Trabajo Fin de Grado

Sincronización en sistemas discretos: Aplicación a los swarmalators

Synchronization in discrete systems: Application to swarmalators

Fabián Hormigo Pereila

Alumno del Grado en Física. Curso 2022-2023

Tutores:

Prof. Francisco de Paula Jiménez Morales

Prof.ª Mª Carmen Lemos Fernández

Departamento de Física de la Materia Condensada

Facultad de Física. Sevilla, Mayo 2023

Índice

1 Resumen

Este documento se centra en la temática de los sistemas autoorganizados, aquellos que presentan propiedades emergentes a partir de la interacción entre sus componentes individuales sin necesidad de una interacción externa. Se presenta el modelo de Kuramoto y su aplicación en el desarrollo de modelos autoorganizativos más complejos, como el "Swarmalator". Este modelo es especialmente útil para estudiar fenómenos biológicos y sociales que implican comportamiento colectivo y movilidad espacial.

Además, se pone de manifiesto la importancia de la modelización numérica en el estudio de los sistemas autoorganizados, y se usa el lenguaje de programación Python como lenguaje actual, potente y versátil. Se sabe que la simulación computacional es una herramienta valiosa para el estudio de los sistemas autoorganizados, ya que permite explorar diferentes escenarios y analizar como cambian las propiedades emergentes del sistema a medida que cambian las condiciones iniciales o los valores de los parámetros del modelo. En particular, se discute cómo los modelos numéricos pueden ser utilizados para simular fenómenos complejos como la formación de patrones en células o el comportamiento colectivo en animales. Estos modelos pueden proporcionar información valiosa sobre como surgen las estructuras complejas a partir de interacciones simples.

2 Introducción

El concepto de autoorganización fue introducido por Immanuel Kant en su obra “Crítica al juicio”. Para Kant los sistemas autoorganizados, en concreto los biológicos, dependen exclusivamente de las interacciones existentes entre sus pequeñas partes. Estos adquieren atributos de conjunto e identidad propia “per se”[1].

En lo cotidiano vemos claros ejemplos, desde los océanos dónde observamos el movimiento unísono de bancos de peces, pasando por las colonias de insectos terrestres, hasta los cielos en los cuáles vemos el movimiento síncrono de bandadas de pájaros. Todos ellos hacen fehaciente la autoorganización como intrínseco de lo natural.

Figura 1: Ejemplos de sistemas autoorganizativos naturales. De izquierda a derecha, banco de peces[2]*, panal de abejas*[3] *y bandada de pájaros*[4].

En microbiología se pueden observar grupos de bacterias que presentan estructura compleja, construyendo así entes más complejos que la suma de sus partes.

Figura 2: Biofilm de una cepa enterotoxigénica de Bacillus cereus sobre agar TY teñido con Rojo Congo y azul de Coomassie en condiciones de anaerobiosis a 30ºC [5].

En biología se presentan los más claros ejemplos; sin embargo, los sistemas autoorganizativos abarcan disciplinas en apariencia dispares, los grupos sociales humanos también presentan propiedades autoorganizativas. Un ejemplo de ello es la diferenciación de la población en zonas urbanas, donde conjuntos de individuos tienden a asentarse en zonas residenciales con los mismos intereses culturales [6].

Por otra parte, en física del estado sólido, los procesos de nucleación en dispersiones coloidales y el crecimiento policristalino pueden ser entendidos como sistemas autoorganizados[7].

En definitiva, con todo lo anterior queda patente que los sistemas autónomos adquieren una dimensión importante en numerosos fenómenos tanto cotidianos como académicos.

En este Trabajo de Fin de Grado (TFG), nos vamos a centrar en sistemas biológicos autoorganizados, estudiándolos desde la perspectiva de la simulación computacional.

2.1 Antecedentes de modelización

2.1.1 Modelo de Kuramoto

Los primeros modelos matemáticos propuestos para ciertos sistemas biológicos autoorganizativos se basan en la hipótesis de sincronización que Winfree formuló en 1967[8]. Años más tarde, en 1975, Kuramoto perfiló estas ideas[9].Estos modelos se basan en el acoplamiento de osciladores, fijando para cada elemento del conjunto un estado interno o fase.

El modelo de Kuramoto consta de una colección de N osciladores acoplados con frecuencias naturales w_i cuyas fases evolucionan según la relación:

$$\dot{\theta}_i = w_i + \sum_{j=1}^{N} k_{ij} \sin(\theta_j - \theta_i) \qquad \text{[Ec.1]}$$

Cada uno de estos osciladores tiene una dinámica propia dada por su frecuencia natural e interaccionan con el resto en función de la matriz de acoplo k_{ij}. Estos osciladores tienden a agruparse de forma que adquieren o no sincronización colectiva. Fijaremos la matriz de acoplo como un escalar K, y tomaremos las frecuencias naturales como nulas y normalizaremos la expresión anterior por el número de osciladores como el modelo original.

En consecuencia:

$$\dot{\theta}_i = K/N \sum_{j=1}^{N} \sin(\theta_j - \theta_i) \qquad \text{[Ec.2]}$$

Se define una constante de estado global llamada parámetro de orden $Re^{i\Psi}$:

$$Re^{i\Psi} = \frac{1}{N}\sum_{j=1}^{N} e^{i\theta_j} \qquad \text{[Ec.3]}$$

Un número complejo cuyo módulo R está acotado entre 0 y 1,el valor del módulo R cuantifica cuan ordenado se encuentra nuestro sistema. Valores del módulo R próximos a la unidad indican un mayor orden, es decir, un mayor número de osciladores tienen la misma fase.

En una primera simulación, todos los osciladores parten de fases iniciales arbitrarias y uniformemente distribuidas en el intervalo $(0,2\pi)$ y limitamos el número de partículas a N=10. Siguiendo el esquema de la figura 3. Los resultados se recogen en la figura 4 para diferentes valores de K. Las fases θ_i están representadas en la circunferencia unidad.

