual será utilizada también en los siguientes escenarios):

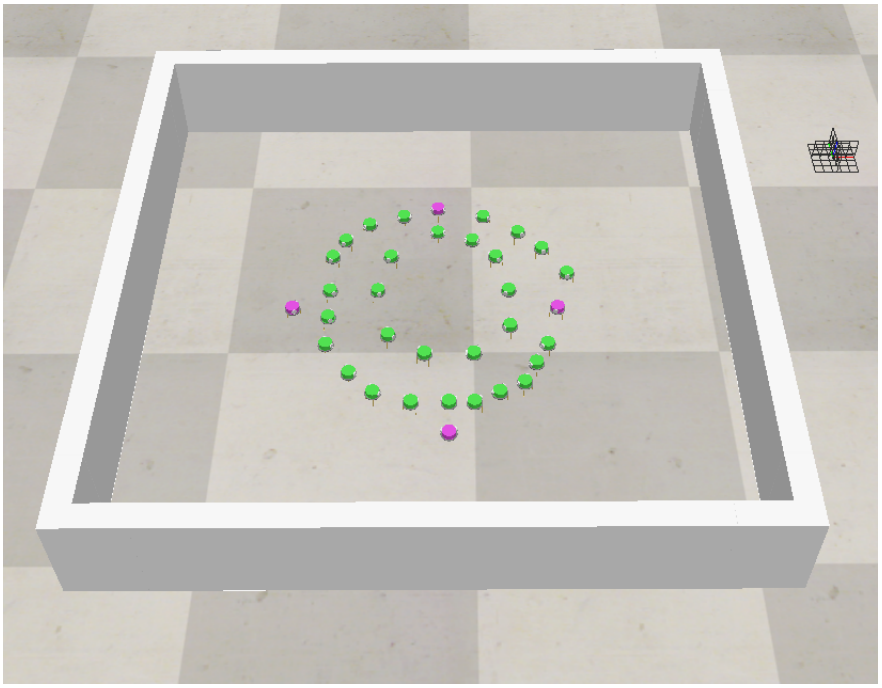
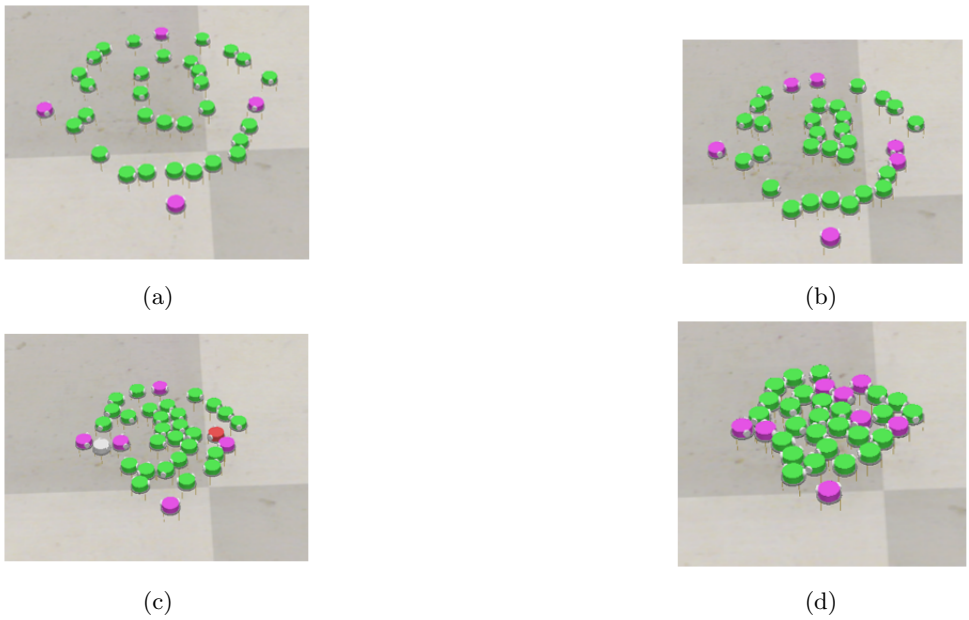


Figura 24: Herramientas de trabajo del escenario 1 población de presas y 1 población de depredadores.

La figura 24 muestra el material utilizado que el propio entorno de programación proporciona al usuario para el acondicionamiento del escenario a tratar. En este caso el material utilizado fue el siguiente:

- 4 muros de 100 cm de largo por 20 cm de altura.
- 4 muros en forma de L de 20 cm de altura, los cuales irán en las esquinas del muros de 100 cm de largo para formar un área cuadrada.
- 34 kilobots, los cuales serán 30 presas y 4 depredadores, es decir, las condiciones iniciales $x(0) = 30$ y $y(0) = 4$.
- 1 graficador, para obtener de manera visual las interacciones de los robots móviles.

Las herramientas mencionadas previamente fueron de gran ayuda para poder realizar la implementación del algortimo presa-depredador en el grupo de robots que se observa en la figura 24, el cuadro exterior resulta ser de 120 cm por 120 cm, y en su interior se encuentra el grupo de pequeños robots móviles, donde los kilobots mostrados en color verde representan a las presas y los kilobots en rosa a los depredadores. Posteriormente se observará la respuesta a la dinámica que realizan los robots móviles al algoritmo implementado en cada uno de ellos, para ello se anexará la siguiente figura:



(a) (b) (c) (d)

Figura 25: Interacciones de respuesta del algoritmo presa-depredador representado por un grupo de pequeños robots móviles. Anexo 10.1 Algoritmo presa-depredador en robots móviles (CoppeliaSim).

La figura 25 representa la implementación del algoritmo presa-depredador en un grupo de 34 kilobots, en los cuales 30 hacen el papel de la presa (con color verde) y 4 hacen el papel del depredador (con color rosa), es decir, $x(0) = 30$ y $y(0) = 4$. La implementación del algoritmo fue realizada usando como guía el diagrama correspondiente a la figura 20, en el cual se indican las interacciones que tienen los kilobots cuando son presas o depredadores y tienen vecinos cercanos.

La figura 25 corresponde a uno de los tantos resultados que puede generar este algoritmo cuando es representado por grupos de robots móviles, ya que la influencia de las condiciones iniciales, posiciones iniciales, parámetros de puntos de vida (Hp), velocidad de los motores de los robots, y otros datos que se explicarán más adelante afectan en gran medida al movimiento de los robots y a las transiciones de los estados de los robots, efecto por parte de las interacciones con sus vecinos, por lo que se debe ser consciente que el algoritmo resulta muy sensible a las variaciones de los parámetros que están en marcha.

Realizando la explicación de la figura 25 cada una de las subfiguras que la componen (a, b, c, y d) fueron tomadas aproximadamente 50 segundos una después de la otra, es decir, la figura 25a fue tomada en el instante aproximado de $t = 50$ segundos, en la cual se observa que los agentes móviles han comenzado a desplazarse, se observa también que en ese instante no ha habido alguna interacción por parte de los robots, debido a que las condiciones de inicio no han cambiado. En la figura 25b, tomada en el instante aproximado de $t = 100$ segundos, se observa que hay un ligero aumento en la población de los depredadores, habiendo seis en ese instante, es decir, $y(t) = 6$, lo cual se debe porque hay un número grande de presas, al haber una cantidad grande de presas favorece el aumento en la población de los depredadores. Posteriormente se tiene la figura 25c, tomada en el instante aproximado de $t = 150$ segundos, en la cual se observa un caso de competencia entre depredadores, dicho caso produce el kilobot en color blanco, el cual hace referencia a un depredador muerto, cada vez que un depredador tenga de vecino a otro depredador se produce el efecto de competencia intraespecífica, el cual afecta negativamente a esos depredadores, por otro lado también se observa un kilobot en color rojo, el cual hace referencia una presa devorada por un depredador, cada vez que una presa tenga de vecino a un depredador la presa sera devorada, haciendo una transición de estado de presa viva a

presa muerta. Después, en la figura 25d, tomada en el instante aproximado de $t = 200$ segundos, se observa que la población de depredadores comienza a crecer, por la razón que se había mencionado anteriormente, la población de depredadores prospera al haber un número abundante de presas (Oganician, 2017). Posteriormente después de visualizar lo que sucede en los diferentes instantes de tiempo se anexará el comportamiento dinámico de los kilobots en un tiempo más prolongado para obtener un panorama más amplio sobre el comportamiento especifico de esta prueba entre las dos poblaciones.

Antes de mostrar la gráfica de la dinámica de los kilobots es muy importante hacer énfasis en que la cantidad de respuestas que generan los kilobots es muy variada, eso se debe a que su movimiento de trayectoria es de forma aleatoria, eso se hizo con la intención que se muevan libremente por al área y puedan interactuar con cualquier vecino que se encuentre cercano a ellos. Con el mismo programa, misma configuración de parámetros, condiciones y posiciones iniciales es posible obtener diferentes resultados, los cuales pueden ser modificables para ajustarlos a las respuestas de los modelos matemáticos.

Una vez explicado lo anterior se mostrará la gráfica de comportamientos de los kilobots relacionada con la figura 25, donde las interacciones del experimentos corresponden a únicamente dos especies, una de presas y otra de depredadores:

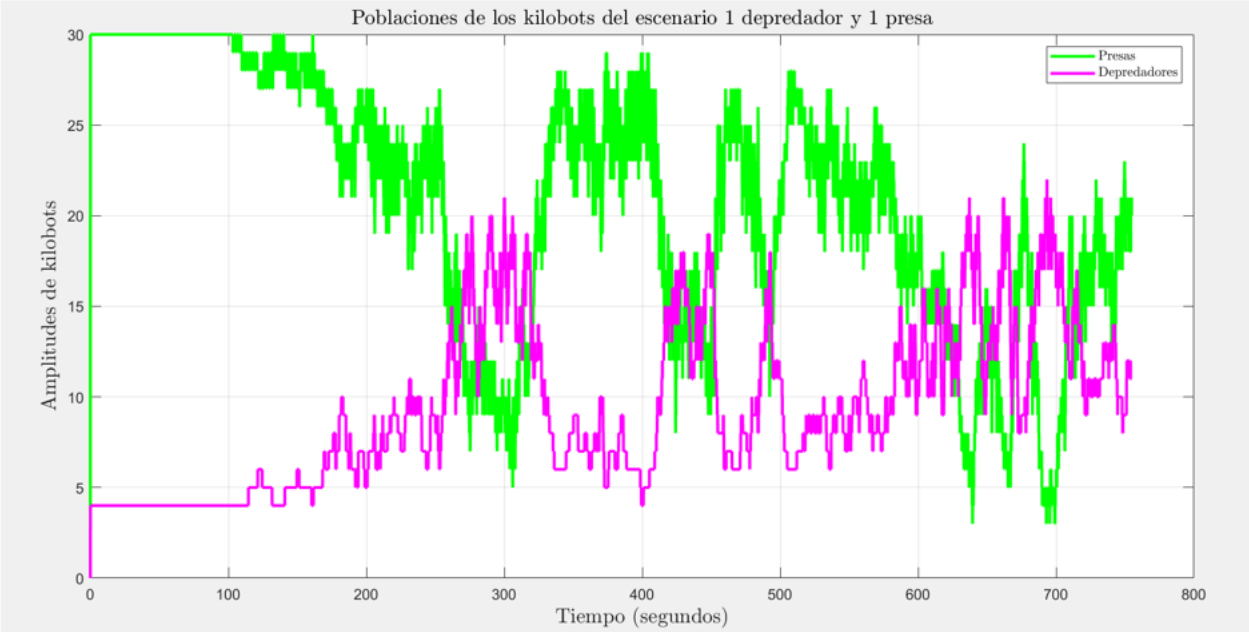
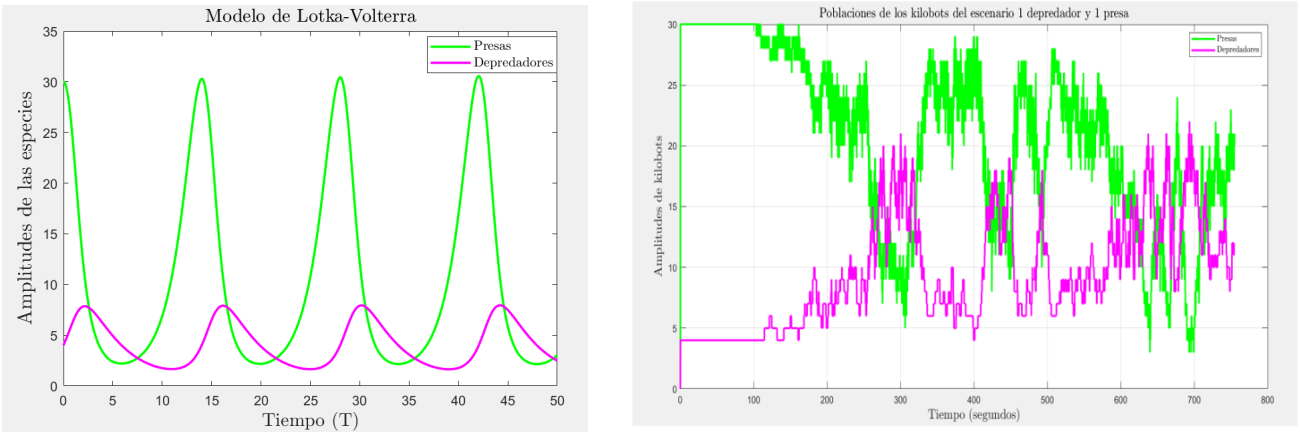


Figura 26: Dinámica de comportamientos de los kilobots en el escenario de 1 población de presas y 1 población de depredadores.

La figura 26 hace referencia a la dinámica de poblaciones de escenario de la figura 25, en la cual se observa algunos comportamientos oscilatorios por parte de las poblaciones de presas y de depredadores, dichos comportamientos comienzan a notarse a partir de la marca de los 100 segundos, en dicho instante de tiempo la población de presas comienza a disminuir y la población de depredadores comienza a crecer a causa de las interacciones de los kilobots, la explicación anterior se sustenta con lo mencionado en los trabajos de (Oganician, 2017) y (Von Arb, 2013), los autores anteriores también comparten la explicación de que cuando la población de depredadores aumenta la de presas disminuye a causa de los efectos de la depredación, lo cual se observa en la gráfica poco antes de la marca de 300 segundos, a partir de ese momento la población de depredadores disminuye, a causa de la disminución de las presas, y la de presas se recupera, a causa del descenso de los depredadores, dando inicio al siguiente ciclo.

Lo que se hará ahora será mostrar una de las dinámicas de poblaciones del sistema de *Lotka-Volterra* con el propósito de comprobar cuales son las similitudes que tiene dicho sistema respecto al modelo matemático implementado (*Algoritmo presa-depredador*) en los robots móviles. A continuación se mostrará dicha comparación:



(a) (b)

Figura 27: Comparación del modelo de *Lotka-Volterra*. (a) Dinámica de comportamientos temporal del escenario de 1 población de presas y 1 población de depredadores mediante la aplicación de un método numérico (*ode*45) en el entorno de MATLAB. (b) Dinámica de comportamientos temporal del escenario de 1 población de presas y 1 población de depredadores utilizando un grupo de robots móviles llamados kilobots en el entorno de CoppeliaSim e importado en MATLAB.

La figura 27 muestra la comparación del modelo de 1 población de presas y 1 población de depredadores mediante la aplicación de un método numérico (a) y un grupo de robots móviles (b). Se observa que la dinámica de (a) como de (b) son muy parecidas en el sentido de que la población de depredadores asciende justo después de que la población de presas desciende, y posteriormente la población de presas se recupera por el descenso de los depredadores. A pesar de que haya un parecido entre ambas dinámicas la parte de los kilobots no es posible que alcance un movimiento suave en sus comportamiento, como lo fue en la aplicación del método numérico, debido a que las interacciones son utilizando individuos reales en un entorno de simulación. De echo eso sustenta en la opinión de (Von Arb, 2013) quien explica en su trabajo que cuando individuos reales, ya sean animales o robots en este caso, participan bajo

el marco de la relación presa-depredador no es posible que alcancen un movimiento suave como la figura 27a, debido a que en este caso los robots móviles interactúan en ciertos instantes de tiempo, ademas de que esas interacciones consisten en datos reales sobre el tiempo, mientras que la figura 27a consiste en datos ideales y continuos, ocasionando la diferencia entre las subfiguras (a) y (b).

Como dato adicional se mencionará la configuración de parámetros con la cual se obtuvo la gráfica de la figura 27a, dicha configuración fue implementada en MATLAB utilizando las ecuaciones (1) y (2) ubicadas en el capítulo 3 y en la sección 4.1 de este trabajo de investigación. El conjunto de parámetros fue el siguiente: $r_1 = 0.8$, $a_1 = 0.2$, $r_2 = 0.3$, $a_2 = 0.028$, $t_0 = 0$ (tiempo inicial) y $t_f = 50$ (tiempo final), además de las condiciones iniciales $x(0) = 30$ (30 presas) y $y(0) = 4$ (4 depredadores). Por otro lado para obtener la figura 27b se utilizaron los algoritmos ubicados en las figuras 21, 22 y 23, junto con las condiciones de trabajo de la figura 24. El algoritmo implementado en cada kilobot se puede encontrar en el anexo 10.1 a este trabajo de investigación con el nombre "Algoritmo presa-depredador en robots móviles (CoppeliaSim)".

Una vez explicado el caso anterior sobre la metodología que se siguió para realizar la comparación del modelo matemático de *1 población de presas y 1 población de depredadores* se procederá a anexar una tabla de valores, con la cual se observarán diferentes resultados obtenidos utilizando la configuración de kilobots de la figura 24. Anteriormente se mencionó que cada prueba generaba una diferente serie de resultados, en este caso con ayuda de una tabla de valores se observarán las cantidades de presas y depredadores que van tomando los kilobots durante diferentes instantes en intervalos de 100 segundos, durante 5 pruebas, tomando como inicio el valor de $t = 0$ y el valor final de $t = 500$. A continuación se mostrará la tabla de valores:

Tiempo					
Pruebas	100 s	200 s	300 s	400 s	500 s
1	PV:30 PM:0 DV:4 DM:0	PV:27 PM:2 DV:5 DM:0	PV:19 PM:6 DV:9 DM:0	PV:20 PM:4 DV:9 DM:1	PV:0 PM:0 DV:14 DM:20
2	PV:30 PM:0 DV:4 DM:0	PV:27 PM:2 DV:5 DM:0	PV:10 PM:5 DV:12 DM:7	PV:21 PM:2 DV:11 DM:0	PV:13 PM:4 DV:13 DM:4
3	PV:30 PM:0 DV:4 DM:0	PV:24 PM:2 DV:5 DM:3	PV:24 PM:3 DV:7 DM:0	PV:9 PM:6 DV:17 DM:2	PV:0 PM:1 DV:7 DM:26
4	PV:30 PM:0 DV:4 DM:0	PV:26 PM:3 DV:4 DM:1	PV:24 PM:2 DV:8 DM:0	PV:20 PM:4 DV:9 DM:1	PV:16 PM:3 DV:14 DM:1
5	PV:29 PM:1 DV:4 DM:0	PV:25 PM:3 DV:6 DM:0	PV:20 PM:2 DV:11 DM:1	PV:17 PM:2 DV:15 DM:0	PV:20 PM:4 DV:9 DM:1

Cuadro 1: Pruebas del escenario 1 población de presas y 1 población de depredadores (PV: Presas vivas, PM: Presas muertas, DV: Depredadores vivos, DM: Depredadores muertos).

La información contenida en la tabla anterior muestra que para diferentes valores de tiempo (cada 100 segundos) las interacciones de los kilobots en diferentes pruebas, utilizando la misma cantidad de condiciones iniciales y los mismos parámetros de reproducción, depredación y competencia entre depredadores, los resultados son variados, sin embargo, se cumplen las características que explica el modelo de *Lotka-Volterra*, las cuales son que: a medida que la población de presas disminuye la población de depredadores aumenta, también, cuando en instantes de tiempo la población de presas aumenta la población de depredadores disminuye (por efectos de recuperación de la población de presas), también se debe considerar que en momentos cuando hay pocas presas o prácticamente no hay (prueba 1 en 500 segundos y prueba 3 en 500 segundos) la cantidad de depredadores muertos aumenta considerablemente, lo cual se debe a la falta de suministro alimenticio. Las características anteriores se pueden observar en la dinámica temporal del modelo correspondiente a la figura 27.