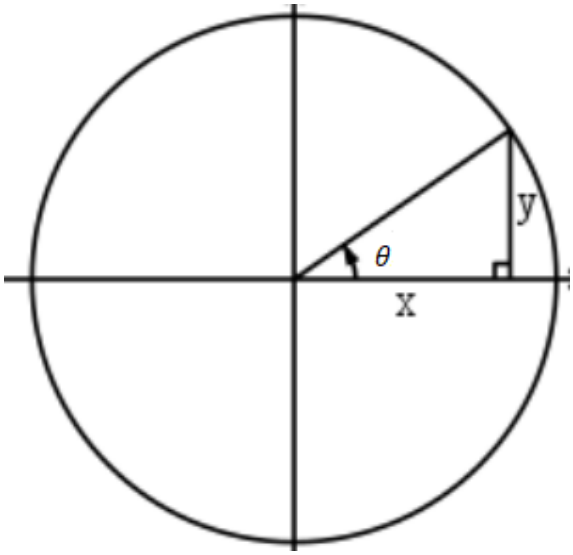

Figura 3:Representación de las fases en la circunferencia unidad, esquema aclarativo.$x = cos(\theta); y = sen(\theta)$.

K= -0.1 K=1

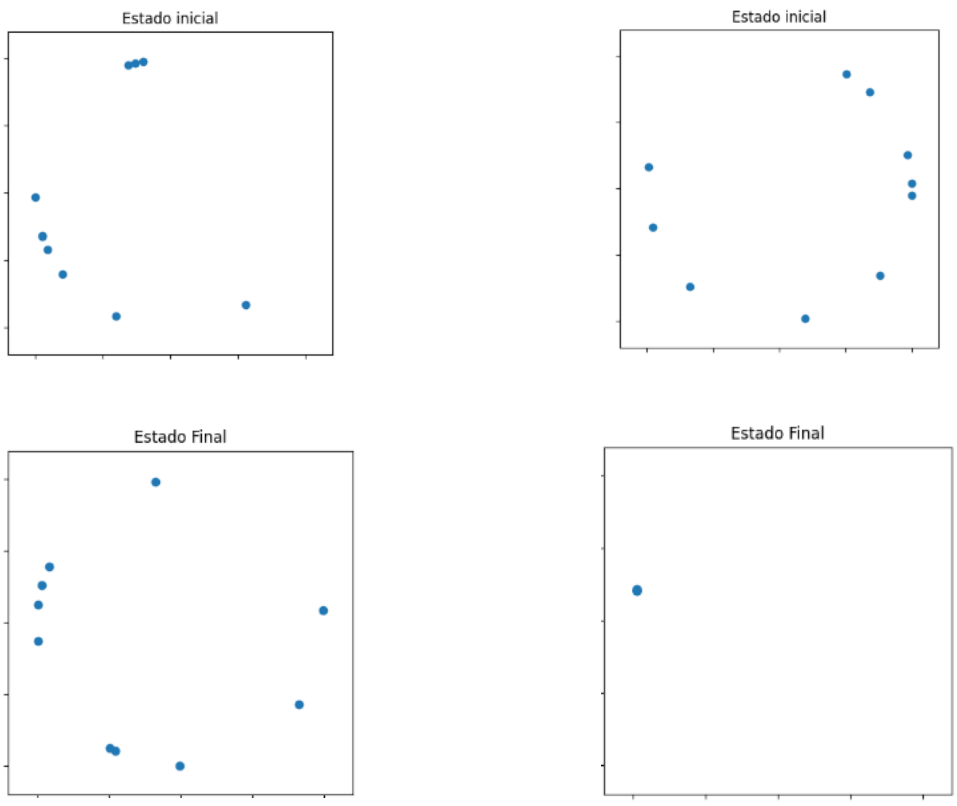

Figura 4:Representación de las fases para el modelo de Kuramoto para diferentes valores de la constante de acoplo K. Fijado un total de N=10, valores iniciales de las fases aleatorios uniformemente distribuidos entre (0,2π), resolución numérica mediante el método de Adams con paso de tiempo dt=0.015 en el intervalo(0,15)s. Se recogen en coordenadas x-y : estado inicial y estado final .Elaboración propia.

Inspeccionando la imagen anterior se pueden concluir los siguientes resultados:

-**Valores negativos de K producen desorden**, debido a que los osciladores se desacoplan, y **valores positivos de K producen orden**.

-La **distribución de fases en el estado final:**

A) Para valores negativos K=-0.1. Los valores de las fases se distribuyen a lo largo del intervalo (0,2π). Desorden.

B) Para valores positivos K=1. Los valores de las fases convergen a un solo valor. Orden y sincronización.

Aunque lo hemos presentado de forma somera, el modelo de Kuramoto supone una buena base para el desarrollo de modelos autoorganizativos más complejos. Hemos observado que, partiendo de una configuración inicial aleatoria, la dinámica interna entre los elementos del sistema permite a éste evolucionar hacia estados ordenados, como se muestra en el caso en que la constante de acoplo K es positiva.

2.2 Swarmalators

El concepto de "Swarmalator" hace referencia a un sistema de partículas oscilantes u osciladores con capacidad de acoplarse entre sí, además cuentan con libertad para moverse en el espacio. En definitiva destacan por su capacidad para comunicarse a través del fenómeno de acoplamiento y su movilidad, dando lugar a propiedades emergentes como la sincronización del sistema o configuraciones espaciales de alta complejidad.

2.2.1 Marco teórico

Entre los diferentes estados del arte en la simulación de sistemas biológicos autoorganizados destacan los modelos que O'Keeffe y Steven Strogatz nos presentan en sus trabajos [10] y [11] . Estos modelos recogen las ideas de osciladores acoplados en el modelo de Kuramoto y añaden movilidad espacial.

Las ecuaciones de evolución presentadas para estos modelos son:

$$\dot{x_i} = \frac{1}{N}\sum_{j\neq i}^{N}\left[I_{att}(x_j - x_i) - I_{rep}(x_j - x_i)\right] \qquad \text{[Ec.4]}$$

Donde x_i y $\dot{x}_i$ son respectivamente posición y velocidad de la partícula i-ésima.