6.1.1. Segunda prueba del escenario 1 especie de presas y 1 especie de depredadores

Una vez realizado el análisis anterior referente al escenario de 1 población de presas y 1 población de depredadores, se realizará una prueba diferente, con el propósito de observar que comportamientos adquieren

los kilobots cuando algunas configuraciones cambian.

Para una nueva prueba se propondrán diferentes condiciones iniciales, diferentes posiciones de inicio (de orientación también), sin embargo, se utilizará el mismo algoritmo de control que fue utilizado en la prueba de la figura 24. A continuación se mostrará una figura con las caracteristicas de la prueba antes mencionada:

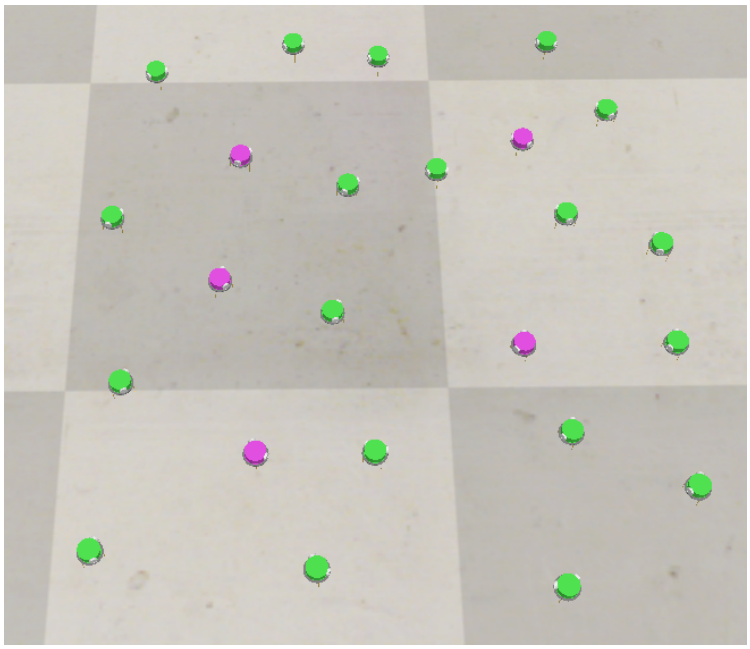


Figura 28: Segunda prueba del escenario 1 población de presas y 1 población de depredadores.

La figura 28 muestra la nueva configuración de kilobots empleada para observar la dinámica de los kilobots. En este caso se utilizó 19 presas y y 5 depredadores, es decir, $x(0) = 19$ y $y(0) = 5$.

Para una mejor representación de los kilobots utilizados se mostrará de manera gráfica las condiciones iniciales utilizadas en esta prueba. A continuación se mostrarán dichas condiciones:

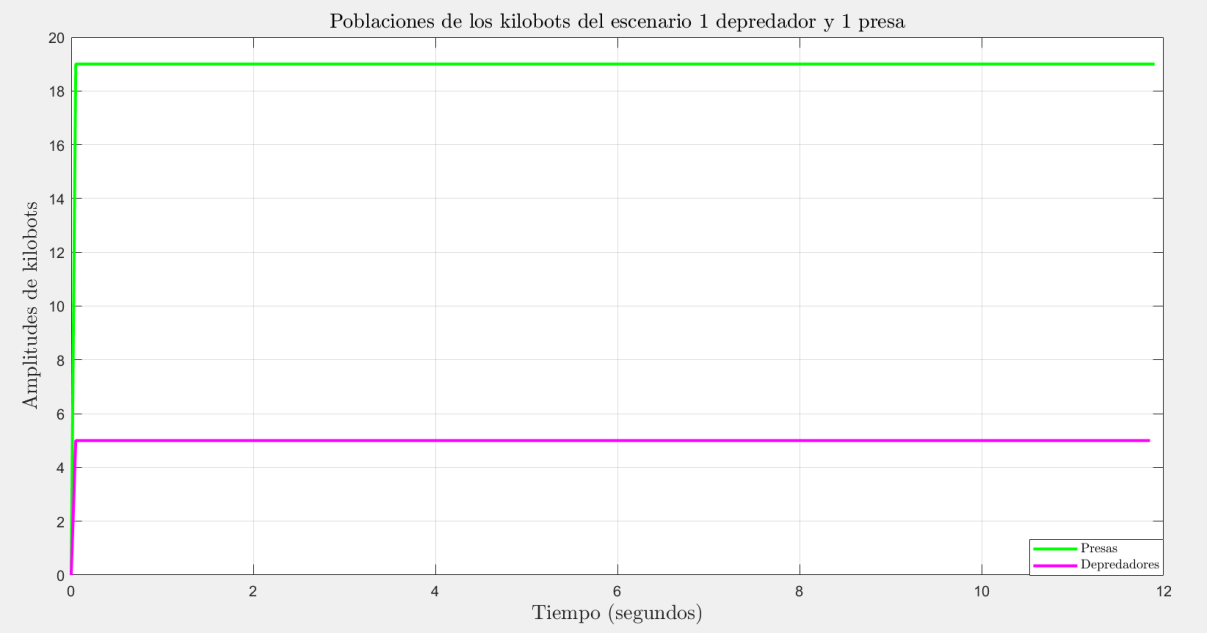
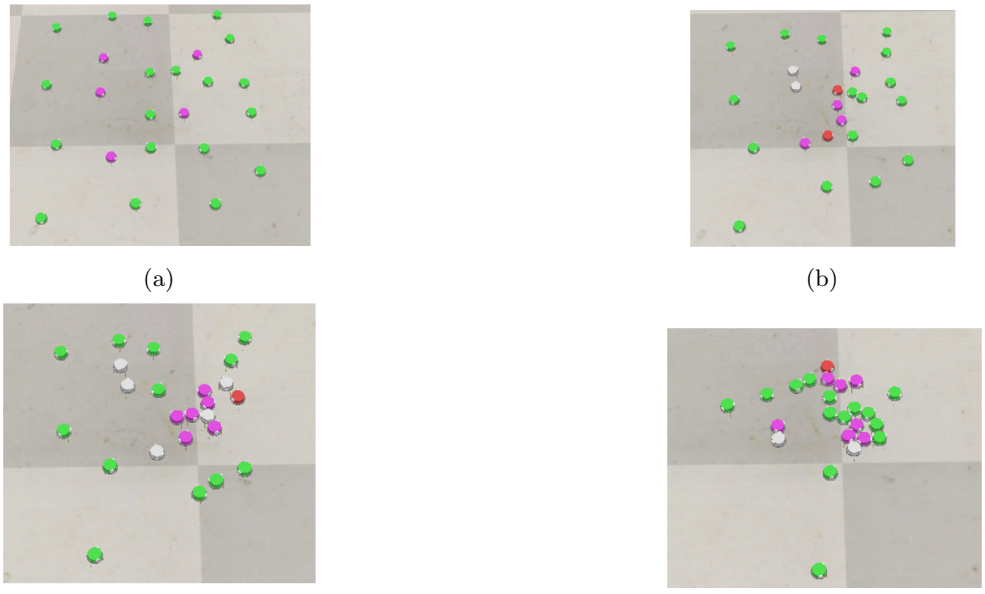


Figura 29: Condiciones iniciales utilizadas de la segunda prueba del escenario 1 presa y 1 depredador.

La figura 29 muestra las condiciones iniciales de la segunda prueba del escenario de 1 población de presas y 1 población de depredadores en el instante $t = 0$, posteriormente con dichas condiciones de kilobots se procederá a mostrar los movimientos de ellos en diferentes instantes de tiempo con el fin de mostrar las interacciones que éstos tienen:



(a) (b)

(c) (d)

Figura 30: Interacciones de respuesta del algortimo presa-depredador implementado en un grupo de pequeños robots móviles del escenario 1 población de presas y 1 población de depredadores (prueba 2).

Las interacciones de los kilobots anteriores corresponden a las del escenario mostrado en la figura 28, donde la subfigura (a), tomada en el instante aproximado de 100 segundos, únicamente muestra desplazamiento entre los agentes, aún no hay encuentros por parte de los kilobots, algunos encuentros se visualizan en (b), el cual fue aproximadamente en 200 segundos, se observa que algunos depredadores han tenido encuentros con presas (kilobots en color rojo) pero también hay depredadores que han tenido encuentros entre sí (kilobots en color blanco). En la subfigura (c), tomada aproximadamente en el instante de 300 segundos, se observa que los depredadores han aumentado por parte de las interacciones que han tenido con las presas, pero como los depredadores han aumentado de igual forma los efectos de competencia entre ellos, lo cual se visualiza con los kilobots de color blanco, posteriormente en (d), la cual fue tomada en un instante más prolongado, aproximadamente en 500 segundos, se observa que los depredadores se han recuperado a causa de que han encontrado presas en el área. En (d) se optó por tomar un instante prolongado (500 segundos) ya que mostraba mejor las interacciones que se dieron en el instante de 400 segundos.

En relación a las interacciones de la prueba anterior se mostrará de manera panorámica una de las dinámicas temporales que generan los kilobots cuando presentan la configuración de la figura 28.

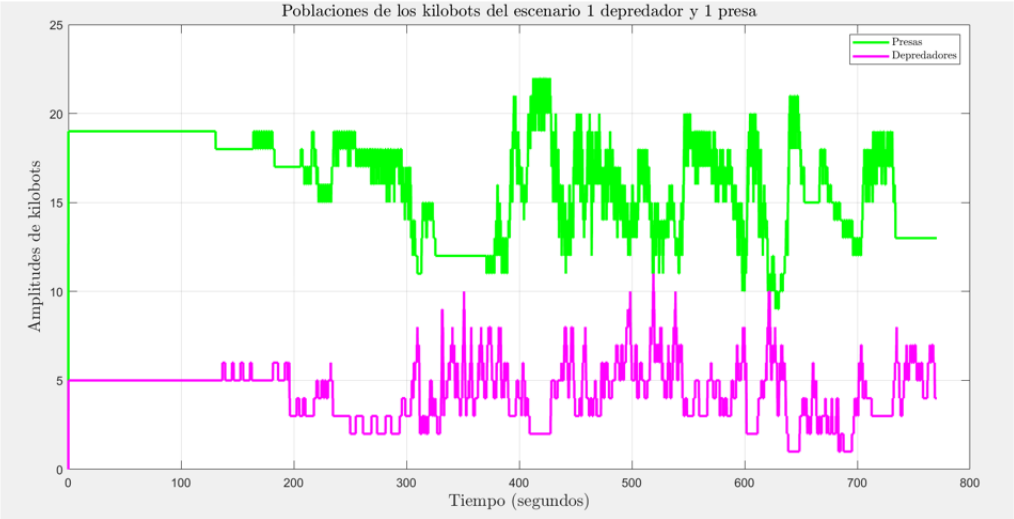
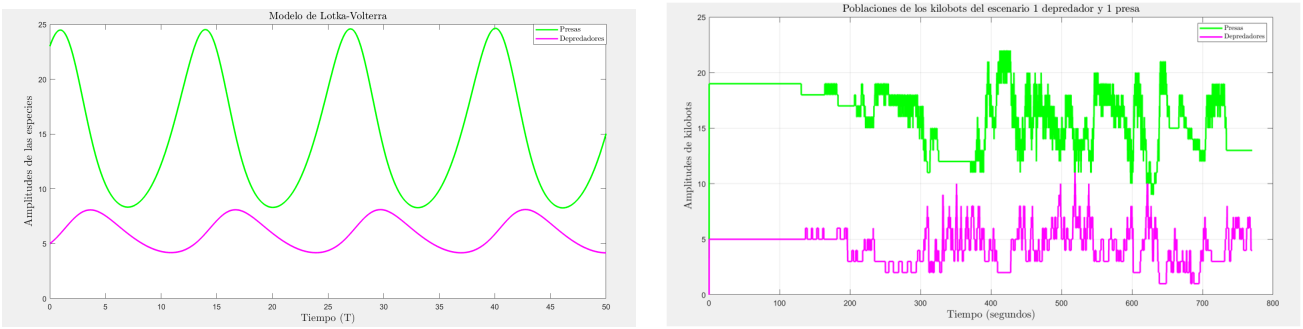


Figura 31: Grafica del escenario 1 población de presas y 1 población de depredadores (prueba 2).

En la figura 31 que muestra la dinámica anterior, se observan ciertas oscilaciones que no fueron tan pronunciadas como en la prueba 1, sin embargo, eso se debe a que se utilizó un menor número de robots móviles que en la prueba anterior, el hecho de que el número de robots sea menor influye de forma en que las interacciones de los kilobots será menor que si se tuvieran más, como fue el caso de la primer prueba. Pero también se observa que las características del sistema de *Lotka-Volterra* se cumplen, en el sentido de que cuando hay un descenso de las presas (inicio de la gráfica) hay un aumento en la población depredadora, por depredación (transición de estado del kilobot), posterior a ello, cuando la población de depredadores está en su punto más alto hay más frecuencia que los depredadores interactúen entre ellos y al no encontrar población de presas tienden al descenso, permitiendo que la población de presas vuelva a crecer (instante de 400 segundos). Lo que ocurre seguido a ello son las mismas interacciones de los kilobots pero a diferentes amplitudes, lo cual se debe a la explicación de (Von Arb, 2013), quien menciona que al utilizar agentes reales (robots en este caso) las condiciones son no ideales y no continuas, lo cual también influye en que el gráfico no tenga una suavidad pura.



(a) (b)

Figura 32: Comparación de la segunda prueba del escenario 1 población de presas y 1 población de depredadores, utilizando métodos numéricos (a) y un grupo de robots móviles llamados kilobots (b).

La figura 32 muestra la comparación de la nueva prueba aplicando métodos numéricos para evaluar el modelo matemático (a) y un grupo de robots móviles (b), cabe mencionar que el algoritmo utilizado fue el mismo para evaluar la nueva prueba generando diferentes resultados (mostrado en (b)), como se mencionó anteriormente, por el hecho de usar una menor cantidad de kilobots para la evaluación de modelo matemático, de echo, eso se sustenta en la evaluación con métodos numéricos (a), debido a que dicha gráfica tiene diferentes los parámetros de entrada, los cuales fueron: $r_1 = 0.8$, $a_1 = 0.135$, $r_2 = 0.3$ y $a_2 = 0.02$. Únicamente los términos que cambiaron fueron a_1 y a_2, siendo reducidos en su valor numérico, éstos términos del modelo matemático son los términos de interacción entre presas y depredadores, los cuales fueron reducidos, ya que en la evaluación del modelo con kilobots las interacciones fueron menores, ya que se usó una menor cantidad de robots, el gráfico resultante fue el que se observa en (a), teniendo un gran parecido al que se observa en (b) que es la aproximación del modelo matemático utilizando kilobots.

6.2. Escenario de 2 especies de presas y 1 especie de depredadores

Después de haber realizado la explicación del modelo referente al escenario de poblaciones de 1 especie de presas y 1 especie de depredadores se realizará una metodología equivalente aplicado al modelo del escenario *2*

poblaciones de presas y 1 población de depredadores. Para lo cual se utilizará las mismas herramientas en cuanto a los muros para delimitar la acción de los kilobots, el controlador para ejecutar la puesta en marcha de los kilobots y la unidad graficadora para obtener las interacciones de los kilobots de forma gráfica. A continuación, se mostrará una figura donde se visualizará las herramientas antes mencionadas:

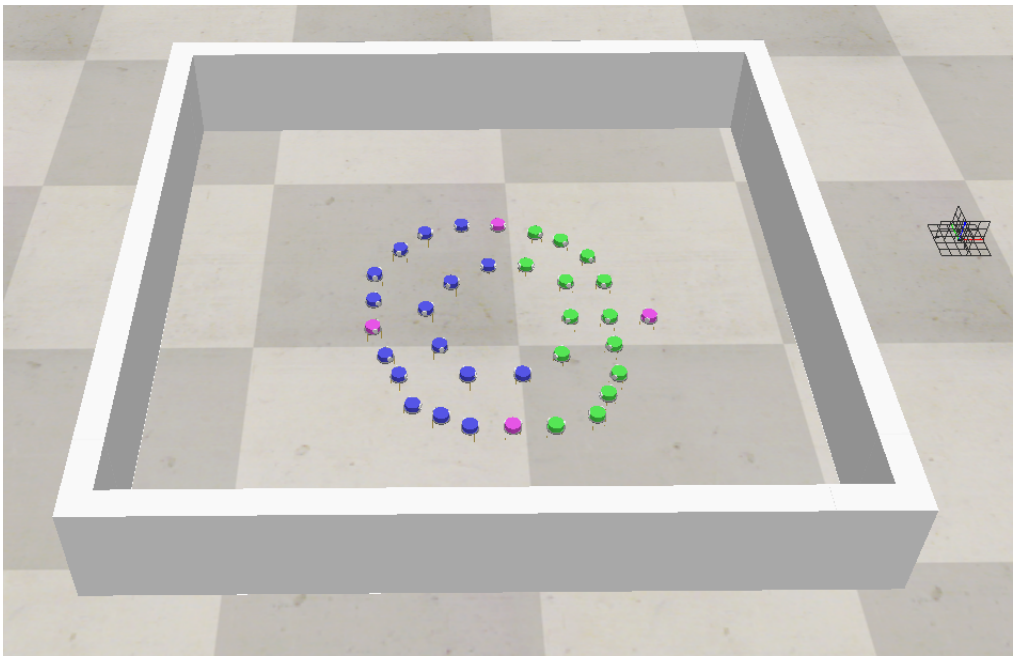


Figura 33: Herramientas de trabajo del escenario 2 especies de presas y 1 especie de depredadores.

La figura 33 es muy similar en cuanto a la figura 24, en cuanto a las herramientas utilizadas, la única diferencia radica en la adición de una nueva especie, correspondiente a una población de presas, la cual afecta de forma positiva a la población de depredadores ya que es considerada como fuente de alimentación para ellos, en cuanto a la otra población de presas no hay consecuencia alguna debido a que no hay competencia o depredación entre ellas. Aunado a ello se mencionarán las condiciones iniciales de este experimento a continuación (las condiciones fueron propuestas de forma aleatoria): $x(0) = 14$ (kilobots en color verde), $y(0) = 16$ (kilobots en color azul) y $z(0) = 4$ (kilobots en color morado).

A partir de la información anterior, sobre las herramientas utiliza-

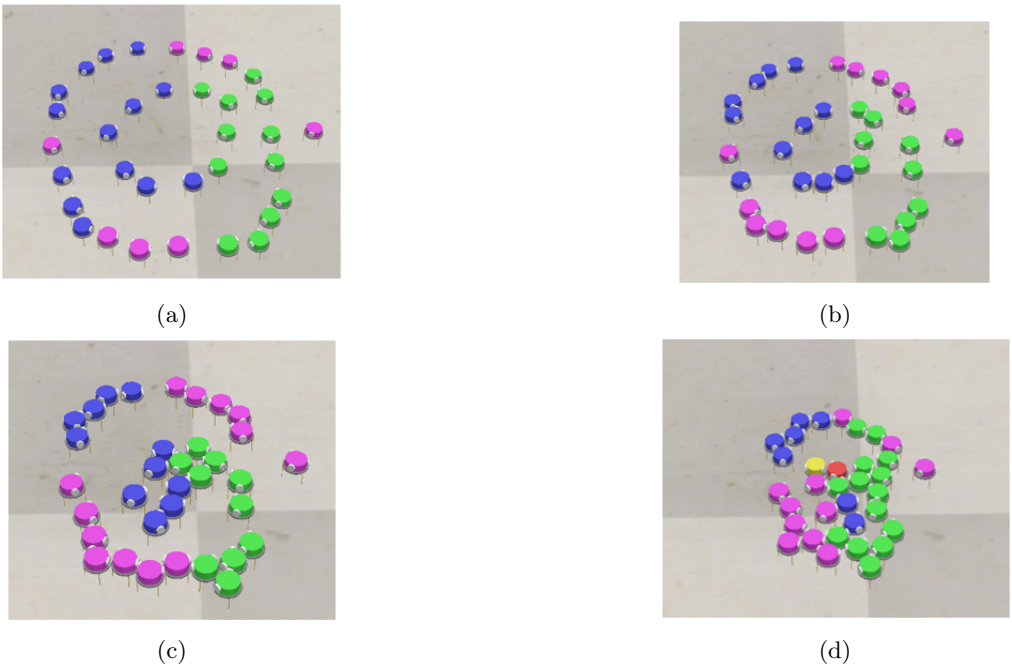
das, la adición de una especie y las condiciones iniciales utilizadas, es muy importante señalar éstas últimas debido a que éstos sistemas son muy sensibles a las variaciones de las condiciones iniciales como de los parámetros iniciales, por lo que se agregará una representación gráfica de estas condiciones para reconocer cuando son los primeros instantes de tiempo, en los cuales no hay interacción alguna entre las dos poblaciones de presas y la población de depredadores representados por el grupo de robots móviles. A continuación se mostrará dicha representación gráfica:

Figura 34: Gráfica del escenario 2 especies de presas y 1 especie de depredadores en los primeros instantes de tiempo.