$I_{att}(x_j - x_i)$ es el término encargado de la interacción atractiva entre elementos del conjunto. $I_{rep}(x_j - x_i)$ es el término encargado de la interacción repulsiva entre elementos del conjunto.

Tanto la interacción repulsiva como atractiva dependen de las posiciones relativas de cada elemento del sistema.

En concreto, el caso de estudio propuesto por O'Keeffe es el siguiente sistema de ecuaciones diferenciales acopladas: dinámica espacial [Ec.5] y dinámica de fase [Ec.6].

$$\dot{\mathbf{x}}_i = \frac{1}{N}\sum_{j\neq i}^{N}\left[\frac{\mathbf{x}_j - \mathbf{x}_i}{|\mathbf{x}_j - \mathbf{x}_i|}\left(1 + J\cos(\theta_j - \theta_i)\right) - \frac{\mathbf{x}_j - \mathbf{x}_i}{|\mathbf{x}_j - \mathbf{x}_i|^2}\right] \qquad \text{[Ec.5]}$$

$$\dot{\theta}_i = w_i + \frac{K}{N}\sum_{j\neq i}^{N}\frac{\sin(\theta_j - \theta_i)}{|\mathbf{x_j} - \mathbf{x_i}|} \qquad \text{[Ec.6]}$$

Donde θ_i indica la fase de la partícula i-ésima.

El sistema de ecuaciones propuesto reduce la interacción de la dinámica fasorial a primeros vecinos y al valor de la constante de acoplo K. Por otra parte, las ecuaciones de evolución para la dinámica espacial dependen de la diferencia entre fases con el resto de los osciladores, del parámetro J que pondera el acoplo entre la dinámica de fases y la espacial, y de las posiciones con respecto a la colección.

Realizamos la aproximación a osciladores idénticos y velocidad nula $v_i = 0$ y, por tanto, $w_i = 0$. De las simulaciones del modelo se extraen un total de 5 estados: 3 estacionarios y 2 no

estacionarios para 2 grados de libertad espacial. Presentamos a continuación estos estados en las posiciones x e y, y el valor de la fase está representado con diferentes colores.

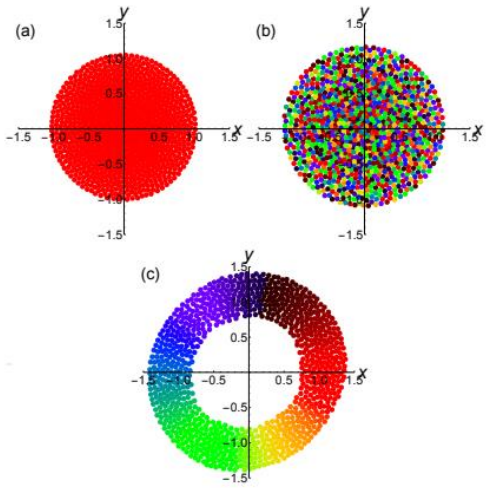

Figura 5:Estados estacionarios para el modelo de O'keeffe. Dos grados de libertad espacial x-y; N=1000; T = 100 unidades de tiempo y paso dT = 0.1. Condiciones iniciales en el espacio uniformemente distribuidas entre (-2,2). Condiciones iniciales de las fases uniformemente distribuidas entre (-π, π). Modificada de [10].

(a) Estado síncrono estático (J, K) = (0.1, 1). (b) Estado asíncrono estático (J, K) = (0.1, −1).

(c) Estado de fase estática (J, K) = (1, 0).

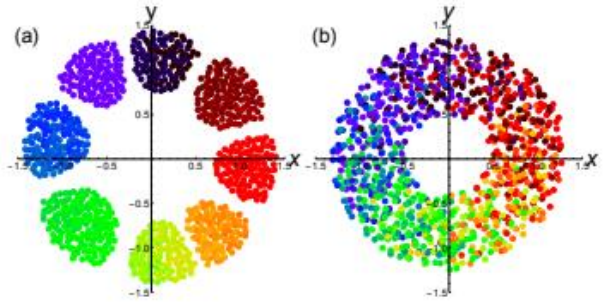

Figura 6:Estados no estacionarios para el modelo de O'keeffe. Dos grados de libertad espacial. N=1000. T = 100 unidades de tiempo y paso dt = 0.1. Condiciones iniciales en el espacio uniformemente distribuidas entre (-2,2). Condiciones iniciales de las fases uniformemente distribuidas entre (-π, π). Modificada de [10].

(a)Estado clústeres fase activa (J, K) = (1, −0.1). (b)Estado de fase activa (J, K) = (1, −0.75).

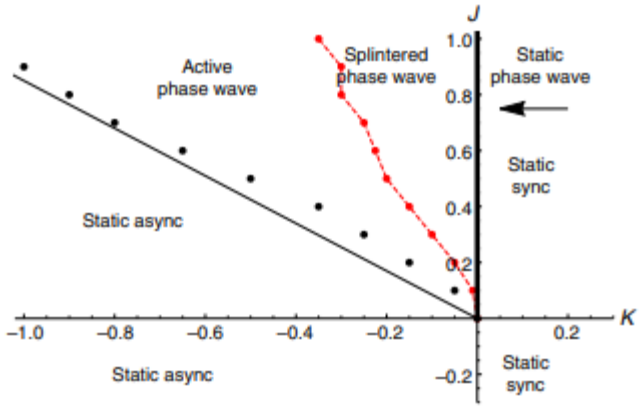

Figura 7:Diagrama de fases para el modelo de O'keeffe. Extraída de [11].

Otros autores han desarrollado modelos similares modificando la interacción a un entorno finito. El modelo de Hong[12] restringe la interacción completa a un cierto entorno finito de cada individuo de la población determinado por el valor r_c.

$$\dot{\mathbf{r}}_i = \frac{1}{N_i(r_c)} \sum_{j \in \Lambda_i(r_c)} \left[\left(1 + J\cos(\theta_j - \theta_i)\right) - \frac{1}{r_{ij}^2} \right] (\mathbf{r}_j - \mathbf{r}_i) \qquad \text{[Ec.7]}$$

Donde el dominio $\Lambda_i(r_c)$ está definido como el disco de radio r_c con respecto al centro de la partícula i-ésima. En otras palabras, todo elemento j-ésimo que se encuentre en ese dominio interaccionará con la partícula i-ésima. $N_i(r_c)$ es un valor del número de partículas que cumplen la condición anterior.

La dinámica de fases viene dada por:

$$\dot{\theta}_i = \omega_i + \frac{K}{N} \sum_{j \neq i} \frac{\sin(\theta_j - \theta_i)}{|\mathbf{x_j} - \mathbf{x_i}|} \qquad \text{[Ec.8]}$$

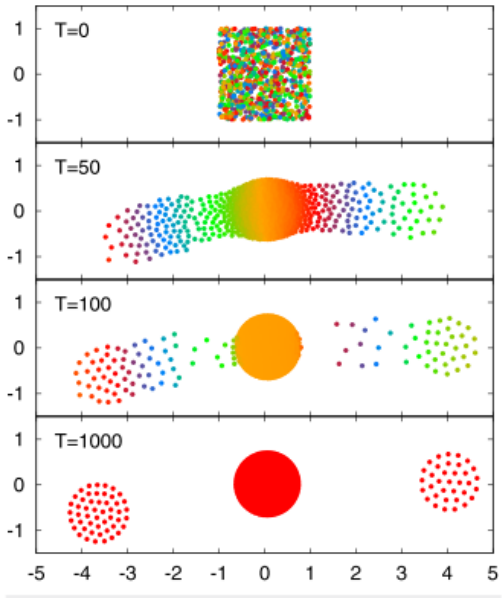

Figura 8:Modelo de Hong. Formación del patrón de estado de sincronización para r_c = 1.5 en un sistema de N = 800 con K= 0.1 y J = 1 que corresponde al límite estacionario. Cada panel es una instantánea en el tiempo T especificado en la esquina superior izquierda. Para la condición inicial, se utilizaron posiciones aleatorias uniformemente distribuidas en un cuadrado de L × L centrado en el origen para L = 2. Extraída y modificada de [12].

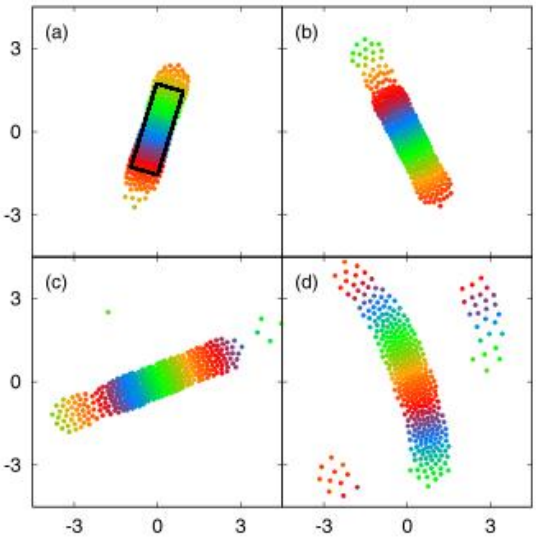

Figura 9:Modelo de Hong Varios patrones en forma de barra en un sistema de N = 400 con K = 0 y J = 1: (a) - (d) se obtienen para r_c = 1.8, 1.6, 1.4 y 1.2, respectivamente. El rectángulo negro en el patrón de (a) tiene una longitud de π y un ancho de 1. Extraída y modificada de [12].

Observamos que para este tipo de interacciones se favorece la creación de clústeres y estructuras alargadas similares a paramecios y bacilo bacterias.

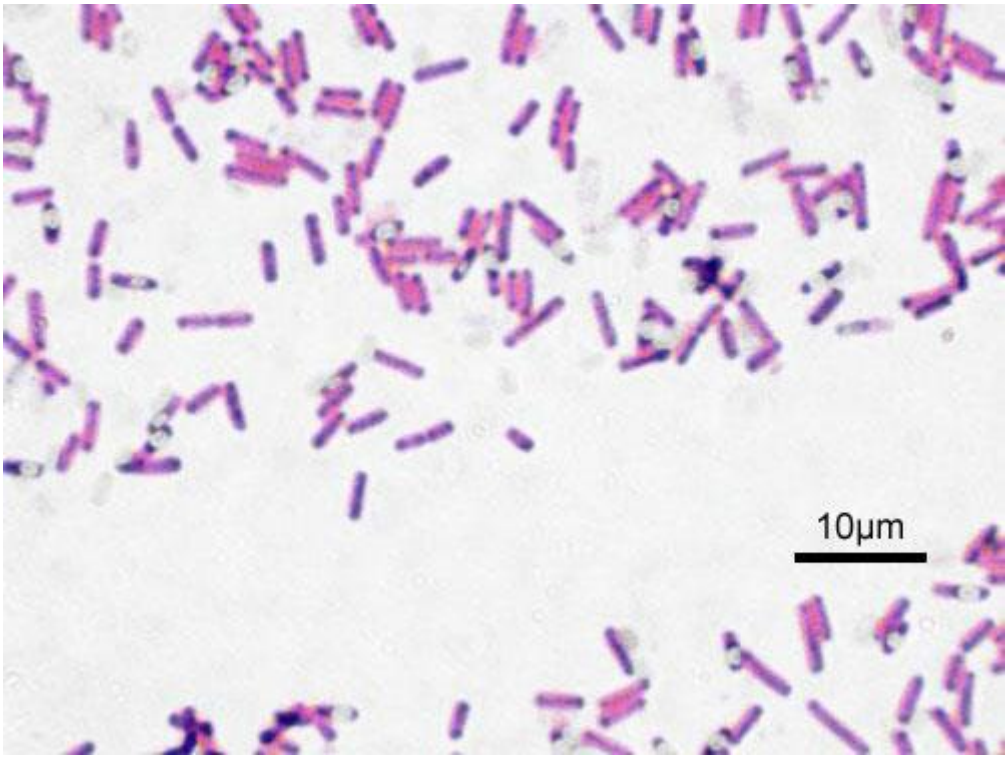

Figura 10:Bacillus subtilis, con tinción de Gram[13].