En la figura 34 se muestran únicamente las condiciones iniciales utilizadas en esta prueba simulada por el escenario de *2 poblaciones de presas y 1 población de depredadores*, las cuales se habían mencionado anteriormente, sin embargo en la figura anterior se observa que dichas condiciones permanecen constantes durante la marca de $t = 10$ segundos, eso significa que hasta ese tiempo no ha habido alguna interacción por parte de las poblaciones de kilobots. A continuación se mostrarán como se manifiestan dichas interacciones mediante una figura y adicionalmen-

te se obtendrá de manera panorámica como se comportan en una prueba.

La parte complicada de esta relación biológica es precisamente la adición de la nueva especie, ya que las interacciones entre los robots móviles comienzan a ser cada vez más, a comparación de la relación anterior donde únicamente se ubicaban dos especies. También se debe tener en cuenta que cada prueba origina resultados variados, lo cual se debe a la marcha aleatoria de los kilobots, sin embargo, la adaptación del modelo matemático implementada en los robots móviles se mostrará a continuación:



(a) (b) (c) (d)

Figura 35: Algoritmo de 2 poblaciones de presas y 1 población de depredadores representado por un grupo de pequeños robots móviles. Anexo 10.2 Algoritmo de 2 poblaciones de presas y 1 población de depredadores (CoppeliaSim).

La figura 35 explica las interacciones que tienen los robots móviles cuando están bajo el régimen del escenario de *2 poblaciones de presas y 1 población de depredadores*. Dicha figura hace referencia a uno de los múltiples resultados que se pueden generar cuando se inicia el experimento. La lógica del algoritmo es muy similar a la del algoritmo de *Lotka-Volterra*, la cual puede ubicarse en el anexo a este trabajo

de investigación. A continuación se explicará la figura anterior con el propósito de definir lo que esta sucediendo. En este caso en la subfigura (a), tomada en el instante aproximado de 50 segundos, se observa que la población de depredadores ha comenzado a crecer, es de esperarse que crezca rápido ya que en este caso se tiene más abundancia de presas (2 poblaciones), quienes hacen el papel del suministro de la población de depredadores. En la subfigura (b), tomada en el instante aproximado de 100 segundos, se observa que la población de depredadores esta creciendo aún más, lo que fomentará el evento de la competencia instraespecífica en instantes posteriores. En la subfigura (c) se observa que las presas están comenzando a agruparse mientras que la población de depredadores no ha sufrido grandes cambios, y posteriormente en la subfigura (d), tomada en el instante aproximado de 200 segundos, se observa que la población de depredadores ha comenzado a disminuir, mientras que la primer población de presas ha comenzado a recuperarse, eso se debe a que los depredadores tuvieron más interacciones con la segunda población de presas (kilobots en azul) que con la primera, por lo que la primer población de presas pudo reproducirse utilizando a los individuos de la segunda población.

Una vez explicado las interacciones de la prueba anterior se agregará una gráfica cuyo contenido será las dinámicas de las poblaciones de kilobots bajo el régimen explicado anteriormente, la cual fue obtenida con el algoritmo de control distribuido implementado en los kilobots, el cual puede ser encontrado en el anexo a este trabajo de tesis. A continuación, se mostrará la dinámica de los kilobots:

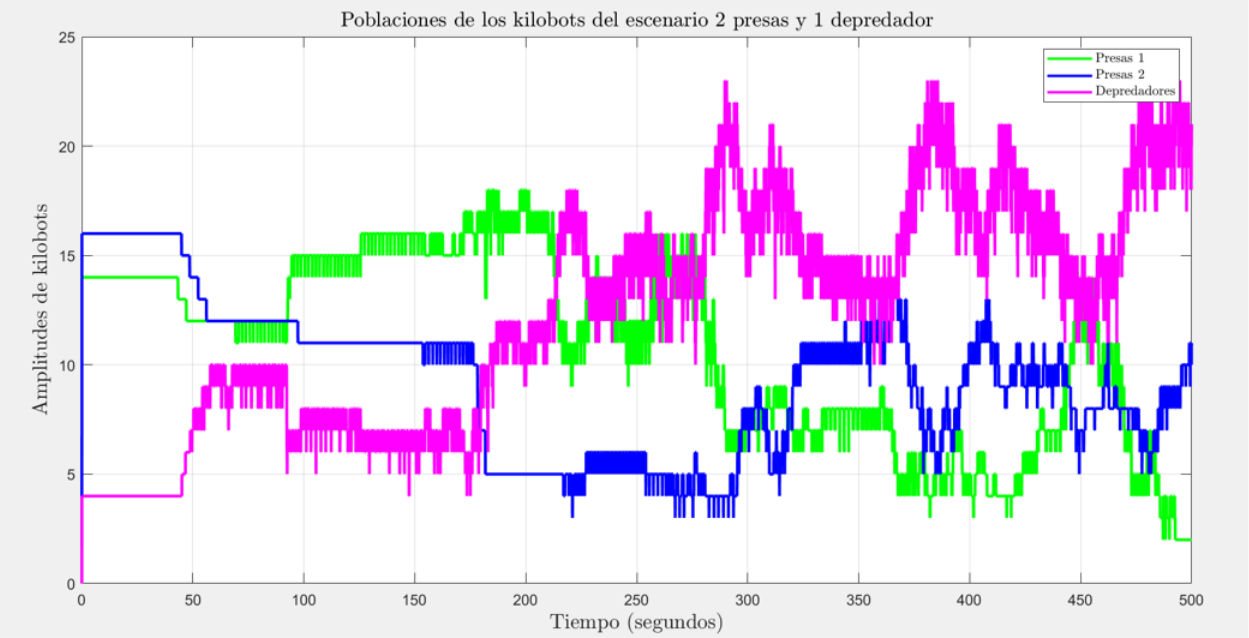
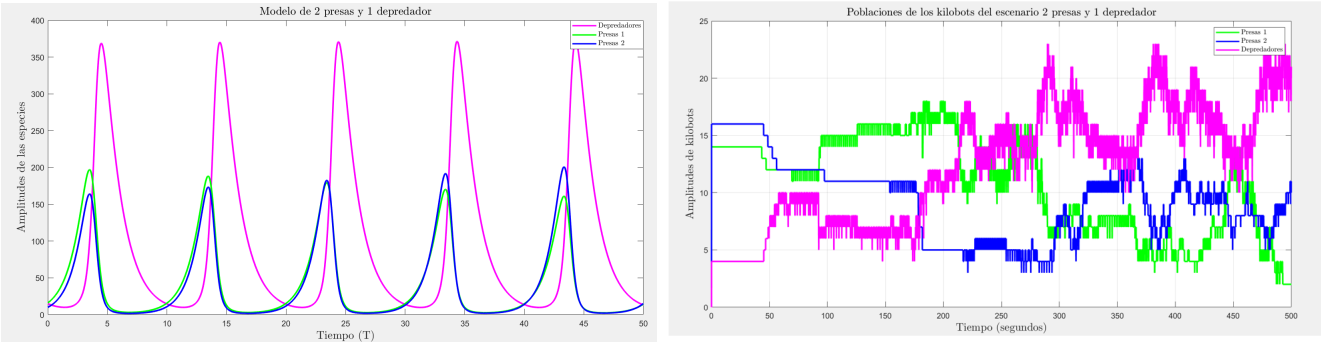


Figura 36: Dinámica de comportamientos de los kilobots en el escenario de 2 poblaciones de presas y 1 población de depredadores. Anexo 10.2 Algoritmo de 2 poblaciones de presas y 1 población de depredadores.

Lo que se puede observar en la gráfica correspondiente a la figura 36 son las interacciones entre las tres especies en un entorno cerrado, donde la población de los depredadores es la más notoria, estando por encima de las poblaciones de las presas, lo cual era esperado, ya que al haber dos poblaciones de presas funcionan como fuente de alimento a los depredadores, fomentando la llegada de nuevos individuos. Por otro lado las poblaciones de presas, ubicadas por debajo de la dinámica de los depredadores, se comportan de forma oscilatoria cada vez que la población de depredadores tiene un descenso, eso se refiere a que los depredadores no encuentran a las presas a su alrededor, por lo que su población comienza a disminuir, dicha disminución fomenta la llegada de nuevas presas, ya que cuando hay descenso de depredadores hay aumento de presas. La explicación anterior se sustentará mediante la comparación entre el modelo matemático en condiciones ideales y continuas, y la aplicación de los robots móviles, al igual que como se hizo con el caso de *1 población de presas y 1 población de depredadores*. A continuación, se mostrará dicha comparación:



(a) (b)

Figura 37: Comparación del modelo de *2 poblaciones de presas 1 población de depredadores*. (a) Dinámica de comportamientos temporal del escenario de 2 poblaciones de presas y 1 población de depredadores mediante la aplicación de un método numérico (*ode*45) en el entorno de MATLAB. (b) Dinámica de comportamientos temporal del escenario de 2 poblaciones de presas y 1 población de depredadores utilizando un grupo de robots móviles llamados kilobots en el entorno de CoppeliaSim e importado en MATLAB.

En la figura 37 se puede observar la comparación del modelo matemático compuesto por dos poblaciones de presas y 1 población de depredadores, el cual puede ser ubicado en el capítulo 3 en la sección 4.3 de este trabajo de investigación, representado por las ecuaciones (6), (7) y (8). La comparación se realizó mediante la aplicación del método numérico *ode*45 en MATLAB (a) y mediante la implementación de un algoritmo de control distribuido en un grupo de robots móviles llamados kilobots en el entorno de CoppeliaSim (b) y posteriormente importando los datos a MATLAB para poder reflejar los comportamientos de los robots en dicho programa.

Puede observarse que en (b) las amplitudes de los depredadores comparten un parecido a las de (a) al ser las más sobresalientes, lo cual es gracias a la influencia de una nueva población de presas, sin embargo puede observarse que en (b) los descensos de los depredadores no son muy abruptos en cuanto a la falta de presas como se observa en (a), lo cual se debe a que no hay muchos decesos de las presas, lo que beneficia a los depredadores siendo los más dominantes. Es muy importante definir de una forma correcta las tasas de reproducción y las de depredación, ya que un mínimo cambio en ellos puede alterar completamente la configu-

ración de los robots móviles, ya que el algoritmo de control distribuido es muy sensible si recibe cambios muy ligeros. También en este escenario se comparte la explicación del escenario anterior, de que al no haber condiciones ideales como en (a) la captura de los movimientos no tienen una suavidad pura como se observa en (b) ya que se usan agentes reales y la captura de movimiento no es totalmente continua como en (a). El algoritmo de control puede ser encontrado en el anexo de este trabajo de investigación.

Al igual que en el escenario anterior se procederá a anexar una tabla, con la cual se realizarán diferentes pruebas utilizando las mismas configuraciones de condiciones, posiciones y parámetros de inicio, esto con la intención de observar la diversa variedad de resultados que se pueden obtener, además de observar como se comportan las especies en un cierto periodo de tiempo. A continuación se mostrará la tabla de valores para este escenario:

	Tiempo			
Pruebas	100 s	200 s	300 s	400s
1	P1V:14 P1M:0 P2V:13 P2M:0 DV:7 DM:0	P1V:16 P1M:0 P2V:0 P2M:0 DV:18 DM:0	P1V:7 P1M:0 P2V:0 P2M:0 DV:25 DM:2	P1V:10 P1M:0 P2V:0 P2M:0 DV:23 DM:1
2	P1V:14 P1M:0 P2V:11 P2M:0 DV:8 DM:1	P1V:17 P1M:0 P2V:6 P2M:0 DV:10 DM:1	P1V:5 P1M:0 P2V:4 P2M:0 DV:23 DM:2	P1V:11 P1M:0 P2V:12 P2M:0 DV:11 DM:0
3	P1V:14 P1M:0 P2V:11 P2M:0 DV:9 DM:0	P1V:12 P1M:1 P2V:5 P2M:0 DV:16 DM:0	P1V:5 P1M:0 P2V:8 P2M:0 DV:20 DM:1	P1V:7 P1M:0 P2V:14 P2M:0 DV:13 DM:0
4	P1V:14 P1M:0 P2V:11 P2M:0 DV:8 DM:1	P1V:14 P1M:0 P2V:11 P2M:0 DV:9 DM:0	P1V:4 P1M:1 P2V:12 P2M:0 DV:17 DM:0	P1V:9 P1M:0 P2V:18 P2M:0 DV:7 DM:0
5	P1V:14 P1M:0 P2V:11 P2M:0 DV:8 DM:1	P1V:14 P1M:1 P2V:7 P2M:0 DV:12 DM:0	P1V:8 P1M:0 P2V:8 P2M:0 DV:18 DM:0	P1V:9 P1M:0 P2V:18 P2M:0 DV:6 DM:1

Cuadro 2: Pruebas del escenario 2 presas y 1 depredador (P1V: Presas 1 vivas, P1M: Presas 1 muertas, P2V: Presas 2 vivas, P2M: Presas 2 muertas, DV: Depredadores vivos, DM: Depredadores muertos).

Los resultados obtenidos anteriormente corresponden a una serie de pruebas realizadas del escenario correspondiente al modelo matemático

de 2 poblaciones de presas y 1 población de depredadores implementado en un grupo de robots móviles llamados kilobots, la tabla anterior se realizó con la intención de observar diferentes tipos de resultados al utilizar un grupo de kilobots implementando el algoritmo de control distribuido relacionado al modelo matemático antes mencionado.

Al existir una nueva especie las interacciones de los robots son más complejas ya que dicha especie influye sobre el resto, en este caso afecta de forma positiva a la población de depredadores, ya que funcionan como alimento para ellos, en el caso de las presas no hay consecuencia alguna.

Las pruebas realizadas en la tabla de valores anteriores muestran una relación similar a la del modelo matemático de *Lotka-Volterra* (1 población de presas y 1 población de depredadores), las cuales son: mientras exista un número abundante de presas (2 poblaciones) los depredadores aumentarán en gran medida, mientras que cuando hay pocas presas los depredadores disminuirán por ausencia de ellas y por el efecto de competencia. Es muy importante hacer hincapié que el efecto de competencia no se muestra explicitamente en el modelo matemático, éste se ha considerado para aumentar el realismo que existen entre las interacciones de los robots móviles y para regular la población de los depredadores cuando hay muchos de ellos desplazándose por el área.

6.2.1. Segunda prueba del escenario 2 especies de presas y 1 especie de depredadores

Para la realización de una nueva prueba de este escenario se hará uso del mismo algoritmo de control que la prueba anterior, con la diferencia de que las condiciones iniciales serán menores, es decir, se usará una menor cantidad de kilobots, con la intención de observar como cambia la respuesta de los robots móviles en relación a la prueba anterior. A continuación, se mostrará la figura utilizada para representar esta nueva

prueba:

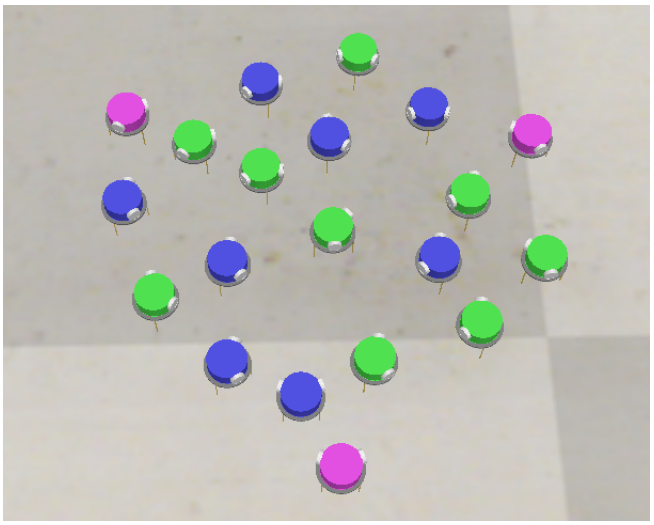
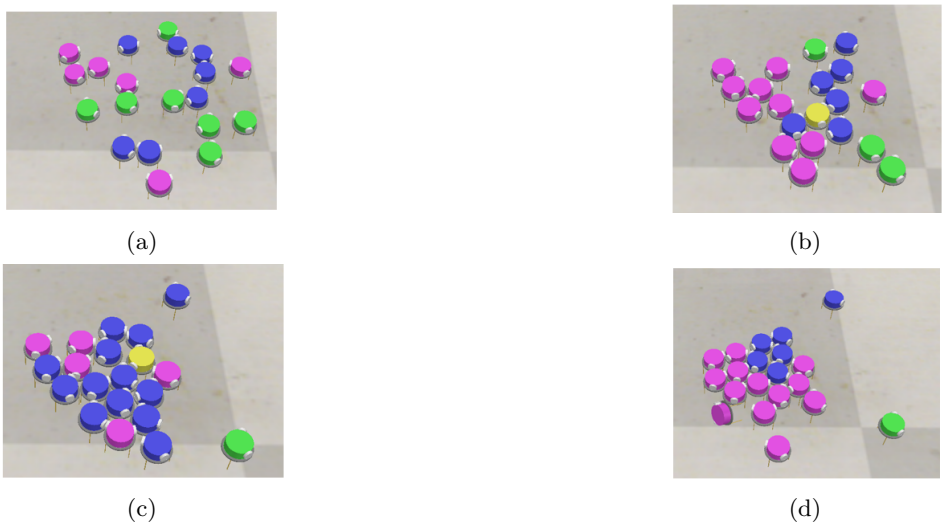


Figura 38: Segunda prueba del escenario 2 poblaciones de presas y 1 población de depredadores.

En la figura 38 se puede observar la cantidad de robots móviles que fueron utilizados para mostrar una nueva prueba en relación al escenario *2 poblaciones de presas y 1 población de depredadores*, en esta ocasión se ha seleccionado un menor número de kilobots a comparación de la primer prueba para representar el escenario, con la siguiente equivalencia de condiciones iniciales: $x(0) = 9$, $y(0) = 8$ y $z(0) = 3$, es decir, 9 presas de la primer población, 8 presas de la segunda población y 3 depredadores, respectivamente, formando un total de 20 kilobots. A continuación se mostrará de forma gráfica dichas condiciones iniciales:

Figura 39: Segunda prueba del escenario 2 poblaciones de presas y 1 población de depredadores (condiciones iniciales).

En la figura 39 se observa que las condiciones iniciales anteriores muestran que desde $t = 0$ los kilobots no han tenido ninguna interacción con sus vecinos durante el primer segundo de la nueva prueba, posteriormente se anexará una figura donde las interacciones son observables para determinar lo que puede suceder, se debe hacer hincapié en que las interacciones de los kilobots pueden ser diferentes con la ejecución de cada simulación de la prueba, por lo que en seguida se mostrará una serie de interacciones que es posible obtener de acuerdo a la configuración del algortimo implementado:



(a) (b)

(c) (d)

Figura 40: Conjunto de interacciones del algoritmo del escenario de 2 poblaciones de presas y 1 población de depredadores representado por un pequeño grupo de robots móviles. Anexo 10.2 Algoritmo de 2 poblaciones de presas y 1 población de depredadores.

En la figura 40 se observa una serie de interacciones que los kilobots adquieren cuando están bajo la implementación del algoritmo presa-depredador de *2 poblaciones de presas y 1 población de depredadores*, en esta ocasión cada subfigura fue capturada en instantes aproximados de 50 segundos cada una. En (a), aproximadamente en el instante de 50 segundos se observa que la población de depredadores ha comenzado a crecer mientras que ambas poblaciones de presas ha disminuido, lo cual es lógico ya que al tener dos conjuntos de presas beneficia más a los robots que siguen el comportamiento del "depredador". En la subfigura (b), capturada en un instante aproximado de 100 segundos, se observa que los depredadores han crecido aún más que en el instante anterior, mientras que la primer población de presas ha sido la más perjudicada al solo quedar 3 kilobots de ellos (kilobots en color verde). La siguiente subfigura (c), capturada en un instante aproximado de 150 segundos, muestra que la población de depredadores ha disminuido, mientras que la segunda población de presas ha crecido a causa de los pocos depredadores del área. Posteriormente en (d) los depredadores han vuelto a crecer a consecuencia de que hubo presas abundantes en el instante anterior (kilobots en color azul), mientras que la población de presas en color

verde (primer población de presas) no ha podido recuperarse debido a que solo quedó 1 unidad de ella, siendo incapaz de reproducirse a través de los kilobots en estado de "muerto".

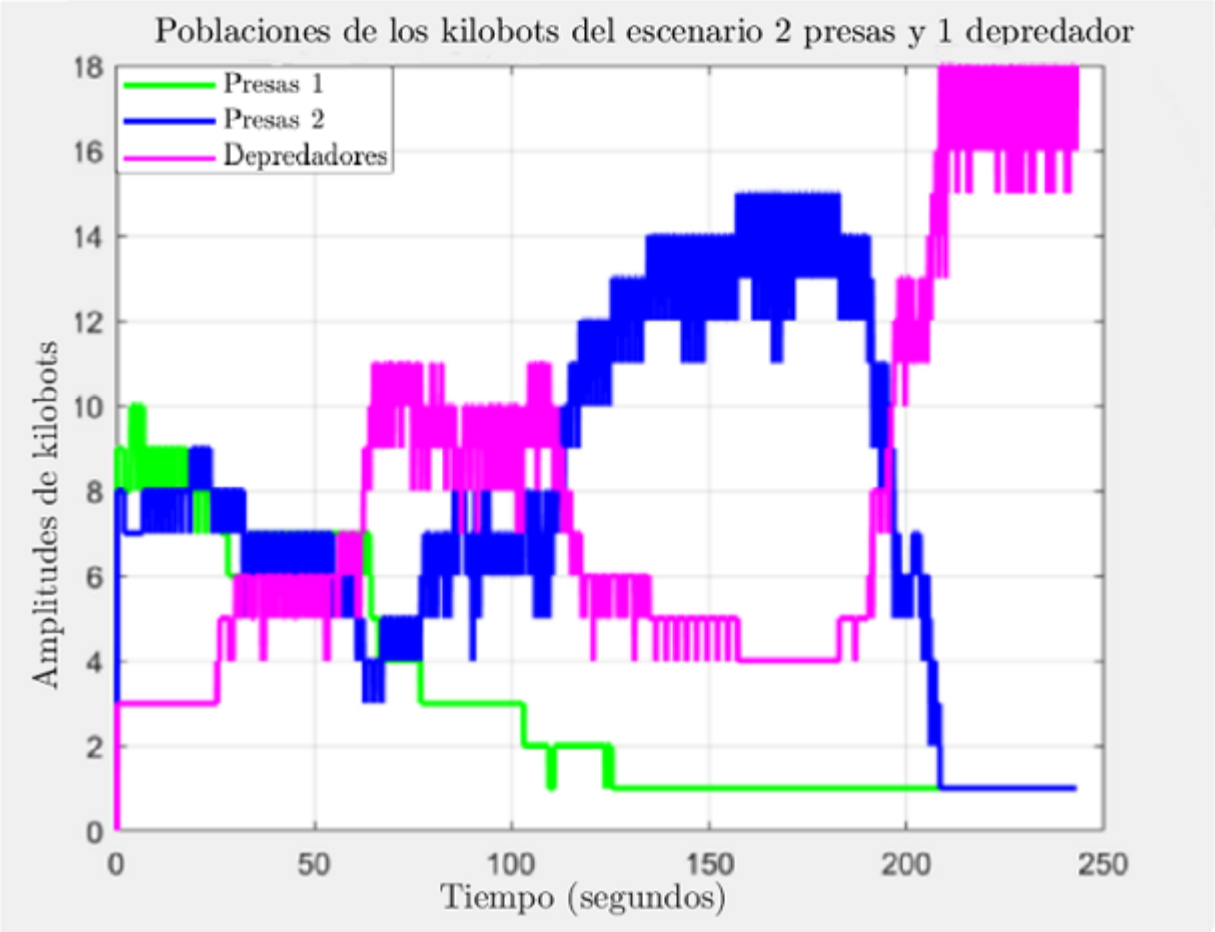


Figura 41: Gráfica del escenario de 2 poblaciones de presas y 1 población de depredadores (segunda prueba).

La figura 41 muestra como se visualiza el comportamientos de los kilobots en relación a la configuración de los kilobots de la figura 38, la descripción de dicha figura es prácticamente equivalente a la de la figura 40. Sin embargo, se debe aclarar que la respuesta obtenida no corresponde en su totalidad a la respuesta del modelo matemático, debido a que cuando ambas poblaciones de presas tienen 1 kilobot (como es el caso) es difícil que puedan recuperarse cuando hay muchos depredadores, pero es una situación que se puede presentar cuando se hacen experimentos con este tipo de algoritmos en robots móviles llamados kilobots.

6.3. Escenario de 2 especies de depredadores y 1 especie de presas

En esta sección se realizará la adaptación del modelo de relación biológica en el cual intervienen *2 poblaciones de depredadores y 1 población de presas* en un grupo de kilobots, con la intención de obtener sus interacciones y poder despegarlas con el fin de corroborar si lo que se obtiene tiene alguna semejanza con el modelo en condiciones ideales.

Con la intención de obtener las interacciones de forma gráfica de este nuevo escenario se hará uso de las mismas herramientas para acondicionar el entorno de los kilobots, a continuación se mostrará una figura que represente las herramientas a utilizar:

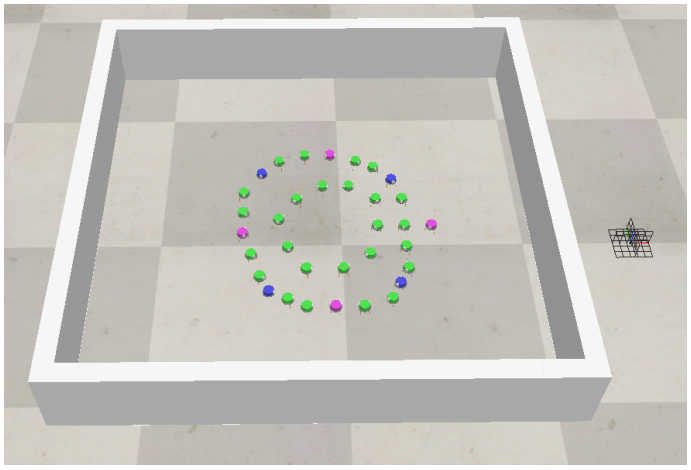


Figura 42: Herramientas del escenario compuesto por 2 poblaciones de depredadores y 1 población de presas.

La figura 42 muestra que se utilizaron las mismas configuraciones de posiciones iniciales que en los casos anteriores, con la excepción de que la nueva especie (variable) adicional añadida en este escenario es de depredadores, con el mismo nivel trófico que la población de depredadores existentes, es decir, son dos grupos de depredadores que compiten entre sí para consumir a las presas pero los depredadores no pueden devorarse entre ellos mientras están vivos (en movimiento), solo compiten cuando se encuentran a una distancia menor o igual a 6 cm (condición de distancia propuesta en el algoritmo).

Para este caso las condiciones iniciales utilizadas fueron las siguientes: $x(0) = 26$, (26 presas), $y(0) = 4$, (4 depredadores del primer grupo) y $z(0) = 4$, (4 depredadores del segundo grupo), en total fueron utilizados 34 kilobots para la prueba de este escenario.

Lo que se hará ahora será mostrar las interacciones que tienen los kilobots cuando están bajo el régimen de *2 poblaciones de depredadores y 1 población de presas*, dicho algoritmo implementado en los kilobots puede ser encontrado en el anexo de este trabajo de investigación. A continuación, se mostrará dicha figura que contiene las interacciones de los kilobots.

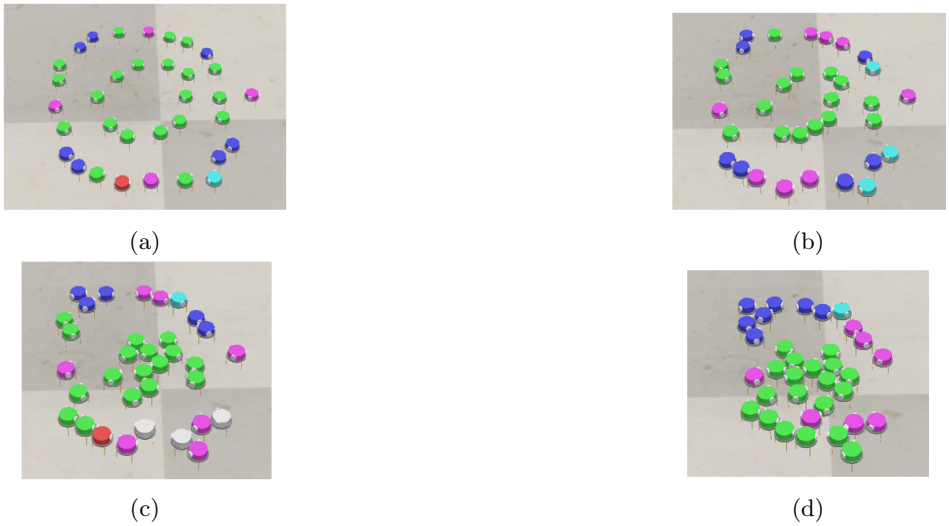


Figura 43: Algoritmo de 2 poblaciones de depredadores y 1 población de presas representado por un grupo de robots móviles. Anexo 10.3 Algoritmo de 2 poblaciones de depredadores y 1 población de presas (CoppeliaSim).

La figura 43 representa el conjunto de interacciones que adquieren los kilobots cuando están bajo condiciones del algoritmo de *2 poblaciones de depredadores y 1 población de presas*. Lo que sucede con los kilobots en la subfigura (a), la cual fue tomada aproximadamente en el instante de 50 segundos, es que la población de depredadores en color azul (segunda población de depredadores) se ha comenzado a reproducir de manera rápida, eso se debe a que han tenido un mayor encuentro con las presas que la primer población de depredadores (kilobots en color

rosa), sin embargo, se observa un kilobot en color azul claro, lo cual se debe a la competencia entre depredadores de la segunda población, se debe tener en cuenta que a mayor número de depredadores el nivel de competencia crecerá. La subfigura (b), tomada en el instante aproximado de 100 segundos, muestra que la primer población de depredadores (kilobots en color rosa) ha comenzado a crecer, debido a que han comenzado a tener más interacciones con las presas, a diferencia de la segunda población de depredadores. Sin embargo, se observa que la competencia entre depredadores de la segunda población comienza a ser más notoria, lo cual se debe a la falta de encuentros con las presas. Después se tiene la subfigura (c), tomada aproximadamente en el instante de 150 segundos, en la cual se observa las presas han tenido un ligero aumento de población, lo cual se debe los depredadores han tenido más encuentros entre ellos, ocasionando que sus poblaciones desciendan lo cual se observa en los kilobots en color blanco, que corresponden a los decesos de la primer población de depredadores, ocasionando que las presas se reproduzcan. Por otro lado en la subfigura (d), tomada en el instante aproximado de 200 segundos, se observa que la segunda población de depredadores ha crecido a causa de los efectos de interacción con las presas y a los efectos de canibalismo a través de la primer población de depredadores, lo cual funciona como recurso cuando los depredadores no tienen interacción con las presas pero tienen a su alcance a un depredador muerto, sirviendo como recurso provocando su reproducción, lo cual también se observa con la primer población de depredadores (kilobots en color rosa).

Después de observar el tipo de comportamientos que adquieren los kilobots con el algoritmo anterior, se añadirá una de las dinámicas de comportamientos que obtienen los kilobots a través del algoritmo de control implementado. A continuación se mostrará dicha dinámica de poblaciones en los robots móviles:

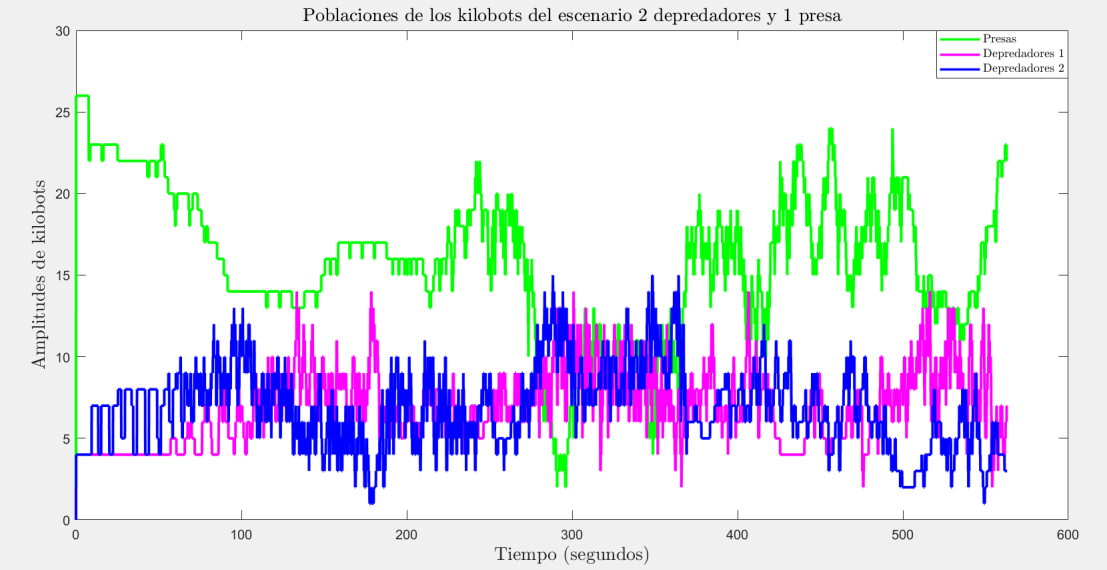
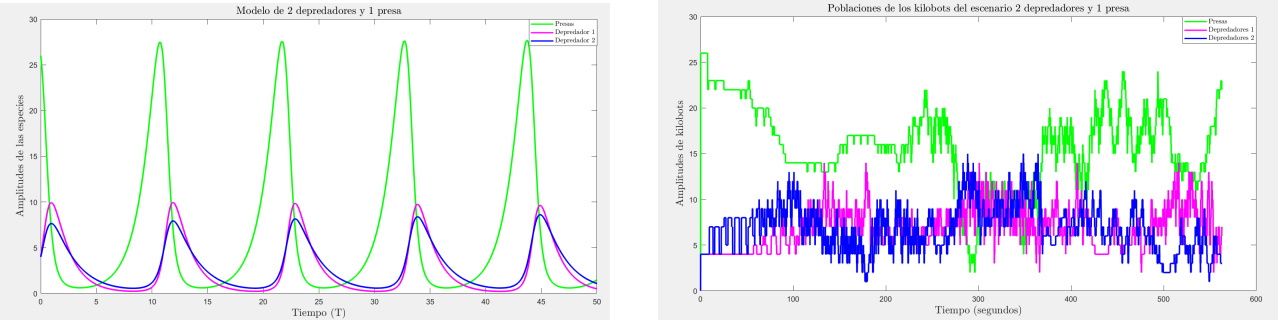


Figura 44: Dinámica de comportamientos de los kilobots en el escenario 2 poblaciones de depredadores y 1 población de presas.

La figura 44 corresponde a las dinámicas generadas de los kilobots con el algoritmo implementado de poblaciones *2 depredadores y 1 presa*. El fenómeno que ocurre en la gráfica anterior es que la población de presas sufre una disminución al principio, lo cual es esperado ya que están bajo la influencia de dos poblaciones de depredadores, lo cual beneficia a las poblaciones de depredadores, observándose un aumento en sus poblaciones a partir de la marca de 100 segundos, posteriormente los depredadores sufren una disminución poco antes de los 200 segundos reflejándose un aumento en las presas en el mismo instante de tiempo. Después el siguiente ciclo se observa en el instante de 300 segundos, donde las presas sufren disminución, consecuencia de los depredadores y éstos tienen una aumento, posteriormente en la marca de 400 segundos los depredadores disminuyen, beneficiando a las presas y éstas vuelven a crecer, generando el siguiente ciclo. Este tipo de interacciones se generan gracias al código de control implementado en los 34 kilobots por igual, la única diferencia radica en el comportamiento de cada robot.



(a) (b)

Figura 45: Comparación del modelo de *2 poblaciones de depredadores y 1 población de presas*. (a) Dinámica de comportamientos temporal del escenario de 2 poblaciones de depredadores y 1 población de presas mediante la aplicación de un método numérico (*ode*45) en el entorno de MATLAB. (b) Dinámica de comportamientos temporal del escenario de 2 poblaciones de depredadores y 1 población de presas utilizando un grupo de robots móviles llamados kilobots en el entorno de CoppeliaSim e importado en MATLAB.

La figura 45 refleja como se visualiza el modelo dinámico de poblaciones *2 depredadores y 1 presa* con condiciones ideales y continuas a través de métodos numéricos (a) y mediante un conjunto de robots móviles a través de condiciones no continuas (b). Comparando las dinámicas de poblaciones frente a frente se observan comportamientos similares en ambos ámbitos, donde evidentemente hay discrepancias por las interacciones de las poblaciones, debido a las condiciones no ideales y discontinuas que se encuentran en (b). Sin embargo, las similitudes que se encuentran entre (a) y (b) son que inicialmente las presas disminuyen a consecuencia de los depredadores, cuando eso sucede las poblaciones de depredadores aumentan, posteriormente sucede el efecto de competencia entre depredadores, el cual afecta negativamente a ambas poblaciones, ocasionando que desciendan ambas dinámicas y permitiendo que crezca la población de presa, lo cual genera que las poblaciones de depredadores crezcan volviendo a repetir la acción anterior. Es importante mencionar que los kilobots van a responder a la respuesta de sus vecinos, siempre y cuando se encuentren a una distancia cercana a ellos (menor o igual a 6 cm, condición de distancia propuesta en el código del algoritmo) en el momento que esa acción no se cumpla permanecerán sin cambiar de estado, únicamente sufrirán el descenso de sus puntos de vida hasta encontrarse con algún vecino.

Una vez explicado el caso anterior correspondiente a las interacciones que tienen los robots móviles con el algoritmo de control implementado referente al modelo matemático *2 poblaciones de depredadores y 1 población de presas*, al igual que los casos anteriores se procederá a anexar una tabla de valores, con el propósito de ver los diferentes resultados que se pueden generar cuando los kilobots siguen las reglas de este algoritmo. A continuación, se mostrará dicha tabla de valores:

Tiempo				
Pruebas	100 s	200 s	300 s	400 s
1	PV:19 PM:0 D1V:7 D1M:1 D2V:5 D2M:2	PV:13 PM:2 D1V:11 D1M:6 D2V:2 D2M:0	PV:21 PM:2 D1V:6 D1M:0 D2V:4 D2M:1	PV:15 PM:0 D1V:9 D1M:2 D2V:8 D2M:0
2	PV:15 PM:0 D1V:10 D1M:2 D2V:7 D2M:0	PV:16 PM:0 D1V:9 D1M:8 D2V:1 D2M:0	PV:21 PM:2 D1V:5 D1M:0 D2V:5 D2M:1	PV:16 PM:0 D1V:11 D1M:2 D2V:4 D2M:1
3	PV:15 PM:1 D1V:5 D1M:5 D2V:5 D2M:3	PV:16 PM:0 D1V:9 D1M:8 D2V:1 D2M:0	PV:14 PM:4 D1V:8 D1M:1 D2V:6 D2M:1	PV:13 PM:1 D1V:16 D1M:2 D2V:2 D2M:0
4	PV:14 PM:1 D1V:5 D1M:5 D2V:6 D2M:3	PV:18 PM:0 D1V:2 D1M:1 D2V:5 D2M:8	PV:25 PM:0 D1V:1 D1M:0 D2V:8 D2M:0	PV:18 PM:0 D1V:7 D1M:2 D2V:6 D2M:1
5	PV:15 PM:0 D1V:5 D1M:2 D2V:8 D2M:4	PV:23 PM:0 D1V:2 D1M:0 D2V:7 D2M:2	PV:23 PM:1 D1V:3 D1M:0 D2V:7 D2M:0	PV:0 PM:0 D1V:10 D1M:3 D2V:14 D2M:7

Cuadro 3: Pruebas del escenario de 2 poblaciones de depredadores y 1 población de presas (PV: Presas vivas, PM: Presas muertas, D1V: Depredadores 1 vivos, D1M: Depredadores 1 muertos, D2V: Depredadores 2 vivos, D2M: Depredadores 2 muertos).