Por otra parte, otros autores como Jiménez-Morales, optan por la modificación del potencial repulsivo, modelando éste de tipo gaussiano [14] y [15]:

$$\dot{\mathbf{x}}_i = \frac{1}{N-1}\sum_{j\neq i}^{N}(\mathbf{x}_j - \mathbf{x}_i)\left[\frac{\left[1 + J\cos(\theta_j - \theta_i)\right]}{|\mathbf{x}_j - \mathbf{x}_i|} - \frac{1}{\sigma}e^{-|\mathbf{x}_j - \mathbf{x}_i|^2/\sigma}\right] \qquad \text{[Ec.9]}$$

$$\dot{\theta}_i = \frac{K}{N-1}\sum_{j\neq i}\frac{\sin(\theta_j - \theta_i)}{|\mathbf{x}_j - \mathbf{x}_i|} \qquad \text{[Ec.10]}$$

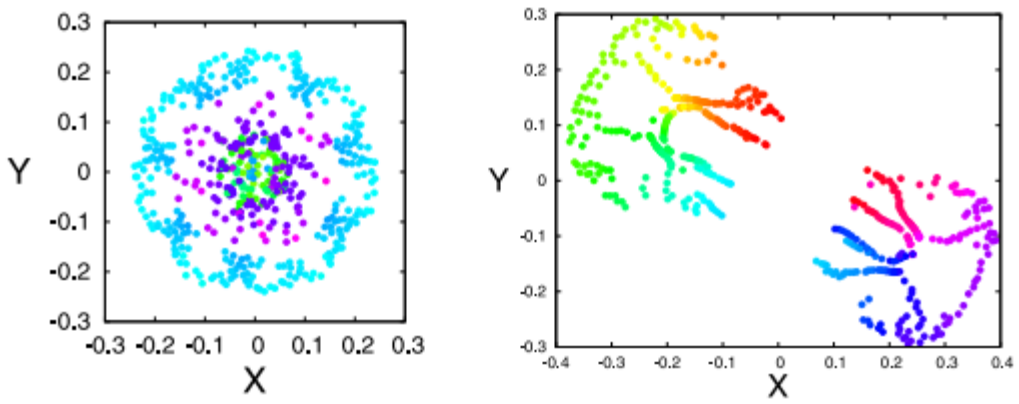

Figura 11:Estados "splintered" obtenidos mediante potencial repulsivo Gaussiano.

Izq K = 0.95, σ = 0.4; dcha K= 0, σ = 0.5. Extraída de [14].

El potencial repulsivo gaussiano de corto alcance genera estructuras estratificadas (figura 11 izq); también se observan estados de sincronización conjunta similares a los ya expuestos en los demás modelos y escisiones o clústeres en el sistema de partículas (figura 11 dcha).

Nuestra aportación en este TFG es modificar el potencial repulsivo del modelo de O'Keeffe, y resolver mediante métodos numéricos con el lenguaje de programación Python las ecuaciones resultantes de la modificación propuesta.

3 "Swarmalator" con potencial repulsivo a cortas distancias.

El modelo propuesto es una modificación del modelo de O'Keeffe, el potencial repulsivo solo se encontrará presente a distancias menores que un cierto parámetro d.

Las ecuaciones de nuestro modelo son:

$$\dot{\mathbf{x}}_i = \sum_{j\neq i}^{N} \left[\frac{1}{N} \frac{\mathbf{x}_j - \mathbf{x}_i}{|\mathbf{x}_j - \mathbf{x}_i|} \left(1 + J\cos(\theta_j - \theta_i)\right) - \delta\left(|\mathbf{x}_j - \mathbf{x}_i|\right) \frac{\mathbf{x}_j - \mathbf{x}_i}{|\mathbf{x}_j - \mathbf{x}_i|^2} \right] \qquad \text{[Ec.11]}$$

Donde $\delta\left(|\mathbf{x}_j - \mathbf{x}_i|\right)$ es definido como:

$$\delta\left(\left|\mathbf{x}_j - \mathbf{x}_i\right|\right) = 0;\ \text{si}\ \left|\mathbf{x}_j - \mathbf{x}_i\right| > d \qquad \text{[Ec.12]}$$

$$\delta\left(\left|\mathbf{x}_j - \mathbf{x}_i\right|\right) = \frac{1}{N_i(r_d)};\ \text{si}\ \left|\mathbf{x}_j - \mathbf{x}_i\right| < d$$

y $N_i(r_d)$ es definido como el número de elementos que se encuentran a una distancia suficiente del elemento i-ésimo para sufrir interacción repulsiva, es decir, elementos a una distancia menor que d.

La dinámica de fases para cada elemento de la colección vendrá dada por:

$$\dot{\theta}_i = \frac{K}{N}\sum_{j\neq i}^{N} \frac{\sin\left(\theta_j - \theta_i\right)}{\left|\mathbf{x_j} - \mathbf{x_i}\right|} \qquad \text{[Ec.13]}$$

Exploraremos sistemas con 2 grados de libertad espacial dados por las [Ec.11] y [Ec.13], es decir, cada elemento del conjunto tendrá 3 coordenadas. Como resultado obtenemos un sistema de 3N ecuaciones diferenciales acopladas, donde N indica el número de partículas.

A continuación presentamos los estados finales obtenidos mediante la integración de las ecuaciones de evolución propuestas. En todas las simulaciones hemos usado como valores J=1 y N=100. Cada figura muestra la posición de las partículas en coordenadas x e y, así como la fase θ_i de cada partícula con colores. Los ejes de referencia para cada una de las imágenes indican los diferentes valores que toman las constantes K y d.

$K \in (-1,1)$ en saltos de 0.2 y $d \in (1,6)$ en saltos de 0.5.