La tabla anterior muestra una serie de resultados que se pueden generar a través de la implementación del algoritmo anterior en el grupo de robots móviles mostrado en la figura 42. Algunas características que pueden observar en este nuevo escenario mostrado son que: A medida que el número de presas disminuya se puede presentar el caso de que ambas poblaciones de depredadores aumenten por el efecto de depredación, pero también que una población sea más beneficiada que la otra, es decir, que una aumente más que la otra, otra característica que es presentada es que al haber dos poblaciones de depredadores éstas tienden a competir más entre ellas, ese efecto produce que las presas se reproduzcan

más rápido, por lo que hay casos donde hay presas abundantes y pocos depredadores, posterior a ellos las presas pueden seguir aumentando mientras una población de depredadores es más beneficiada que la otra.

6.3.1. Segunda prueba del escenario 2 depredadores y 1 presa

En esta ocasión se realizará la segunda prueba de este escenario, utilizando un número menor de agentes móviles al que se usó en la primer prueba, con el propósito de visualizar si hay cambios en la respuesta del algoritmo de control. A continuación se mostrará una figura la cual mostrará la configuración que tendrán los agentes móviles:

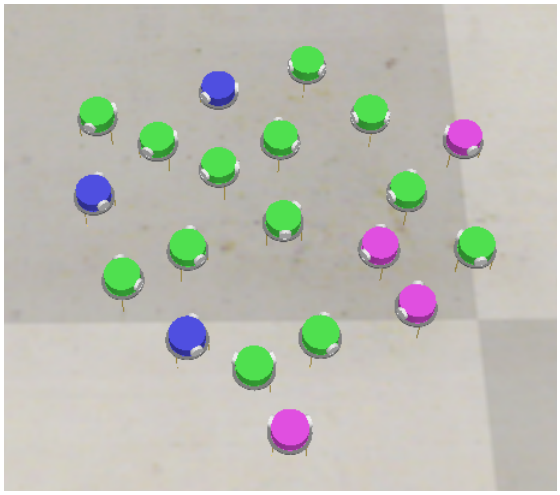


Figura 46: Segunda prueba del escenario de 2 poblaciones de depredadores y 1 población de presas.

Se observa en la figura anterior un número reducido de agentes móviles para representar este escenario a comparación de la primer prueba, donde las condiciones de inicio son las siguientes: $x(0) = 13$, $y(0) = 4$ y $z(0) = 3$, es decir, 13 presas (kilobots en color verde), 4 depredadores de la primer población (kilobots en color rosa) y 3 depredadores de la segunda población (kilobots en color azul) respectivamente. Adicionalmente se mostrarán las condiciones de inicio de forma gráfica con el motivo de observarlas de una forma más sencilla, sin necesidad de contabilizarlas de forma manual:

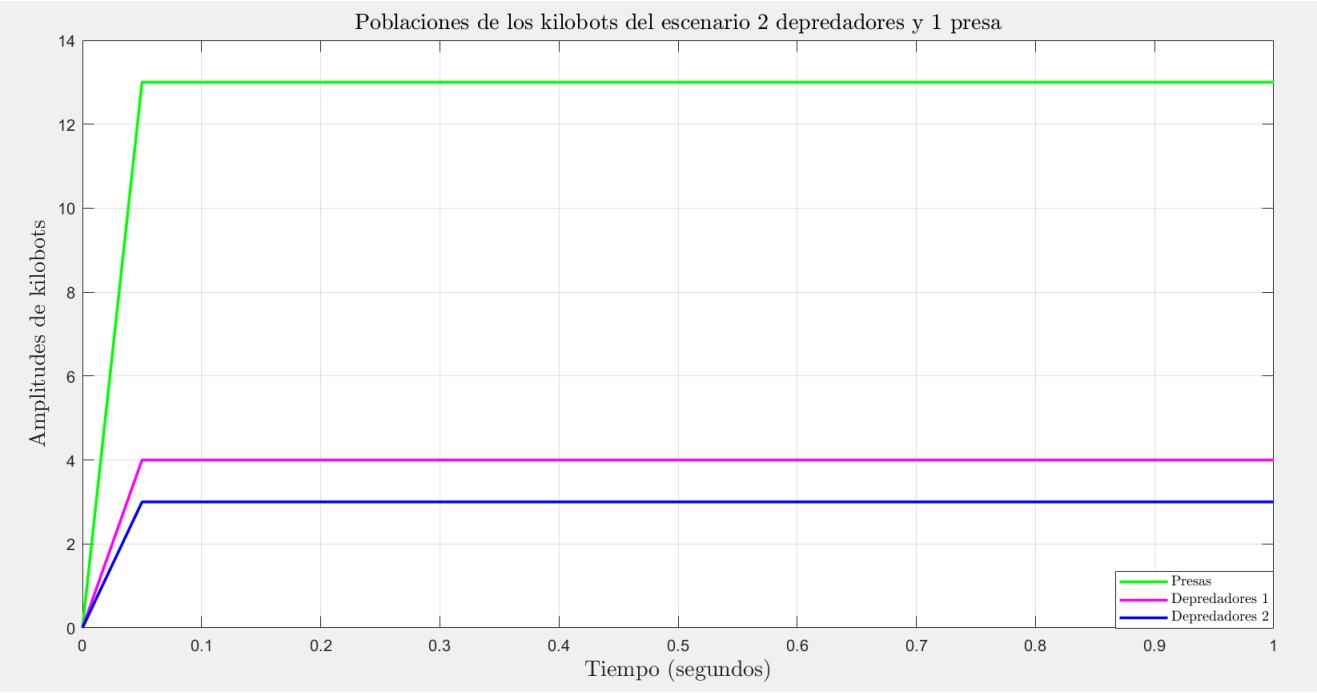
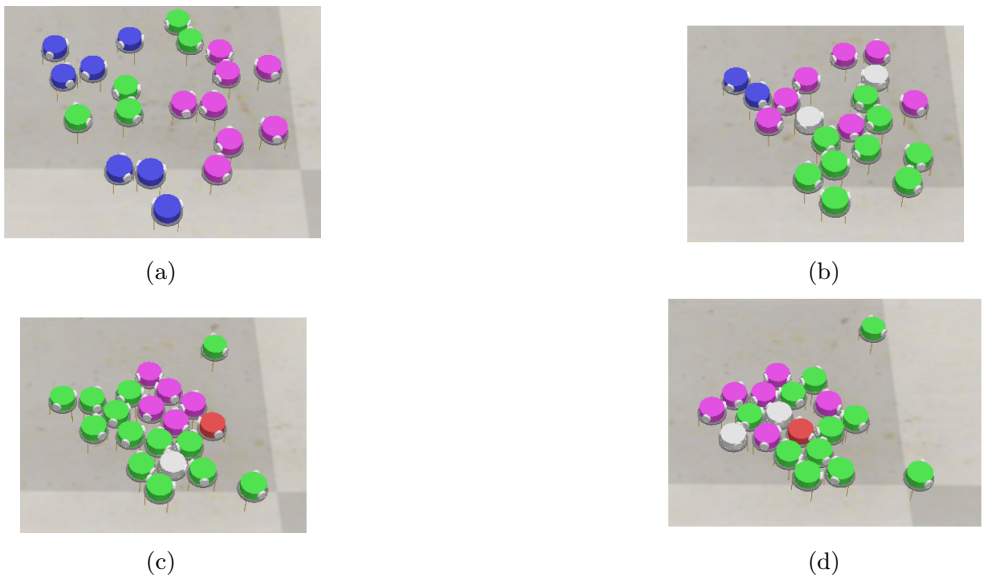


Figura 47: Segunda prueba del escenario de 2 poblaciones de depredadores y 1 población de presas (condiciones iniciales).

La figura anterior simplemente muestra que los kilobots no han tenido algún tipo de interacción durante el principio de la prueba (primer segundo de la simulación), sin embargo, a continuación se mostrará el conjunto de interacciones que los kilobots tuvieron en esta prueba. A continuación, se mostrará dicho conjunto:



(a) (b) (c) (d)

Figura 48: Conjunto de interacciones del algoritmo del escenario 2 depredadores y 1 presa representado por un pequeño grupo de robots móviles. Anexo 10.3 Algoritmo de 2 poblaciones de depredadores y 1 población de presas (CoppeliaSim).

La figura 48 muestra un conjunto de resultados que puede ser obtenido cuando el algoritmo correspondiente es implementado en un grupo de robots móviles con la configuración de la figura 46. Las interacciones que fueron obtenidas anteriormente fueron tomadas en instantes aproximados de 50 segundos cada una. En (a), capturada aproximadamente en 50 segundos, se observa que ambas poblaciones de depredadores han comenzado a crecer por la presencia de las presas, representados por los kilobots en color rosa (primer población depredadora) y los kilobots en color azul (segunda población depredadora). En la subfigura (b), capturada en el instante de 100 segundos aproximadamente, se observa que ahora ambas poblaciones depredadoras han comenzado a disminuir mientras que las presas han aumentado, eso se debe a que al no haber interacciones con presas los depredadores tienden a la transición de estado de individuo “ muerto”, el factor de competencia entre depredadores también influye para llevar a cabo esa transición. Después, en la subfigura (c), capturaba en el instante aproximado de 150 segundos, se observa que la segunda población depredadora ha desaparecido, por el hecho de no detectar a las presas a tiempo, mientras que la primer población depredadora tuvo una

ligera disminución. En la subfigura (d), capturada en el instante aproximado de 200 segundos, se observa que la población depredadora restante está volviendo a aumentar, de echo por el aumento de depredadores se ha observado algunos indicios de competencia (kilobots en color blanco).

Para observar lo que sucedió de forma panorámica se anexará una figura de comportamientos, desde el inicio del experimento ($t = 0$ segundos) hasta el instante final ($t = 200$ segundos), con la intención de observar las interacciones de los kilbotos con una configuración distinta a la primer prueba, a continuación, se mostrará dicha figura:

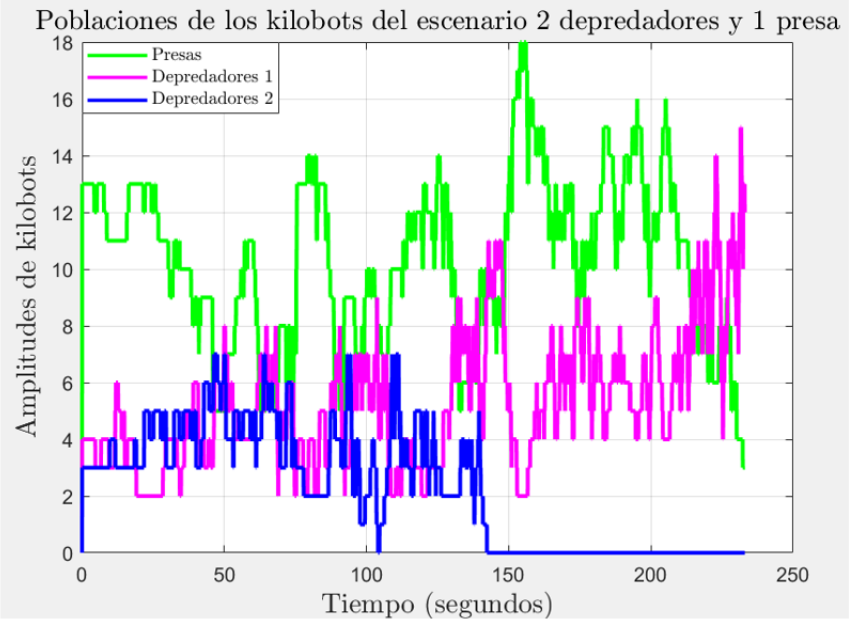


Figura 49: Gráfica de comportamientos del escenario de 2 poblaciones de depredadores y 1 población de presas (segunda prueba).

La figura anterior muestra el comportamientos de los robots referente a la figura 46, la explicación es prácticamente equivalente a la figura 48, durante el principio de la dinámica temporal las presas tienden al descenso, mientras que ambas poblaciones de depredadores presentan un aumento (50 segundos), cuando ambas poblaciones depredadoras descienden por falta de interacción con las presas ellas tienden a volver aumentar (instante previo a los 100 segundos), posteriormente los de-

predadores vuelven a encontrar a las presas provocando un aumento en sus poblaciones y el procedimiento se repite. Algo que hay que remarcar en esta prueba es que la segunda población depredadora (curva en color azul) llega a cero en el instante previo de 150 segundos, por el hecho de no interactuar con las presas a tiempo, ese comportamiento no se observa en la dinámica temporal del modelo matemático en condiciones ideales, sin embargo, es importante mencionar que es una situación que se puede dar cuando se trabajan con agentes móviles en condiciones no ideales y no continuas, también es una situación que suele presentarse en la realidad, cuando existen varias poblaciones depredadoras tienden a competir por el recurso de su sobrevivencia, en este caso dicho recurso son las presas.

Capítulo 7

7. Conclusiones generales

Las relaciones biológicas que pueden ser encontradas en la naturaleza a menudo son observadas por los seres humanos sin conocer la ciencia que éstas conllevan. En la realización de este trabajo de investigación una de las principales intenciones fue representar algunas relaciones biológicas de interés del tipo presa-depredador en forma matemática para poder visualizar su comportamiento dinámico de manera temporal (en función del tiempo), lo anterior fue de gran ayuda porque las formas matemáticas permiten ver las interacciones que se pueden dar por las especies y como afectan a las dinámicas de población.

Las relaciones biológicas presa-depredador de interés fueron desarrolladas en forma de control distribuido mediante algoritmos implementados en grupos de robots móviles de bajo costo llamados kilobots, con el fin de reproducir esos fenómenos biológicos en sistemas físicos, como lo son los grupos de robots móviles, y poder validar si los comportamientos generados por los kilobots son similares a las respuestas generadas por los modelos matemáticos de cada relación biológica. Después de la realización del presente trabajo de investigación se comprobó que la implementación en los robots móviles despliega comportamientos similares a la de los modelos matemáticos estudiados. Sin embargo, las respuestas de los kilobots no son completamente fieles a la de los modelos matemáticos, lo cual se debe a que con los robots móviles hay desplazamientos entre agentes, por lo que las interacciones con sus vecinos no son continuas a diferencia de las condiciones representadas por los modelos matemáticos.

A través del estudio que se generó con las implementaciones de los algoritmos en los robots móviles se pudo comprobar también algunas

características de las relaciones biológicas, las cuales fueron:

- La población de presas representa el sustento de la población de los depredadores, es decir, entre más presas existan la población de depredadores aumentará.

- A falta de presas los depredadores descenderán por falta de alimento.

- Con el fin de aumentar el realismo con los grupos de robots móviles se agregó el termino de competencia entre depredadores, ya que ésta surge cuando depredadores del mismo grupo buscan una presa en común.

De esta forma es como se pudo reproducir los modelos matemáticos en los grupos de robots móviles. En un futuro se buscará reproducir esos comportamientos en un ambiente experimental con mejoras en el código del algoritmo para mejorar la respuesta en relación a los modelos matemáticos y a las interacciones de los kilobots.

7.1. Trabajo futuro

En tiempos futuros sería recomendable actualizar el código de los algoritmos implementados utilizando comunicación local entre los kilobots para evadir obstáculos que se encuentren en un área, es decir, que presas y depredadores circulen con obstaculos para evasión y captura respectivamente.

Otra pauta que se podría realizar como trabajo futuro es obtener un rango de valores paramétricos de las interacciones de los kilobots en el código implementado, para mejorar la respuesta de los grupos de robots móviles a partir de diferentes tratamientos de datos.

8. Referencias

Ballerini M., Cabibbo N., Candelier R., Cavagna A., Cisbani E., Giardina I., Lecomte V., Orlandi A., Parisi G., Procaccini A., Viale M., y Zdravkovic V., (2007). *Interaction ruling animal collective behavior depends on topological rather than metric distance: Evidence from a field study.*

Bara A., y Dale S., (2009). *Dynamic modeling and stabilization of wheeled mobile robot.* CONTROL'09: Proceedings of the 5th WSEAS international conference on Dynamical systems and control. PP. 87-92.

Carrasco Gutiérrez L., (2021). *Implementación de modelos presa-depredador en enjambres de robots móviles.* [Tesis de maestría, Centro de Investigación Científica y de Educación Superior de Ensenada]. Repositorio CICESE.

Carrasco Gutiérrez L., Martínez Clark R., Pliego Jiménez J., y Cruz Hernández C., (2020). *Lotka-Volterra's Prey-Predator Model Interpretation In a Group of Low-cost Mobile Robots.*

Cont R., y Bouchaud J-P., (2000). *HERD BEHAVIOR AND AGGREGATE FLUCTUATIONS IN FINANCIAL MARKETS.* Macroeconomic Dynamics. Vol. 4. PP 170-196.

Dimidov C., Oriolo G., y Trianni V., (2016). *Random Walks in Swarm Robotics: An Experiment with Kilobots.* ANTS 2016, LNCS 9882. PP. 185-196. DOI: 10.1007/978-3-319-44427-7 16.

Helbing D., Farkas I., y Vicsek T., (2000). *Simulating Dynamical Features of Escape Panic.*

Hualong Q., Danyang Z., Jinzhu X., Yuze J., Hua Y., Tao L., Guodong L., y Changsheng Q., (2018). *Physiological mechanism of social insects behavior: nerve conduction, endocrine regulation, and genetic basis*.

Jadbabaie A., Lin J., y Morse S. A., (2003). *Coordination of Groups of Mobile Autonomous Agents Using Nearest Neighbor Rules*. IEEE TRANSACTIONS ON AUTOMATIC CONTROL. Vol. 48. PP. 988-1001.

Karlson P., Butenandt A., (1959). *PHEROMONES (ECTOHORMONES) IN INSECTS*. Annual Review of Entomology. Vol. 4. PP. 39-58.

Khalaf A. (2016). *Predator-Prey Relationships System*.

Krause J., y Ruxton G., (2002). *Living in Groups*. (Oxford Univ Press, Oxford). Primera edición.

Kumar S. (2019). *LOTKA-VOLTERRA MODEL*.

Maldonado C. E., y Gómez-Cruz N., (2014). *Synchronicity Among Biological and Computational Levels of an Organism: Quantum Biology and Complexity*. Procedia Computer Science, 36, (2014). PP. 177-184.
Nicholis G., y Prigogine I., (1994). *La estructura de lo complejo. En el camino hacia una nueva comprensión de las ciencias*. Madrid: Alianza, 1994.

Ordaz-Rivas E., Rodriguez Liñan A., y Torres-Treviño L., (2020). *Flock of Robots with Self-Cooperation for Prey-Predator Task*. Journal of Intelligent & Robotic Systems 101:39. https://doi.org/10.1007/s10846-

020-01283-0

Parrish J. K., y Edelstein-Keshet L., (1999). *Complexity, pattern, and evolutionary trade-offs in animal aggregation.* Science. Vol. 284. PP. 99-101.

Pelkonen J., (2018). *Exploration of Spatiotemporal Systems with Swarm of Kilobts in V-REP Simulation*. Master's Thesis. Aalto University. School of Electrical Engineering.

Pitcher T.J., (1993). *Behaviour of Teleost Fishes*, segunda edición, (Chapman and Hall, London).

Ponce Muñoz P., (2009). *ANÁLISIS DE LA TEORÍA DE SISTEMAS COMPLEJOS Y SU APLICACIÓN A SISTEMAS ORGANIZACIONES*. REVISMAR 1/2009 PP. 52-67.

Robinson G. E., Grozinger C. M., y Whitfield C. W., (2005). *SOCIOGENOMICS: SOCIAL LIFE IN MOLECULAR TERMS*. Nature Reviews Genetics. Vol. 6. PP. 257-268.

Roli A., (2015). *An introduction to complex systems science.*

Rohmer E., Singh S.P.N., y Freese M., (2013). *V-REP: A versatile and scalable robot simulation framework.*

Rubenstein M., Ahler C., Hoff N., Cabrera A., y Nagpal R., (2014). *Kilobot: A low cost robot with scalable operations designed for collective behaviors*. Robotics and Autonomous Systems. Vol. 62. PP. 966-975.

Sidón Ayala M. A., (2017). *COMPORTAMIENTOS COLECTIVOS*

CON ROBOTS KILOBOTS. [Tesis de licenciatura, Instituto Tecnológico de Ensenada]. Repositorio TecNM.

Von Arb R., (2013). *Predator Prey Models in Competitive Corporations. Honors Program Projects. 45..*
https : *//digitalcommons.olivet.edu/honr_proj/*45.

Wilson E.O., (1971). *The Insect Societies*. Cambridge, Massachusetts: Harvard Univ Press.

Zarzosa Gómez M. A., (2017). *DESCRIPCIÓN DE COMPORTAMIENTOS ANIMALES QUE SE UTILIZAN PARA ALGORITMOS DE OPTIMIZACIÓN E INTELIGENCIA ARTIFICIAL* [Tesis de licenciatura en biología, Universidad de La Laguna]. Repositorio Institucional de la Universidad de La Laguna.
http://riull.ull.es/xmlui/handle/915/6212.

Zhu B., Xie L., Han D., Meng X., y Teo R., (2017). *A survey on recent progress in control of swarm systems*. Science China Information Sciences. 60:070201. https://doi.org/10.1007/s11432-016-9088-2

9. Anexos

En el siguiente apartado se podrán encontrar los códigos empleados para la generación de cada dinámica de los sistemas utilizados en este trabajo de tesis.

9.1. Sistema de Lotka-Volterra

```
clear all;
clc;
close all;

[T,X] = ode45(@M1P1D, [0:0.001:100], [10 10]);

plot(T,X(:,1),'g','Linewidth',1.75); hold on;
plot(T,X(:,2),'m','Linewidth',1.75);
title('Modelo de Lotka-Volterra');
xlabel('Tiempo (T)'); ylabel('Amplitudes de las especies');
legend('Presas','Depredadores');

figure(2)
plot(X(:,1),X(:,2),'Linewidth',1.75);
title('Proyección del atractor'); xlabel('Presas');
ylabel('Depredadores');

function [xp] = M1P1D(t,x)
%Parametros constantes
r1=1.1;
a1=0.4;
r2=0.4;
a2=0.1;

%Almacenamiento de información
xp = zeros(2,1);

%Modelo matemático
 xp(1) = r1*x(1)-a1*x(1)*x(2);
 xp(2) = a2*x(1)*x(2)-r2*x(2);
end
```

9.2. Sistema de dos especies de depredadores y una especie de presas

```
clear all
clc
close all
```

```
[T,X] = ode45(@M2D1P, [0:0.001:50], [30 5 4]);

plot(T,X(:,1),'g','Linewidth',1.75); hold on;
plot(T,X(:,2),'m','Linewidth',1.75);
plot(T,X(:,3),'b','Linewidth',1.75);
title('Modelo de 2 depredadores y 1 presa');
xlabel('Tiempo (T)'); ylabel('Amplitudes de las especies');
legend('Presas', 'Depredadores #1', 'Depredadores #2');

 figure(2)

 plot3(X(:,1),X(:,2),X(:,3),'Linewidth',1.75);
 title('Proyección del atractor');
 xlabel('Presas'); ylabel('Depredadores 1');
 zlabel('Depredadores 2');

function [xp] = M2D1P(t,x)

    %Parametros constantes
    a = 1.5;
    b = 0.2;
    c = 0.1;
    d = 0.1;
    e = 0.715;
    f = 0.07;
    g = 0.5;

    %Almacenamiento de información
    xp = zeros(3,1);

    %Modelo matemático
    xp(1) = a*x(1) - b*x(1)*x(2) - c*x(1)*x(3);
    xp(2) = d*x(1)*x(2) - e*x(2);
    xp(3) = f*x(1)*x(3) - g*x(3);

end
```

9.3. Sistema de dos especies de presas y una especie de depredadores

```
clear all
clc
close all

[T,X] = ode45(@M2P1D, [0:0.001:50], [30 20 5]);

plot(T,X(:,1),'m','Linewidth',1.75); hold on;
plot(T,X(:,2),'g','Linewidth',1.75);
plot(T,X(:,3),'b','Linewidth',1.75);
title('Modelo de 2 presas y 1 depredador');
xlabel('Tiempo (T)'); ylabel('Amplitudes de las especies');
legend('Depredadores', 'Presas #1', 'Presas #2');

figure(2)
plot3(X(:,1),X(:,2),X(:,3),'Linewidth',1.75);
title('Proyección del atractor'); xlabel('Depredadores');
ylabel('Presas 1'); zlabel('Presas 2');

function [xp] = M2P1D(t,x)
%Parametros constantes
e1 = 0.008;
e2 = 0.009;
d = 0.7;
k1 = 0.9;
c1 = 0.009;
k2 = 1;
c2 = 0.01;

%Almacenamiento de información
xp = zeros(3,1);

%Modelo matemático
xp(1) = -d*x(1) + e1*x(1)*x(2) + e2*x(1)*x(3);
xp(2) = k1*x(2) - c1*x(1)*x(2);
xp(3) = k2*x(3) - c2*x(1)*x(3);
end
```

9.4. Sistema de Lotka-Volterra modificado

```
clear all;
clc;
close all;


[T,X] = ode45(@M1P1D, [0:0.001:100], [10 10]);

plot(T,X(:,1),'g','Linewidth',1.75); hold on; plot(T,X(:,2),'m','Linewidth',1.75);
```

```
title('Modelo de Lotka-Volterra modificado','Interpreter','Latex','Fontsize',14);
xlabel('Tiempo (T)','Interpreter','Latex','Fontsize',14);
ylabel('Amplitudes de las especies','Interpreter','Latex','Fontsize',15);
legend('Presas','Depredadores','Interpreter','Latex','Fontsize',10);

figure(2)
plot(X(:,1),X(:,2),'Linewidth',1.75);
title('Proyección del atractor','Interpreter','Latex','Fontsize',12);
xlabel('Presas','Interpreter','Latex','Fontsize',15);
ylabel('Depredadores','Interpreter','Latex','Fontsize',15);

function [xp] = M1P1D(t,x)
%Parametros constantes
r1=1.1;
a1=0.4;
r2=0.4;
a2=0.1;
k = 0.01;

%Almacenamiento de información
xp = zeros(2,1);

%Modelo matemático
 xp(1) = r1*x(1)-a1*x(1)*x(2);
 xp(2) = a2*x(1)*x(2)-r2*x(2)-k*(x(2)^2);
end
```

9.5. Sistema de dos especies de depredadores y una especie de presas modificado

```
clear all
clc
close all

[T,X] = ode45(@M2D1P, [0:0.001:50], [30 5 4]);

plot(T,X(:,1),'g','Linewidth',1.75); hold on; plot(T,X(:,2),'m','Linewidth',1.75);
plot(T,X(:,3),'b','Linewidth',1.75);
title('Modelo de 2 especies de depredadores y 1 especie de presas modificado',
'Interpreter','Latex','Fontsize',14);
xlabel('Tiempo (T)','Interpreter','Latex','Fontsize',14);
ylabel('Amplitudes de las especies','Interpreter','Latex','Fontsize',14);
legend('Presas', 'Depredador 1', 'Depredador 2','Interpreter','Latex');

 figure(2)

 plot3(X(:,1),X(:,2),X(:,3),'Linewidth',1.75);
 title('Proyección del atractor');
 xlabel('Presas'); ylabel('Depredadores 1'); zlabel('Depredadores 2');

function [xp] = M2D1P(t,x)

    %Parametros constantes
    a = 1.5;
    b = 0.2;
```

```
    c = 0.1;
    d = 0.1;
    e = 0.715;
    f = 0.07;
    g = 0.5;
    k1 = 0.01;
    k2 = 0.005;
    k3 = 0.01;
    k4 = 0.005;

    %Almacenamiento de información
    xp = zeros(3,1);

    %Modelo matemático
    xp(1) = a*x(1) - b*x(1)*x(2) - c*x(1)*x(3);
    xp(2) = d*x(1)*x(2) - e*x(2) - k1*(x(2)^2) - k2*x(2)*x(3);
    xp(3) = f*x(1)*x(3) - g*x(3) - k3*(x(3)^2) - k4*x(2)*x(3);

end
```

10. Sistema de dos especies de presas y una especie de depredadores modificado

```
close all

[T,X] = ode45(@M2P1D, [0:0.001:50], [30 20 5]);

plot(T,X(:,1),'m','Linewidth',1.75); hold on; plot(T,X(:,2),'g','Linewidth',1.75);
plot(T,X(:,3),'b','Linewidth',1.75);
title('Modelo de 2 especies de presas y 1 especie de depredadores',
'Interpreter','Latex','Fontsize',14);
xlabel('Tiempo (T)','Interpreter','Latex','Fontsize',14);
ylabel('Amplitudes de las especies','Interpreter','Latex','Fontsize',14);
legend('Depredadores', 'Presas 1', 'Presas 2','Interpreter','Latex');

figure(2)
plot3(X(:,1),X(:,2),X(:,3),'Linewidth',1.75);
title('Proyección del atractor'); xlabel('Depredadores');
ylabel('Presas 1'); zlabel('Presas 2');

function [xp] = M2P1D(t,x)
%Parametros constantes
e1 = 0.008;
e2 = 0.009;
d = 0.7;
k1 = 0.9;
c1 = 0.009;
k2 = 1.0;
c2 = 0.01;
T = 0.001;
%Almacenamiento de información
xp = zeros(3,1);
```

```
%Modelo matemático
xp(1) = -T*(x(1)^2)-d*x(1) + e1*x(1)*x(2) + e2*x(1)*x(3);
xp(2) = k1*x(2) - c1*x(1)*x(2);
xp(3) = k2*x(3) - c2*x(1)*x(3);
end
```

10.1. Algoritmo presa-depredador en robots móviles (CoppeliaSim)

```
--Variables globales de comportamiento
        prey = 1
        predator = 2

        behaviour = prey
        limite = 60
        dead = 0 --0 = alive, 2 = dead
        hpp = 20000
        hpd = 5000
        hp = hpp
        bandera = 1

        --Condiciones para la llegada de nuevos individuos
        revivepredator = 0
        revivezone = 25

        --Condiciones de la grafica de comportamientos
        run = 1
        x = 0 --presas
        y = 0 --depredadores
        z = 0 --poblacion de muertos

        sim.setInt32Signal("preycount", 0)
        --Posicionar las presas en 0 en la grafica (x = 0)
        sim.setInt32Signal("predatorcount", 0)
         --Posicionar los depredadores en 0 en la grafica (y = 0)
        sim.setInt32Signal("deadcount", 0)
         -- Posicionar a los muertos en 0 en la grafica (z = 0)


        ------------------------------------------
        -- Functions similar to C API
        ------------------------------------------

        function user_prgm()
        --//////////////////////////////////////////////////////////////////////////////
        --//user program code goes below.  this code needs to exit in a resonable amount of time
        --//so the special message controller can also run
        --//////////////////////////////////////////////////////////////////////////////

        --//////////////////////////////////////////////////////////////////////////////
        --//
        --//  In the example below, the robot moves and display its distance
        --//  to other robots with its color led.
```

```
--//
--////////////////////////////////////////////////////////////////////////////////

    --Inicializar la grafica

    if (run == 1) then
        if(behaviour == prey) then
        --Si el comportamiento del kilobot es presa entonces?
            x = sim.getInt32Signal("preycount")
            --Se almacena el valor de las presas en la variable x
            sim.setInt32Signal("preycount", x+1)
            -- Se posiciona en el valor de 1 la poblacion de las presas

        elseif(behaviour == predator) then
        --Si el comportamiento del kilobot es de un depredador entonces?
            y = sim.getInt32Signal("predatorcount")
            --Se almacena el valor de los depredadores en la variable y
            sim.setInt32Signal("predatorcount", y+1)
            -- Se posiciona en el valor de 1 la poblacion de los depredadores
        end
        run = 0
        --La bandera run recibe el valor 0 para salir del ciclo de especies iniciales
    end

    get_message()--El kilobot actual recibe informacion
    de vecinos cercanos (maximo a 7 cm de distancia)

   if(behaviour == prey) then --Si el comportamiento del kilobot es presa entonces?
        message_out(prey, 0, dead) -- El robot avisa a los vecinos cercanos
        que es una presa viva
        if(message_rx[3] == 0) then -- Si un vecino cercano esta vivo entonces?
            if(message_rx[6] == 1) then -- Si el robot recibe un mensaje nuevo entonces?
                if(message_rx[4] <= limite) then -- Si un vecino cercano esta a 6 cm o
                 menos de distancia entonces?
                    if(message_rx[1] == prey) then -- Si un vecino cercano
                     es una presa entonces?
                        if(dead == 2) then -- Si el kilobot actual esta muerto entonces?
                            reviveprey = reviveprey + 6.4 -- Se activa el contador
                             para revivir como presa
                            if(reviveprey >= revivezone) then -- Si el contador
                             supera la zona de revivir entonces?
                                behaviour = prey --El kilobot renace como presa
                                hp = hpp -- Sus puntos de vida estan completos
                                dead = 0 --La presa adquiere estado de vivo
                                reviveprey = 0 -- Al ser una nueva presa
                                su contador debe estar reiniciado
                                bandera = 1 --La bandera tiene el valor de 1
                                debido a que la presa esta viva
                                message_out(prey,0,dead)

                                x = sim.getInt32Signal("preycount") --Se obtiene
                                el valor de las presas
                                sim.setInt32Signal("preycount", x+1) -- Al valor de
                                las presas se le suma 1

                                z = sim.getInt32Signal("deadcount") -- Se obtiene
```

```
                        el valor de la poblacion de especies muertas
                        sim.setInt32Signal("deadcount", z-1) -- Se disminuye
                        1 unidad a la poblacion de muertos
                    end
                else -- Si la presa actual tiene de vecino a una presa, entonces?
                    -- No pasa nada
                end
            elseif(message_rx[1] == predator) then --Si un vecino cercano
             es un depredador entonces?
                if(dead == 2) then --Si el kilobot actual esta muerto entonces?
                    revivepredator = revivepredator + 0.85 -- Se activa
                    el contador para revivir como depredador
                    if(revivepredator >= revivezone) then -- Si el
                    contador supera la zona de revivir entonces?
                        behaviour = predator --El kilobot renace como depredador
                        hp = hpd --Sus puntos de vida estan completos
                        dead = 0 --El depredador aquiere el estado de vivo
                        revivepredator = 0 -- Al ser un nuevo
                        depredador su contador debe estar reiniciado
                        bandera = 1 --La bandera tiene el valor de
                        1 debido a que el depredador esta vivo
                        message_out(predator,0,dead)

                        y = sim.getInt32Signal("predatorcount") --Se
                        obtiene el valor de los depredadores
                        sim.setInt32Signal("predatorcount", y+1) -- Al
                        valor de los depredadores se le suma 1

                        z = sim.getInt32Signal("deadcount") --Se obtiene
                         el valor de la poblacion de especies muertas
                        sim.setInt32Signal("deadcount", z-1) -- Se disminuye
                        1 unidad a la poblacion de muertos
                    end
                else -- Si la presa actual tiene
                de vecino a un depredador entonces?
                    hp = hp - 2700 -- La presa esta siendo devorada
                end
            end -- Si un vecino cercano es presa/depredador entonces?
        end -- Si un vecino cercano esta a 6 cm o menos de distancia entonces?
    end -- Si el robot recibe un mensaje nuevo entonces?
end -- Si un vecino cercano esta vivo entonces?

if(hp>0) then -- Si la presa tiene puntos de vida arriba de cero entonces?
    hp = hp - 1 -- Mientras avanza el tiempo iran disminuyendo
else
    dead = 2 -- En caso contrario la presa adquiere el estado de muerta
    if(bandera == 1) then --Bandera logica para contabilizar muertes
        bandera = 0 -- La bandera cambia de valor
        para salir del ciclo de las muertes

        z = sim.getInt32Signal("deadcount") --Se obtiene el valor
        de la poblacion de especies muertas
        sim.setInt32Signal("deadcount", z+1) -- Al morir la presa se
        aumenta 1 unidad a la poblacion de muertos

        x = sim.getInt32Signal("preycount") --Se obtiene el valor de las presas
```

```
            sim.setInt32Signal("preycount", x-1) -- Al morir la presa
            se disminuye 1 unidad la poblacion de presas
        end
    end

if(dead == 0) then --Si la presa esta viva entonces?
    set_color(0,3,0) -- color verde
    set_motor(math.random(0,40),40-math.random(0,40)) -- Caminata aleatoria
else -- Si la presa esta muerta entonces?
    set_color(3,0,0) -- Color rojo
    set_motor(0,0) -- Sin movimiento
end


elseif(behaviour == predator) then --Si el comportamiento del kilobot
es depredador entonces?
message_out(predator, 0, dead) --El robot avisa a los vecinos cercanos
que es un depredador vivo
if(message_rx[3] == 0) then --Si un vecino cercano esta vivo entonces?
    if(message_rx[6] == 1) then --Si el robot recibe un mensaje nuevo entonces?
        if(message_rx[4] <= limite) then --Si un vecino cercano
        esta a 6 cm o menos de distancia entonces?
            if(message_rx[1] == prey) then --Si un vecino cercano
             es una presa entonces?
                if(dead == 2) then -- Si el depredador actual esta muerto entonces?
                    reviveprey = reviveprey + 6.4 --Se activa el contador
                    para revivir como presa
                    if(reviveprey >= revivezone) then -- Si el contador supera
                    el valor de la zona de revivir entonces?
                        behaviour = prey --El robot renace como presa
                        hp = hpp --Sus puntos de vida estan completos
                        dead = 0 --Adquiere el estado de vivo
                        reviveprey = 0 --Su contador se reinicia por ser nueva presa
                        bandera = 1 -- La bandera tiene el valor de 1
                        porque la presa esta viva
                        message_out(prey,0,dead)

                        x = sim.getInt32Signal("preycount") --Se obtiene el valor
                        de las presas
                        sim.setInt32Signal("preycount", x+1) -- Al revivir una
                         nueva presa se aumenta en 1 unidad su valor

                        z = sim.getInt32Signal("deadcount") --Se obtiene el valor
                        de la poblacion de muertos
                        sim.setInt32Signal("deadcount", z-1) -- Al revivir
                        una nueva presa la poblacion de muertos disminuye en 1 unidad
                    end
                else --Si el depredador actual esta vivo entonces?
                    hp = hp + 100 -- Devora a la presa, por lo que aumenta sus hp
                end
            elseif(message_rx[1] == predator) then --Si un vecino cercano
            es un depredador entonces?
                if(dead == 2) then -- Si el depredador actual esta muerto entonces?
                    revivepredator = revivepredator + 0.85 --Se activa el contador
                    para revivir como un depredador
                    if(revivepredator >= revivezone) then -- En el momento
```

```
                    en el que el contador supera la zona de revivir entonces?
                        behaviour = predator -- El robot renace como depredador
                        hp = hpd --Sus puntos de vida estan completos
                        dead = 0 --Adquiere el estado de vivo
                        revivepredator = 0 --Su contador se reinicia
                        por ser un nuevo depredador
                        bandera = 1 --La bandera tiene el valor de 1
                        debido a que el depredador esta vivo
                        message_out(predator,0,dead)

                        y = sim.getInt32Signal("predatorcount") --Se obtiene
                         el valor de los depredadores
                        sim.setInt32Signal("predatorcount", y+1) -- Al revivir
                         un depredador se aumenta en 1 unidad su valor

                        z = sim.getInt32Signal("deadcount") -- Se obtiene
                        el valor de la poblacion de los muertos
                        sim.setInt32Signal("deadcount", z-1) -- Al revivir
                        un depredador se disminuye la poblacion de los
                         muertos en 1 unidad
                    end
                else --Si el depredador actual esta vivo y se encuentra a un
                depredador cercano entonces?
                    hp = hp - 250 -- Competencia entre depredadores,
                    lo cual afecta de forma negativa los hp del depredador actual
                end
            end -- Si un vecino cercano es una presa/depredador entonces?
        end --Si un vecino cercano esta a 6 cm o menos de distancia entonces?
    end --Si el robot recibe un mensaje nuevo entonces?
end --Si un vecino cercano esta vivo entonces

if (hp>0) then -- Si el depredador tiene puntos de vida arriba de cero entonces?
    hp = hp - 1 -- Mientras avanza el tiempo iran disminuyendo
else
    dead = 2 -- Si sus hp no son mayores a cero entonces
    el depredador adquiere el estado de muerto
    if(bandera == 1) then -- Bandera logica para contabilizar muertes
        bandera = 0 -- La bandera logica cambia de valor para salir del ciclo

        z = sim.getInt32Signal("deadcount") --Cuando el depredador muera
         por que sus hp = 0 entonces?
        sim.setInt32Signal("deadcount", z+1) -- Se aumenta en 1 unidad
        la poblacion de especies muertas

        y = sim.getInt32Signal("predatorcount") --Se obtiene el valor
         de los depredadores
        sim.setInt32Signal("predatorcount", y-1) -- Se disminuye el valor
        de los depredadores en 1 unidad porque sus hp = 0
    end
end

if(dead == 0) then --Si el depredador esta vivo entonces?
    set_color(3,0,3) -- color morado
    set_motor(math.random(0,40),40-math.random(0,40)) -- Caminata aleatoria
else -- Si el depredador esta muerto entonces?
    set_color(3,3,3) -- Color blanco
```

```
            set_motor(0,0) -- Sin movimiento
        end

    end -- FIN DEL ESCENARIO 1 DEPREDADOR Y 1 PRESA

    --////////////////////////////////////////////////////////////////////////////////
    --//END OF USER CODE
    --////////////////////////////////////////////////////////////////////////////////
    end
```