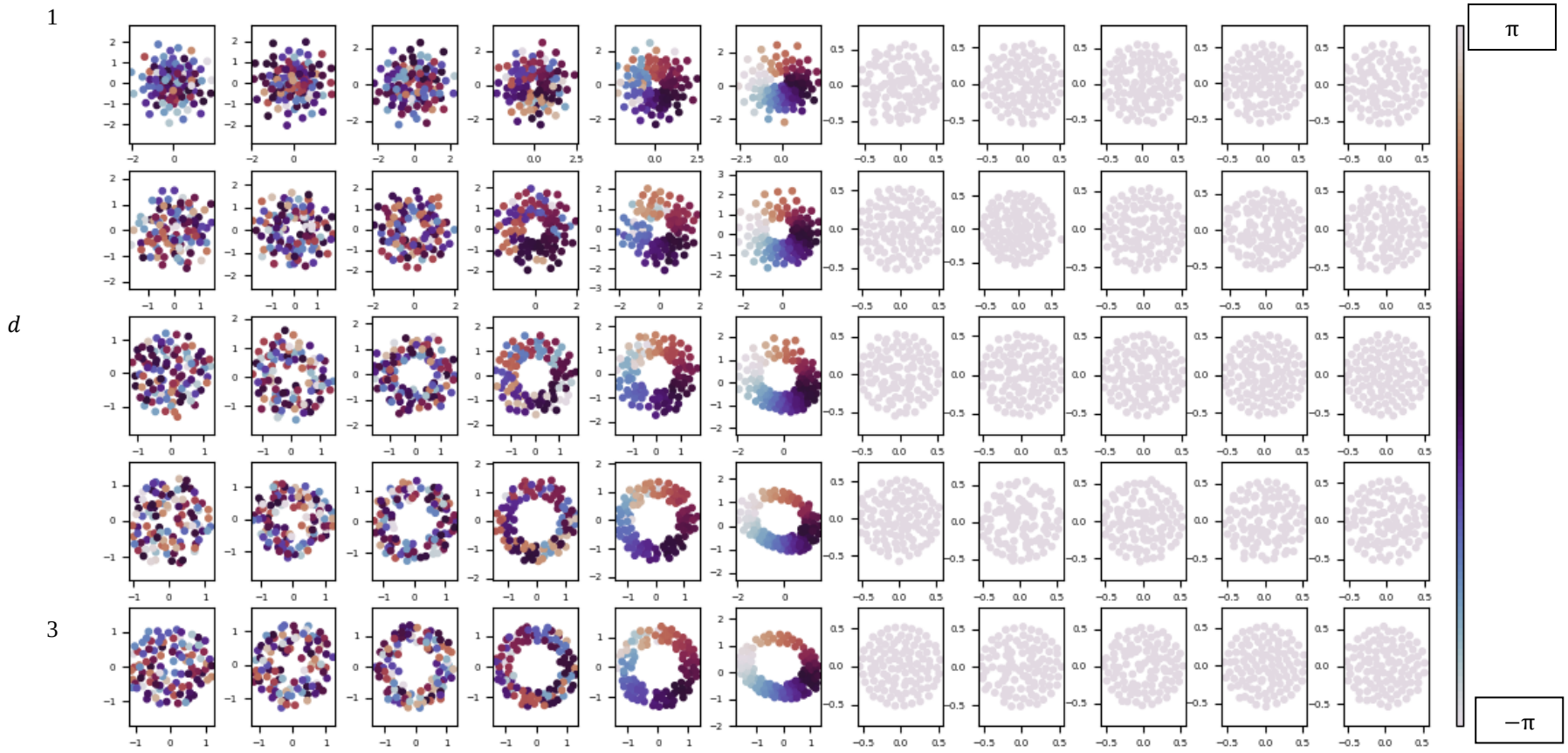

Figura 12: Espacio de estados. Estados finales para el modelo de potencial repulsivo existente a cortas distancias. Representados con respecto a su centro de masas.

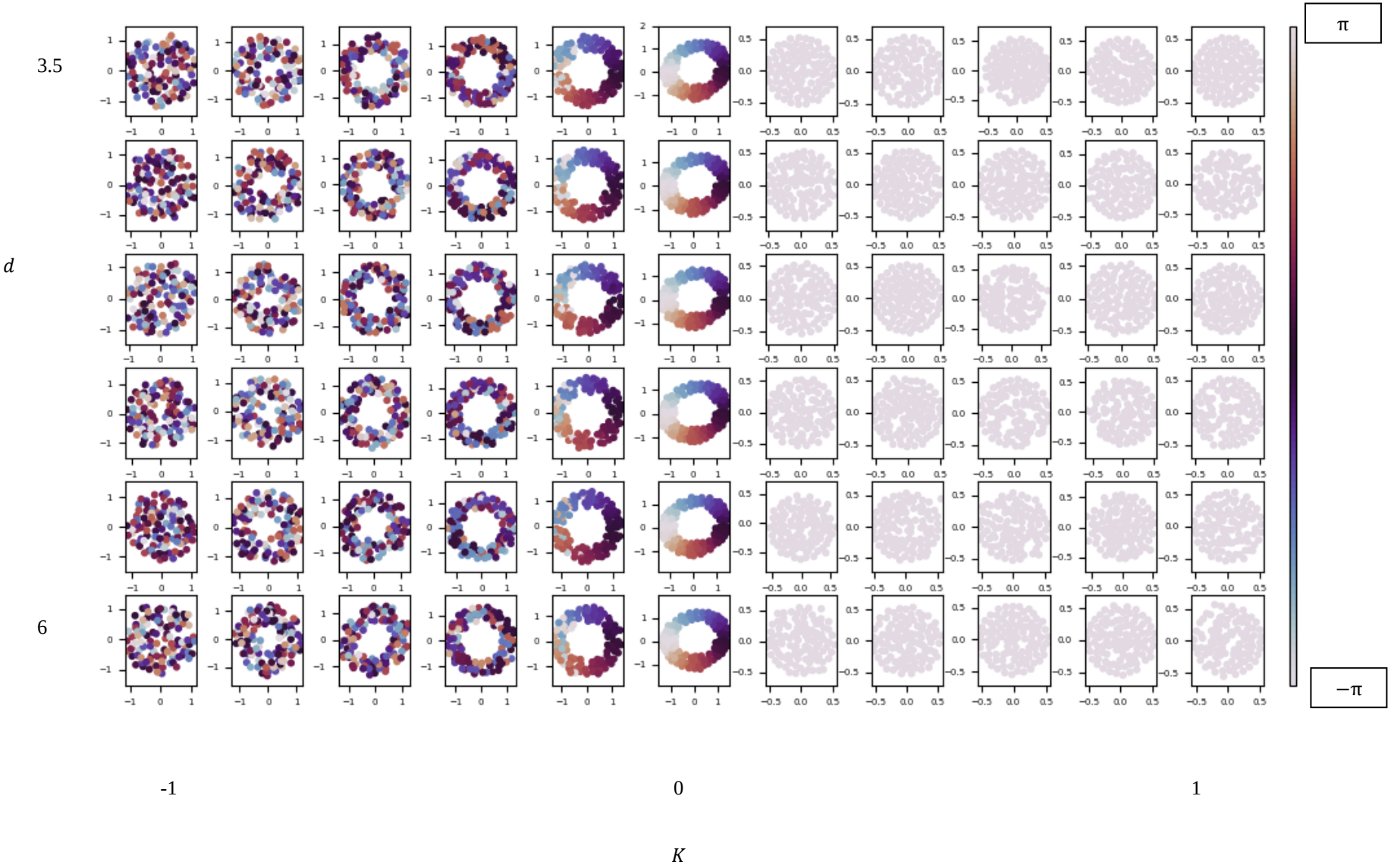

Figura 13:Espacio de estados. Estados finales para el modelo de potencial repulsivo existente a cortas distancias. Representados con respecto a su centro de masas.

Las figuras 10 y 11 presentan el espacio de estados explorado del modelo propuesto en función de los parámetros K y d, mostrando un total de 121 estados finales.

El método numérico utilizado para resolver el sistema acoplado de ecuaciones diferenciales es el método Runge-Kutta de orden 4, en un intervalo de tiempo $\mathrm{t} = (0{,}200)$, con un paso de tiempo $dt = 0.001$.Las condiciones iniciales para la posición de cada elemento y la fase provienen de una distribución aleatoria en los intervalos $\mathrm{x_i}(0), \mathrm{y_i}(0), \theta_\mathrm{i}(0) \in [0{,}10] \times [0{,}10] \times [0{,}2\pi]$.

Los métodos de integración numérica Runge-Kutta son una familia de métodos para aproximar la solución de ecuaciones diferenciales y problemas de valor inicial.

$$\frac{\mathrm{d}\mathbf{I}}{\mathrm{dt}} = \mathrm{f}(\mathrm{t}, \mathbf{I}) \qquad \text{[Ec.14]}$$

$$\mathbf{I}_{\mathrm{t+dt}} = \mathbf{I}_\mathrm{t} + \int_\mathrm{t}^{\mathrm{t+dt}} \mathrm{f}(\mathrm{t}, \mathbf{I})\, \mathrm{dt}$$

En nuestro caso $\mathbf{I}_\mathrm{t}$ indica el vector de posiciones y fases de todo el conjunto en cada instante de tiempo; cada instante $\mathbf{I}_{\mathrm{t+dt}}$ está conectado con el anterior mediante la integral de la función $\int_\mathrm{t}^{\mathrm{t+dt}} \mathrm{f}(\mathrm{t}, \mathbf{I})\, \mathrm{dt}$. Así los métodos de Runge-Kutta estiman el valor de la integral en ese intervalo y nos proporcionan el estado del sistema en la siguiente interacción, es decir, en el instante de tiempo posterior.

En las imágenes 10 y 11 se representan con colores las posiciones y el valor de las fases de los estados finales para cada par de valores d y K.

Para caracterizar los diferentes estados definimos 3 parámetros globales similares a los que define Jiménez-Morales en su modelo [14].

El parámetro de orden modificado S, mide la correlación entre las fases de cada elemento y su distribución espacial.

$$W_{\pm} = \frac{1}{N}\sum_{j=1}^{N} e^{i(\phi_j \pm \theta_j)} \qquad \text{[Ec.15]}$$

$$\phi_i = \tan^{-1}\left(\frac{y_i}{x_i}\right)$$

$$S = \text{máx}\left(\left|W_{\pm}\right|\right)$$

La energía cinética T y el momento lineal total P, servirán para diferenciar estados que rotan y estados que no se mueven.

$$T = \frac{1}{2N}\sum_{i=1}^{N} |\dot{\mathbf{x}}_i|^2 \qquad \text{[Ec.16]}$$

$$P = \frac{1}{N}\left|\sum_{i=1}^{N} \dot{\mathbf{x}}_i\right|$$

En el caso de estudio los valores de P son pequeños en comparación con T, en consecuencia, el centro de masas de los “swarmalators” estudiados apenas se mueve.

Podremos distinguir diferentes tipos de estados; estados con una energía cinética $T > 0$ indicará que las partículas se encuentran rotando.

S es un parámetro definido en el intervalo (0,1); valores cercanos a uno indicarán una alta correlación entre la posición y la fase de cada elemento.

A continuación, se muestran los mapas de calor con los resultados de S, T y P.

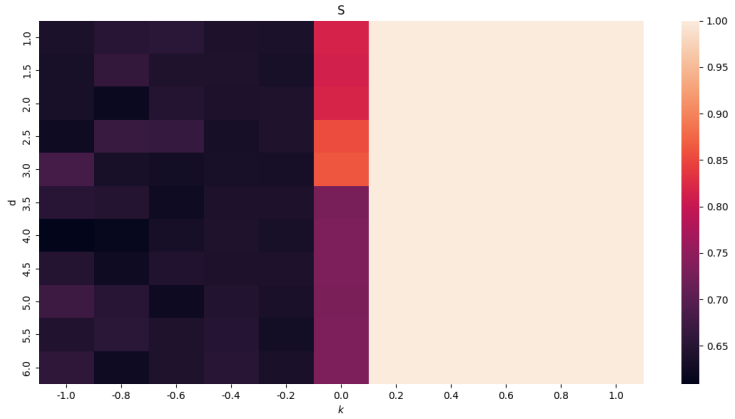

Figura 14:Mapa de calor del parámetro S. Valores cercanos a S=1 indican mayor correlación entre la distribución espacial y la fase.