10.2. Algoritmo de 2 poblaciones de presas y 1 población de depredadores (CoppeliaSim)

```
--Variables globales de comportamiento
    prey_1 = 1
    prey_2 = 2
    predator = 3

    behaviour = prey_2
    limite = 60
    dead = 0 --0 = alive, 2 = dead
    hpp1 = 20000
    hpp2 = 20000
    hpd = 5000
    hp = hpp2

    --Condiciones para la llegada de nuevos individuos
    reviveprey_1 = 0
    reviveprey_2 = 0
    revivepredator = 0
    revivezone = 25
    rebirth = 1 --Bandera para contar el ciclo despues de la muerte

    --Condiciones de la grafica de comportamientos
    run = 1
    w = 0 --Presas_1
    x = 0 --presas_2
    y = 0 --depredadores
    z = 0 --muertos

    sim.setInt32Signal("preycount_1", 0) --Posicionar las presas_1 en 0 en la grafica
    sim.setInt32Signal("preycount_2", 0) --Posicionar las presas_2 en 0 en la grafica
    sim.setInt32Signal("predatorcount", 0) -- Posicionar los depredadores en 0 en la grafica
    sim.setInt32Signal("deadcount", 0) -- Posicionar a los muertos en 0 en la grafica


    ------------------------------------------------
    -- Functions similar to C API
    ------------------------------------------------

    function user_prgm()
    --////////////////////////////////////////////////////////////////////////////////
    --//user program code goes below.  this code needs to exit in a resonable amount of time
    --//so the special message controller can also run
```

```
        --////////////////////////////////////////////////////////////////////////////////

        --////////////////////////////////////////////////////////////////////////////////
        --//
        --//  In the example below, the robot moves and display its distance
        --//  to other robots with its color led.
        --//
        --////////////////////////////////////////////////////////////////////////////////

        if (run == 1) then --Bandera para contabilizar los robots iniciales
            if (behaviour == prey_1) then -- Si el robot es una presa_1, entonces?
                w = sim.getInt32Signal("preycount_1") -- En w se guarda la cantidad de presas_1
                sim.setInt32Signal("preycount_1", w+1) --Se aumentan las presas_1 en una unidad

            elseif (behaviour == prey_2) then --Si el robot es presa_2, entonces?
                x = sim.getInt32Signal("preycount_2") -- En x se guarda la cantidad de presa_2
                sim.setInt32Signal("preycount_2", x+1) --Se aumentan las presas_2 en una unidad

            elseif(behaviour == predator) then -- Si el robot es depredador, entonces?
                y = sim.getInt32Signal("predatorcount") --En y se guarda la cantidad de depredadores
                sim.setInt32Signal("predatorcount", y+1) --Se aumentan los depredadores en una unidad
            end
            run = 0 -- La bandera cambia de valor para evitar un ciclo infinito
        end         -- en la contabilidad de las condiciones iniciales

------------------------------------------------------------
        get_message() --Obtenemos los mensajes de los kilobots cercanos

        if(behaviour == prey_1) then
            message_out(prey_1,0,dead) -- Envia mensaje a los demas kilobots
            if(message_rx[3] == 0) then -- Si un vecino cercano esta vivo, entonces?
            if (message_rx[6] == 1) then -- Si el robot recibe un nuevo mensaje, entonces?
                if (message_rx[4] < limite) then --Si robot esta cercas (distancia menor a 6 cm),
                entonces?
                    if(message_rx[1] == prey_1) then --Si robot cercano es presa_1, entonces?
                        if(dead == 2) then -- En caso de que la presa_1 este muerta, entonces?
                            reviveprey_1 = reviveprey_1 + 3.5
                            --Condiciones para revivir presa_1
                            if(reviveprey_1 >= revivezone) then
                                behaviour = prey_1 --Comportamiento de nueva presa_1 para el kilobot
                                hp = hpp1 -- Puntos de vida al maximo
                                dead = 0 --Adquiere el estado de vivo
                                rebirth = 1 --renacimiento
                                message_out(prey_1,0,dead)

                                w = sim.getInt32Signal("preycount_1") --Se obtiene la señal
                                 de las presas_1
                                sim.setInt32Signal("preycount_1", w+1) --Se agrega una nueva
                                 presa_1 a la grafica

                                z = sim.getInt32Signal("deadcount") --Se obtiene la señal
                                de los decesos
                                sim.setInt32Signal("deadcount",z-1) -- Debido a que revivio
                                una nueva presa_1 se quita 1
                                 unidad a los decesos
                            end
```

```
    else
        --Si la presa_1 esta viva y tiene una presa_1 cerca no hay consecuencia
    end
elseif(message_rx[1] == prey_2) then --Si robot cercano es presa_2, entonces?
    if(dead == 2) then -- En caso de que la presa_1 este muerta, entonces?
        reviveprey_2 = reviveprey_2 + 2.3
        if(reviveprey_2 >= revivezone) then
        --Condiciones para revivir como prey_2
            behaviour = prey_2 -- Comportamiento de nueva presa_2 para el kilobot
            hp = hpp2 -- Puntos de vida al maximo
            dead = 0 -- Adquiere el estado de vivo
            rebirth = 1 -- Renacimiento
            message_out(prey_2,0,dead)

            x = sim.getInt32Signal("preycount_2") --Se obtiene
            la señal de las presas_2
            sim.setInt32Signal("preycount_2", x+1) --Se agrega
             una nueva presa_2 a la grafica

            z = sim.getInt32Signal("deadcount") --Se obtiene la señal
             de los decesos
            sim.setInt32Signal("deadcount", z-1) -- Debido a que revivio una
            nueva presa_2 se quita una unidad a los decesos

        end
    else
        --Si la presa_1 esta viva y tiene una presa_2 cerca no hay consecuencia
    end
elseif(message_rx[1] == predator) then --Si robot cercano
es depredador, entonces?
    if(dead == 2) then -- En caso de que la presa_1 este muerta, entonces?
        revivepredator = revivepredator + 3.6
        if(revivepredator >= revivezone) then
            --Condiciones  para revivir como depredador
            behaviour = predator -- Comportamiento de nuevo depredador
            para el kilobot
            hp = hpd -- Adquiere los puntos de vida de un depredador
            dead = 0 -- Adquiere el estado de vivo
            rebirth = 1 -- Renacimiento
            message_out(predator,0,dead)

            y = sim.getInt32Signal("predatorcount") --Se obtiene la señal
             de los depredadores
            sim.setInt32Signal("predatorcount", y+1) --Se agrega un
            nuevo depredador a la grafica

            z = sim.getInt32Signal("deadcount") --Se obtiene la señal
             de los decesos
            sim.setInt32Signal("deadcount", z-1) -- Debido a que revivio un
             nuevo depredador se quita una unidad a los decesos
        end
    else
        --Si la presa_1 esta viva y se encuentra a un depredador debe ser devorada
        hp = hp - 2500 -- Proceso de devoracion
    end
end --Si robot cercano es prey_1/prey_2/predator
```

```
        end --Si robot esta cercas (distancia menor a 6 cm), entonces?
    end -- Si el robot recibe un nuevo mensaje, entonces?
    end -- Si un vecino cercano esta vivo, entonces?

if(hp>0) then -- Si los puntos de vida del depredador son mayores a 0, entonces?
    hp = hp-1 -- Se iran restando de uno por uno
else
    dead = 2 -- Cuando no sean mayores a 0 la presa_1 adquirira el estado de muerto
    if(rebirth == 1) then -- Bandera logica para contabilizar muertes
        rebirth = 0 -- La bandera cambia de valor para salir del ciclo inifinito

        z = sim.getInt32Signal("deadcount") -- Se obtiene la señal de los decesos
        sim.setInt32Signal("deadcount", z+1) -- Debido a que murio la presa_1 se aumenta
                                             -- en una unidad la poblacion de decesos

        w = sim.getInt32Signal("preycount_1") -- Se obtiene el valor de las presas_1
        sim.setInt32Signal("preycount_1",w-1) -- Se disminuye en una unidad a las presas_1
    end
end

if (dead==2) then -- Si la presa_1 esta muerta, entonces?
    set_motor(0,0) -- No tiene movimiento
    set_color(3,0,0) -- Color rojo
else                    --presa_1 viva
    set_motor(math.random(0,40),40-math.random(0,40)) -- Caminata aleatoria
    set_color(0,3,0) -- Color verde
end -- Fin del comportamiento de la presa_1


elseif(behaviour == prey_2) then --Comportamiento de presa_2
    message_out(prey_2,0,dead) -- Envia un mensaje a los demas kilobots
    if(message_rx[3] == 0) then -- Si un kilobot vecino esta vivo, entonces?
        if(message_rx[6]==1) then -- Si el kilobot actual recibe
        un nuevo mensaje, entonces?
            if(message_rx[4]<limite) then -- Si un kilobot esta a 6 cm
            o menos de distancia, entonces?
                if(message_rx[1] == prey_1) then--Si un robot vecino
                es una presa_1, entonces?
                    if(dead==2) then -- Si el kilobot actual (presa_2)
                     esta muerto, entonces?
                        reviveprey_1 = reviveprey_1 + 3.5
                        if(reviveprey_1>=revivezone) then
                            --Condiciones para revivir como presa_1
                            behaviour=prey_1 -- Adquiere el comportamiento de presa_1
                            hp=hpp1 -- Puntos al maximo
                            dead=0 -- Adquiere el estado de vivo
                            rebirth=1 -- Renacimiento
                            message_out(prey_1,0,dead)

                            w=sim.getInt32Signal("preycount_1") -- Se obtiene la señal
                             de presas_1
                            sim.setInt32Signal("preycount_1", w+1) -- Se aumenta
                             en una unidad a las presas_1

                            z = sim.getInt32Signal("deadcount") --Se obtiene la señal
                             de los decesos
```

```
            sim.setInt32Signal("deadcount",z-1) -- Se disminuye
             en una unidad los decesos
        end
    else
        --Si la presa_2 esta viva y se encuentra a una presa_1 viva
        no hay consecuencia
    end
elseif(message_rx[1] == prey_2) then --Si un robot vecino
es una presa_2, entonces?
    if(dead==2) then -- Si el kilobot actual (presa_2)
    esta muerto, entonces?
        reviveprey_2 = reviveprey_2 + 2.3
        if(reviveprey_2>=revivezone) then
            --Condiciones para revivir como presa_2
            behaviour=prey_2 -- Adquiere el comportamiento de presa_2
            hp=hpp2 -- Puntos al maximo
            dead=0 -- Adquiere el estado de vivo
            rebirth=1 -- Renacimiento
            message_out(prey_2,0,dead)

            x=sim.getInt32Signal("preycount_2") -- Se obtiene la señal
             de presas_2
            sim.setInt32Signal("preycount_2", x+1) -- Se aumenta en
             una unidad a las presas_2

            z = sim.getInt32Signal("deadcount") --Se obtiene la señal
             de los decesos
            sim.setInt32Signal("deadcount",z-1) -- Se disminuye
             en una unidad los decesos
        end
    else
        --Si la presa_2 esta viva y se encuentra a una presa_2 viva
        no hay consecuencia
    end
elseif(message_rx[1] == predator) then --Si un robot vecino
 es una depredador, entonces?
        if(dead==2) then -- Si el kilobot actual (presa_2)
         esta muerto, entonces?
            revivepredator=revivepredator + 3.6
            --Condiciones para revivir como depredador
            if(revivepredator >= revivezone) then
                behaviour = predator -- Adquiere el estado de depredador
                hp=hpd -- Puntos al maximo del depredador
                dead=0 -- Adquiere el estado de vivo
                rebirth=1 -- Renacimiento
                message_out(predator,0,dead)

                y = sim.getInt32Signal("predatorcount") -- Se obtiene la
                señal de los depredadores
                sim.setInt32Signal("predatorcount",y+1) -- Se aumenta en
                una unidad la poblacion de depredadores

                z = sim.getInt32Signal("deadcount") -- Se obtiene la señal
                 de los decesos
                sim.setInt32Signal("deadcount",z-1) -- Se disminuye
                 en una unidad su poblacion
```

```
                        end
                    else
                        --Si la presa_2 esta viva y se encuentra a un
                         depredador, debe ser devorada
                        hp = hp - 2500 -- Proceso de devoracion
                    end
                end -- Fin vecino /presa_1/presa_2/depredador
            end -- Si un kilobot esta a 6 cm o menos de distancia, entonces?
        end -- Si el kilobot actual recibe un nuevo mensaje, entonces?
        end -- Si un kilobot vecino esta vivo, entonces?
    if(hp > 0) then -- Si la presa_2 tiene puntos arriba de 0, entonces?
        hp = hp-1 -- Iran disminuyendo de uno en uno
    else
        dead = 2 -- muerto
        if(rebirth == 1) then -- Bandera logica para contabilizar a los muertos
            rebirth = 0 -- La bandera cambia de valor para salir del ciclo infinito

            z = sim.getInt32Signal("deadcount") -- Se obtiene el valor de los decesos
            sim.setInt32Signal("deadcount",z+1) -- Aumenta en una unidad los decesos

            x = sim.getInt32Signal("preycount_2") -- Se obtiene el valor de las presas_2
            sim.setInt32Signal("preycount_2", x-1) -- Se disminuye en una unidad su valor
        end
    end

    if(dead == 2) then
        set_motor(0,0) --Sin movimiento
        set_color(3,3,0) --Amarillo
    else
        set_motor(math.random(0,40),40-math.random(0,40)) -- Caminata aleatoria
        set_color(0,0,3) --Azul
    end -- Fin comportamiento de presa_2

elseif(behaviour == predator) then --Comportamiento de depredador
    message_out(predator,0,dead)
    if(message_rx[3] == 0) then
        if(message_rx[6] == 1) then --Nuevo mensaje
            if(message_rx[4] < limite) then --Robot cercano
                if(message_rx[1] == prey_1) then -- Si robot cercano es prey_1
                    if(dead == 2) then --si se encuentra a una presa_1 muerta
                        reviveprey_1 = reviveprey_1 + 3.5

                        if(reviveprey_1 >= revivezone) then
                        --Condiciones para revivir como presa_1
                            behaviour = prey_1
                            hp = hpp1
                            dead = 0
                            rebirth = 1
                            message_out(prey_1,0,dead)

                            w = sim.getInt32Signal("preycount_1")
                            sim.setInt32Signal("preycount_1", w+1)

                            z = sim.getInt32Signal("deadcount")
                            sim.setInt32Signal("deadcount", z-1)
                        end
```

```
                else
                    hp = hp + 100 --devora a la presa_1
                end
            elseif(message_rx[1] == prey_2) then
                if(dead == 2) then --si se encuentra a una presa_2 muerta
                    reviveprey_2 = reviveprey_2 + 2.3

                    if(reviveprey_2 >= revivezone) then
                    --Condiciones para revivir como presa_2
                        behaviour = prey_2
                        hp = hpp2
                        dead = 0
                        rebirth = 1
                        message_out(prey_2,0,dead)

                        x = sim.getInt32Signal("preycount_2")
                        sim.setInt32Signal("preycount_2", x+1)

                        z = sim.getInt32Signal("deadcount")
                        sim.setInt32Signal("deadcount", z-1)
                    end
                else
                    hp = hp + 100 --devora a la presa_2
                end
            elseif(message_rx[1] == predator) then
                 if(dead == 2) then --si se encuentra a un depredador muerto
                    revivepredator = revivepredator + 3.6

                    if(revivepredator >= revivezone) then
                    --Condiciones para revivir como depredador
                        behaviour = predator
                        hp = hpd
                        dead = 0
                        rebirth = 1
                        message_out(predator,0,dead)

                        y = sim.getInt32Signal("predatorcount")
                        sim.setInt32Signal("predatorcount", y+1)

                        z = sim.getInt32Signal("deadcount")
                        sim.setInt32Signal("deadcount", z-1)
                    end
                else
                    hp = hp - 250 -- competencia entre depredadores
                end
            end
        end -- Robot cercano
    end --Nuevo mensaje
    end
if(hp>0) then --Si el depredador esta vivo
    hp = hp-1
else
    dead=2
    if(rebirth == 1) then
        --Condiciones cuando el depredador esta muerto
        rebirth = 0
```

```
                z = sim.getInt32Signal("deadcount")
                sim.setInt32Signal("deadcount",z+1)

                y = sim.getInt32Signal("predatorcount")
                sim.setInt32Signal("predatorcount",y-1)
            end
        end

        if(dead == 2) then -- Si el deredador esta muerto, entonces?
            set_motor(0,0)
            set_color(3,3,3) -- blanco
        else -- Si el depredador esta vivo, entonces?
            set_motor(math.random(0,40), 40-math.random(0,40))
             set_color(3,0,3) --morado
        end
    end --Fin del comportamiento 2 presas y 1 depredador

    --////////////////////////////////////////////////////////////////////////////////
    --//END OF USER CODE
    --////////////////////////////////////////////////////////////////////////////////
    end
```

10.3. Algoritmo de 2 poblaciones de depredadores y 1 población de presas (CoppeliaSim)

```
--Variables globales de comportamiento
prey_1 = 1
predator_1 = 2
predator_2 = 3

  behaviour = prey_1
        limite = 60
        dead = 0 --0 = alive, 2 = dead
        hpp = 70000
        hpd1 = 5000
        hpd2 = 5000
        hp = hpp
        bandera = 1

        --Condiciones para la llegada de nuevos individuos
        reviveprey_1 = 0
        revivepredator_1 = 0
        revivepredator_2 = 0
        revivezone = 25

        --Condiciones de la grafica de comportamientos
        run = 1
        w = 0 --Presas_1
        x = 0 --depredadores_1
        y = 0 --depredadores_2
        z = 0 --poblacion de muertos

        sim.setInt32Signal("preycount_1", 0) --Posicionar las presas_1 en 0 en la grafica
        sim.setInt32Signal("predatorcount_1", 0) --Posicionar los depredadores_1 en 0 en la grafica
```

```
sim.setInt32Signal("predatorcount_2", 0) -- Posicionar los depredadores_2 en 0 en la grafica
sim.setInt32Signal("deadcount", 0) -- Posicionar a los muertos en 0 en la grafica


-------------------------------------
-- Functions similar to C API
-------------------------------------

function user_prgm()
--//////////////////////////////////////////////////////////////////////////////////////
--//user program code goes below.  this code needs to exit in a resonable amount of time
--//so the special message controller can also run
--//////////////////////////////////////////////////////////////////////////////////////

--//////////////////////////////////////////////////////////////////////////////////////
--//
--//  In the example below, the robot moves and display its distance
--//  to other robots with its color led.
--//
--//////////////////////////////////////////////////////////////////////////////////////

    --Inicializar la grafica

    if(run == 1) then -- Bandera para contabilizar a las condiciones iniciales
        if(behaviour == prey_1) then -- Si el comportamiento del robot es de
        presa, entonces?
            w = sim.getInt32Signal("preycount_1") -- Se obtiene la señal de las presas
            sim.setInt32Signal("preycount_1", w+1) -- Se aumenta en una unidad a las presas


        elseif(behaviour == predator_1) then -- Si el comportamiento del robot
         es de depredador 1, entonces?
            x = sim.getInt32Signal("predatorcount_1") -- Se obtiene la señal
             de los depredadores 1
            sim.setInt32Signal("predatorcount_1", x+1) -- Se aumenta en una unidad
            a los depredadores 1


        elseif(behaviour == predator_2) then -- Si el comportamiento del robot es de
         depredador 2, entonces?
            y = sim.getInt32Signal("predatorcount_2") -- Se obtiene la señal
             de los depredadores 2
            sim.setInt32Signal("predatorcount_2", y+1) -- Se aumenta en una unidad
             a los depredadores 2

        end
        run = 0 -- La bandera cambia de valor para salir del ciclo infinito
    end

    get_message() -- Se reciben mensajes de kilobots cercanos

    if(behaviour == prey_1) then -- Si el kilobot presente es presa_1, entonces?
        message_out(prey_1, 0, dead) --manda mensaje a los demas kilobots diciendo
        que es presa_1 y esta viva
        if(message_rx[6]==1) then -- Si el kilobot actual recibe un mensaje
        de un kilobot vecino, entonces?
```

```
if(message_rx[3] == 0) then -- Si un kilobot vecino esta vivo, entonces?
    if(message_rx[4] <= limite) then -- Si un kilobot vecino esta cerca
    (6 cm o menos), entonces?
        if(message_rx[1] == prey_1) then -- Si un kilobot vecino es
        una presa_1, entonces?
            if(dead == 2) then --Si el kilbot presente esta
            muerto, y tiene de vecino a una presa_1, entonces?
                reviveprey_1 = reviveprey_1 + 6 --Contador para
                 revivir como presa_1
                if(reviveprey_1 >= revivezone) then -- Cuando se supere
                 la zona de revivicion, que pasa?
                    behaviour = prey_1 -- Comportamiento del kilobot que
                    acaba de revivir como presa_1
                    hp = hpp --Puntos de vida al maximo
                    dead = 0 -- estado de vivo
                    reviveprey_1 = 0 -- Se reinicia el contador de la nueva presa
                    bandera=1 -- bandera de nueva presa viva
                    message_out(prey_1,0,dead)

                    w = sim.getInt32Signal("preycount_1")
                    sim.setInt32Signal("preycount_1", w+1) -- Una nueva
                    presa se agrega a la grafica

                    z = sim.getInt32Signal("deadcount")
                    sim.setInt32Signal("deadcount", z-1) -- Un individuo
                    muerto se quita de la grafica en una unidad
                end
            else -- Si el kilobot presente esta vivo (presa_1)
            y tiene una presa_1 cercas, entonces?
                -- No hay consecuencia
                 (si una presa es vecino de una presa no pasa nada)
            end
        elseif(message_rx[1] == predator_1) then -- si kilobot vecino
         es un depredador_1, entonces?
            if(dead == 2) then --Si la presa_1 esta muerta, entonces
             debe revivir como depredador_1
                revivepredator_1 = revivepredator_1 + 2.7
                if(revivepredator_1 >= revivezone) then -- Cuando se supere
                 la zona de revivicion, que pasa?
                    behaviour = predator_1 -- la presa revive como depredador_1
                    hp = hpd1 -- Puntos de vida de depredador 1 al maximo
                    dead = 0 -- Adquiere el estado de vivo
                    revivepredator_1 = 0 -- Se reinicia
                     el contador del depredador 1
                    bandera=1 -- Bandera de nuevo depredador_1 vivo
                    message_out(predator_1,0,dead)

                    x = sim.getInt32Signal("predatorcount_1")
                    sim.setInt32Signal("predatorcount_1", x+1) --Se agrega
                    un nuevo depredador_1 en la grafica

                    z = sim.getInt32Signal("deadcount")
                    sim.setInt32Signal("deadcount", z-1) -- Un individuo
                    muerto se quita de la grafica en una unidad
                end
            else -- Si el kilobot presente esta vivo
```

```
                (presa_1) y tiene a un depredador_1 cercas, entonces?
                    hp = hp - 2000 -- Proceso de devoracion
                end
            elseif(message_rx[1] == predator_2) then -- Si el kilobot vecino
            es un depredador_2, entonces?
                if(dead == 2) then -- Si el kilobot presente esta muerto, entonces?
                    revivepredator_2 = revivepredator_2 + 2.4
                        if(revivepredator_2 >= revivezone) then
                            behaviour = predator_2 --Comportamiento de depredador_2
                            hp = hpd2 -- Puntos de vida al maximo de depredador 2
                            dead = 0 -- Nuevo depredador_2, estado de vivo
                            revivepredator_2 = 0 -- Se reinicia
                            el contador del depredador_2
                            bandera=1 -- Bandera de nuevo depredador_2 vivo
                            message_out(predator_2,0,dead)

                            y = sim.getInt32Signal("predatorcount_2")
                            sim.setInt32Signal("predatorcount_2", y+1) --Se agrega
                            un nuevo depredador_2 a la grafica

                            z = sim.getInt32Signal("deadcount")
                            sim.setInt32Signal("deadcount", z-1) -- Un individuo
                             muerto se quita de la grafica en una unidad
                        end
                else -- Que le pasa a la presa cuando se encuentra a un depredador_2
                    hp = hp - 2000 -- Proceso de devoracion
                end
            end -- Si un kilobot presa_1
            se encuentra a un vecino (prey_1/predator_1/predator_2)
        end -- Si un kilobot vecino esta dentro del rango
        de distancia de 6 cm, entonces?
    end -- Si un kilobot vecino esta vivo, entonces?
end -- Si el kilobot actual recibe un mensaje de un kilobot vecino

if (hp>0) then -- Si la presa_1 tiene puntos de vida arriba de cero, entonces?
    hp = hp - 1 -- Sus puntos de vida van bajando mientras avanza el tiempo
else
    dead=2 -- De lo contrario adquiere el estado de muerto
    if(bandera==1) then -- Bandera logica para contabilizar
    unidades de individuos muertos

        bandera=0 -- La bandera cambia de valor para salir del ciclo inifinito

        z = sim.getInt32Signal("deadcount")
        sim.setInt32Signal("deadcount", z+1) --Cuando la presa tiene hp=0
        aumenta en una unidad la especie de muertos

        w = sim.getInt32Signal("preycount_1")
        sim.setInt32Signal("preycount_1", w-1) --Cuando la presa tiene hp=0
        se disminuye en una unidad su poblacion
    end
end


if (dead == 0) then --Si la presa_1 esta viva entonces?
    set_color(0,3,0) -- Color verde
```

```
        set_motor(math.random(0,40), 40-math.random(0,40)) -- Caminata aleatoria
    else
        set_color(3,0,0) -- Color rojo si la presa esta muerta
        set_motor(0,0) -- Sin movimiento
    end -- FIN DEL COMPORTAMIENTO DE LA PRESA_1

elseif(behaviour == predator_1) then -- Si el comportamiento es de un
DEPREDADOR_1, entonces?
    message_out(predator_1, 0, dead) -- Envia un mensaje a los demas kilobots
    diciendo que es un depredador_1 y esta vivo
    if(message_rx[6]==1) then -- Si el kilobot actual recibe un mensaje
     de un kilobot vecino
        if(message_rx[3] == 0) then -- Si un vecino (cualquiera) esta vivo, entonces?
            if(message_rx[4] <= limite) then -- Si un vecino (cualquiera)
             esta en el rango de 6 cm, entonces?
                if(message_rx[1] == prey_1) then -- Si el depredador_1 tiene
                de vecino a una presa_1, entonces?
                    if(dead == 2) then -- Si el estado actual del depredador_1
                     es muerto, entonces debe revivir como presa_1
                        reviveprey_1 = reviveprey_1 + 6
                        if(reviveprey_1 >= revivezone) then
                            behaviour = prey_1 -- Comportamiento de presa
                            hp = hpp -- Puntos de vida al maximo de presa
                            dead = 0 -- Adquiere el estado de vivo
                            reviveprey_1 = 0 -- Se reinicia el contador de la presa
                            bandera = 1 -- bandera de nueva presa viva
                            message_out(prey_1,0,dead)

                            w = sim.getInt32Signal("preycount_1")
                            sim.setInt32Signal("preycount_1", w+1) -- Ha revivido
                             una nueva presa_1

                            z = sim.getInt32Signal("deadcount")
                            sim.setInt32Signal("deadcount", z-1) --Se disminuye
                            en una unidad la poblacion especies muertas
                        end
                    else -- Si el estado actual del depredador es vivo, entonces
                     debe devorar a la presa_1
                        hp = hp + 100 -- Se incrementan los hp del depredador_1
                    end
                elseif(message_rx[1] == predator_1) then -- Si el depredador_1
                tiene de vecino a un depredador_1, entonces?
                    if(dead == 2) then --Si el estado actual del depredador_1
                    es muerto, entonces debe revivir como depredador_1
                        revivepredator_1 = revivepredator_1 + 2.7
                        if(revivepredator_1 >= revivezone) then
                            behaviour = predator_1 -- Comportamiento de depredador 1
                            hp = hpd1 -- Puntos de vida del depredador 1 al maximo
                            dead = 0 -- Adquiere el estado de vivo
                            revivepredator_1 = 0 -- Se reinicia el
                            contador de depredador_1
                            bandera = 1 -- bandera de nuevo depredador_1 vivo
                            message_out(predator_1,0,dead)

                            x = sim.getInt32Signal("predatorcount_1")
                            sim.setInt32Signal("predatorcount_1", x+1) --Ha revivido
```

```
                        un nuevo depredador_1

                        z = sim.getInt32Signal("deadcount")
                        sim.setInt32Signal("deadcount", z-1) --Se disminuye
                        en una unidad la poblacion especies muertas
                    end
                else -- Si el estado actual del depredador es vivo,
                 entonces hay competencia entre ambos depredadores_1
                    hp = hp - 250 --Se disminuyen los hp del depredador_1 actual
                end
            elseif(message_rx[1] == predator_2) then -- Si el depredador_1
            tiene de vecino a un depredador_2, entonces?
                if(dead == 2) then --Si el estado actual del depredador_1
                es muerto, entonces debe revivir como depredador_2
                    revivepredator_2 = revivepredator_2 + 2.4
                    if(revivepredator_2 >= revivezone) then
                        behaviour = predator_2 -- Comportamiento de depredador_2
                        hp = hpd2 -- Puntos de vida al maximo de depredador 2
                        dead = 0 -- Adquiere el estado de vivo
                        revivepredator_2 = 0 -- Se reinicia el
                         contador del depredador_2
                        bandera = 1 -- bandera de nuevo depredador_2
                        message_out(predator_2,0,dead)

                        y = sim.getInt32Signal("predatorcount_2")
                        sim.setInt32Signal("predatorcount_2", y+1) --Ha revivido
                         un nuevo depredador_2

                        z = sim.getInt32Signal("deadcount")
                        sim.setInt32Signal("deadcount", z-1) --Se disminuye
                         en una unidad la poblacion especies muertas
                    end
                else -- Si el estado actual del depredador es
                vivo, entonces hay competencia entre los depredadores 1 y 2
                    hp = hp - 250 --Se disminuyen los hp del depredador_1 actual
                end
            end -- Si un kilobot se encuentra a un vecino
            (prey_1/predator_1/predator_2)
        end -- Si un vecino (cualquiera) esta en el rango de 6 cm, entonces?
    end -- Si un vecino (cualquiera) esta vivo, entonces?
end -- Si el kilobot actual recibe un mensaje de un kilobot vecino

if(hp>0) then -- Si el depredador_1 tiene hp>0, entonces?
    hp = hp - 1 -- Se iran disminuyendo de uno en uno mientras el tiempo avanza
else
    dead = 2 -- Si el depredador_1 tiene hp=0, entonces adquiere el estado de muerto
    if(bandera == 1) then -- Bandera para contabilizar la muerte del depredador_1
        bandera = 0 -- Cambio de valor de la bandera para salir del ciclo infinito

        z = sim.getInt32Signal("deadcount")
        sim.setInt32Signal("deadcount", z+1) --El depredador_1 murio porque sus hp=0

        x = sim.getInt32Signal("predatorcount_1")
        sim.setInt32Signal("predatorcount_1", x-1) --El depredador_1 muere
        cuando sus hp=0
    end
```

```
    end

    if(dead == 0) then --Si el depredador_1 esta vivo, entonces?
        set_color(3,0,3) -- Color morado
        set_motor(math.random(0,40), 40-math.random(0,40)) -- Caminata aleatoria
    else
        set_color(3,3,3) -- Color blanco
        set_motor(0,0) -- Sin movimiento
    end -- FIN DEL COMPORTAMIENTO DEL DEPREDADOR_1

elseif(behaviour == predator_2) then -- Si el comportamiento es
de un depredador_2, entonces?
    message_out(predator_2, 0, dead) -- Envia un mensaje a los demas robots
    diciendo que es un depredador_2 vivo
    if(message_rx[6]==1) then -- Si el kilobot actual recibe un mensaje
    de un kilobot vecino
        if(message_rx[3] == 0) then -- Si hay robots vecinos vivos
        cercas, entonces?
            if(message_rx[4] <= limite) then -- Si un vecino (cualquiera) esta
             en el rango de 6 cm, entonces?
                if(message_rx[1] == prey_1) then -- Si un vecino es
                 una presa_1, entonces?
                    if(dead == 2) then -- Si el kilobot actual (depredador_2)
                     esta muerto, entonces debe revivir como presa_1
                        reviveprey_1 = reviveprey_1 + 6
                        if(reviveprey_1 >= revivezone) then
                            behaviour = prey_1 -- Comportamiento del kilobot
                            que acaba de revivir como presa_1
                            hp = hpp --Puntos de vida al maximo
                            dead = 0 -- estado de vivo
                            reviveprey_1 = 0 -- Se reinicia el
                             contador de la nueva presa
                            bandera=1 -- bandera de nueva presa viva
                            message_out(prey_1,0,dead)

                            w = sim.getInt32Signal("preycount_1")
                            sim.setInt32Signal("preycount_1", w+1) --Ha revivido
                             una nueva presa_1

                            z = sim.getInt32Signal("deadcount")
                            sim.setInt32Signal("deadcount", z-1) --Se quita
                            una unidad a la poblacion de muertos
                        end
                    else -- Si el estado del robot actual (depredador_2)
                     es vivo, y se encuentra a una presa cerca, entonces?
                        hp = hp + 100 -- El depredador_2 devora a la presa,
                        por lo que sus hp incrementan
                    end
                elseif(message_rx[1] == predator_1) then -- Si un vecino es
                un predator_1, entonces?
                    if(dead == 2) then -- Si el kilobot actual (depredador_2)
                     esta muerto, entonces debe revivir como predator_1
                        revivepredator_1 = revivepredator_1 + 2.7
                        if(revivepredator_1 >= revivezone) then
                            behaviour = predator_1 -- Comportamiento de depredador 1
                            hp = hpd1 -- Puntos de vida del depredador 1 al maximo
```

```
                        dead = 0 -- Adquiere el estado de vivo
                        revivepredator_1 = 0 -- Se reinicia
                         el contador de depredador_1
                        bandera = 1 -- bandera de nuevo depredador_1 vivo
                        message_out(predator_1,0,dead)

                        x = sim.getInt32Signal("predatorcount_1")
                        sim.setInt32Signal("predatorcount_1", x+1) -- Ha revivido
                         un nuevo depredador_1

                        z = sim.getInt32Signal("deadcount")
                        sim.setInt32Signal("deadcount", z-1) --Se quita una
                        unidad a la poblacion de muertos
                    end
                else -- Si el estado del robot actual (depredador_2) es vivo,
                y se encuentra a una depredador_1 cerca, entonces?
                    hp = hp - 250 -- Se le disminuyen los hp debido a
                    la competencia entre depredadores
                end
            elseif(message_rx[1] == predator_2) then -- Si un vecino es
             un depredador_2, entonces?
                if(dead == 2) then --Si el kilobot actual (depredador_2)
                 esta muerto, entonces debe revivir como predator_2
                    revivepredator_2 = revivepredator_2 + 2.4
                    if(revivepredator_2 >= revivezone) then
                        behaviour = predator_2 -- Comportamiento de depredador_2
                        hp = hpd2 -- Puntos de vida al maximo de depredador 2
                        dead = 0 -- Adquiere el estado de vivo
                        revivepredator_2 = 0 -- Se reinicia el
                        contador del depredador_2
                        bandera = 1 -- bandera de nuevo depredador_2
                        message_out(predator_2,0,dead)

                        y = sim.getInt32Signal("predatorcount_2")
                        sim.setInt32Signal("predatorcount_2", y+1) -- Ha revivido
                         un nuevo depredador_2

                        z = sim.getInt32Signal("deadcount")
                        sim.setInt32Signal("deadcount", z-1) --Se quita
                        una unidad a la poblacion de muertos
                    end
                else -- Si el estado del robot actual (depredador_2) es vivo,
                 y se encuentra a una depredador_2 cerca, entonces?
                    hp = hp - 250 -- Se le disminuyen los hp debido
                    a la competencia entre depredadores
                end
            end -- Comportamientos cuando hay vecinos
             cercas (prey_1/predator_1/predator_2)
        end -- Si los robots vecinos se encuentran dentro del
        rango de 6 cm o menos, entonces?
    end -- Si hay robots vecinos vivos cercas, entonces?
end -- Si el kilobot actual recibe un mensaje de un kilobot vecino

if(hp>0) then -- Si los hp del depredador_2 son positivos entonces esta vivo
    hp = hp-1 -- Mientras avanza el tiempo se iran disminuyendo sus hp
else
```

```
                dead = 2 -- Cuando el depredador_2 tenga hp=0 adquirira el estado de muerto
                if(bandera == 1) then -- Bandera para contabilizar la muerte del depredador_2
                    bandera = 0 -- Cambio de valor de la bandera para salir del ciclo infinito

                    z = sim.getInt32Signal("deadcount")
                    sim.setInt32Signal("deadcount", z+1) --Se aumenta una unidad
                    a la poblacion de muertos

                    y = sim.getInt32Signal("predatorcount_2")
                    sim.setInt32Signal("predatorcount_2", y-1) -- Ha muerto un depredador_2
                end
            end

            if(dead == 0) then -- Si el depredador_2 esta vivo, entonces?
                set_color(0,0,3) -- Color azul
                set_motor(math.random(0,40), 40-math.random(0,40)) -- Caminata aleatoria
            else -- Cuando el depredador_2 esta muerto, entonces?
                set_color(0,3,3) -- Color aqua
                set_motor(0,0) -- Sin movimiento por el estado de muerto
            end
        end -- FIN DEL CODIGO 2 DEPREDADORES Y 1 PRESA



    --////////////////////////////////////////////////////////////////////////////////
    --//END OF USER CODE
    --////////////////////////////////////////////////////////////////////////////////
    end
```

10.4. Algoritmo para obtener la gráfica de comportamientos de los robots móviles (CoppeliaSim)

```
function sysCall_init()
    -- Inicializar la grafica
    graphHandle = sim.getObject("/Graph")

    --Agregar etiquetas de las especies a la grafica
    presas = sim.addGraphStream(graphHandle, "Presas", "User unite", 0, {0,1,0})
    depredadores = sim.addGraphStream(graphHandle, "Depredadores", "User unite", 0, {1,0,1})
    muertos = sim.addGraphStream(graphHandle, "Muertos", "User unite", 0, {1,0,0})

    --Crear archivo de texto para almacerar los datos
    archivo = io.open("C:/Users/namh0/OneDrive/Documentos/Maestria UABC/Semestre #4/
    Kilobots en coppeliasim/Lotka-Volterra en Coppeliasim/Datos.txt", "w+")
end

function sysCall_sensing()
    sim.setGraphStreamValue(graphHandle, presas, sim.getInt32Signal("preycount"))
    sim.setGraphStreamValue(graphHandle, depredadores, sim.getInt32Signal("predatorcount"))
    sim.setGraphStreamValue(graphHandle, muertos, sim.getInt32Signal("deadcount"))

    if presas then
        sim.handleGraph(graphHandle, sim.getSimulationTime())
    end
```

```
    if depredadores then
        sim.handleGraph(graphHandle, sim.getSimulationTime())
    end

    if muertos then
        sim.handleGraph(graphHandle, sim.getSimulationTime())
    end

    --Almacanemiento de datos en el archivo de texto
    archivo: write(string.format("%1f %1i %1i %1i \n", sim.getSimulationTime(),
    sim.getInt32Signal("preycount"), sim.getInt32Signal("predatorcount"),
     sim.getInt32Signal("deadcount")))
end

function sysCall_cleanup()
    archivo: close()
end
```