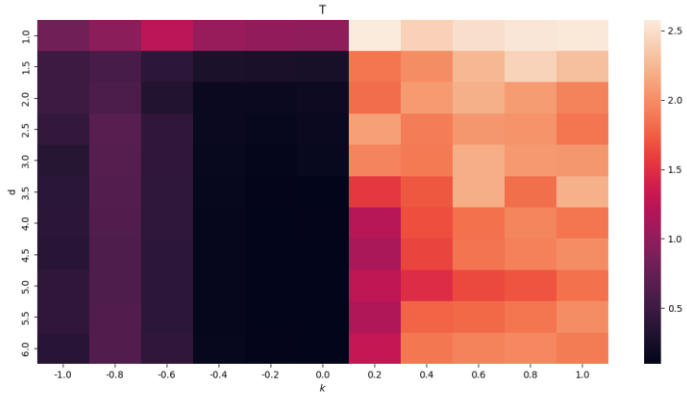

Figura 15:Mapa de calor del parámetro T, mayores valores de energía cinética indican mayor giro de las partículas con respecto al centro de masas.

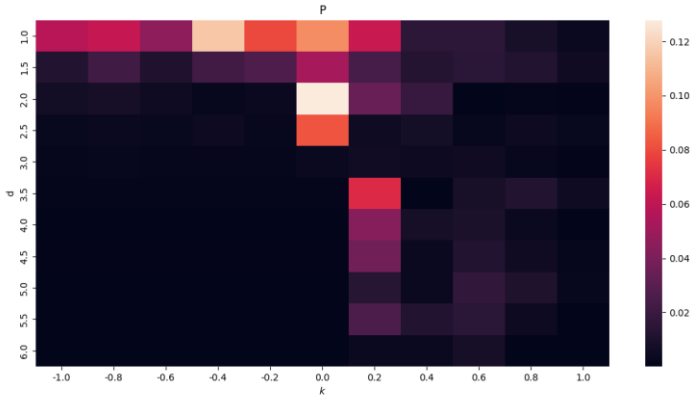

Figura 16:Mapa de calor del parámetro P, mayores valores de P indican mayores velocidades del centro de masas.

Con todo ello identificamos un total de 3 estados: estado síncrono, estado de fase estática y estado asíncrono de fase dinámica.

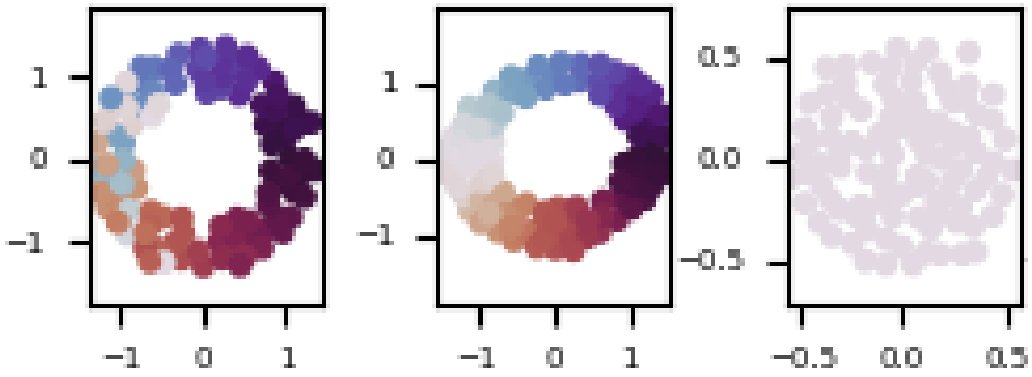

Figura 17:De izquierda a derecha: estado asíncrono de fase dinámica (d=6, K=-0.2); estado asíncrono de fase estática (d=6, K=0);estado síncrono (d=6,K=0.2).

El estado asíncrono de fase dinámica se caracteriza por un valor de K negativo y un bajo valor de S. En su mayoría presentan momento lineal muy bajo, son estados giratorios respecto al centro de masas. Cabe destacar que para K=-1, el anillo interior desaparece, esto es debido al aumento de la constante de acoplo, que para valores negativos tiende un mayor desorden en la dinámica de fases y en consecuencia en la espacial.

El estado asíncrono de fase estática presenta altos valores de S debido a la disposición de las partículas en forma de anillo y su correlación con las fases.

El estado síncrono se caracteriza por converger todos los elementos del conjunto a la misma fase, presenta altos valores de S, tiene una alta energía cinética, pero momento total nulo.

Las estructuras anulares son características de estos sistemas; en nuestro caso se presentan para valores de K desde cero hasta uno. Para estudiar la extensión espacial de nuestros swarmalators, definimos la constante W.

W indica la diferencia entre la distancia radial desde la partícula más cercana al centro de masas hasta la partícula más alejada del centro de masas. Se adjunta una esquemática visual de este parámetro.

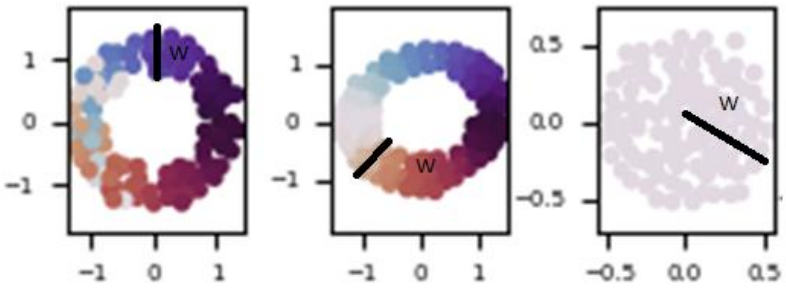

Figura 18:Esquemático del parámetro W.

El parámetro expuesto nos permite estimar la extensión espacial de la estructura.

Para los estados asíncronos W es una estimación de la estructura anular; por otra parte, para los estados síncronos es una estimación del radio de la estructura.

Los estados anulares son característicos de nuestro sistema; a continuación, mostramos las curvas de anchura del estado anular frente a la constante d para diferentes valores de la constante de acoplo K.

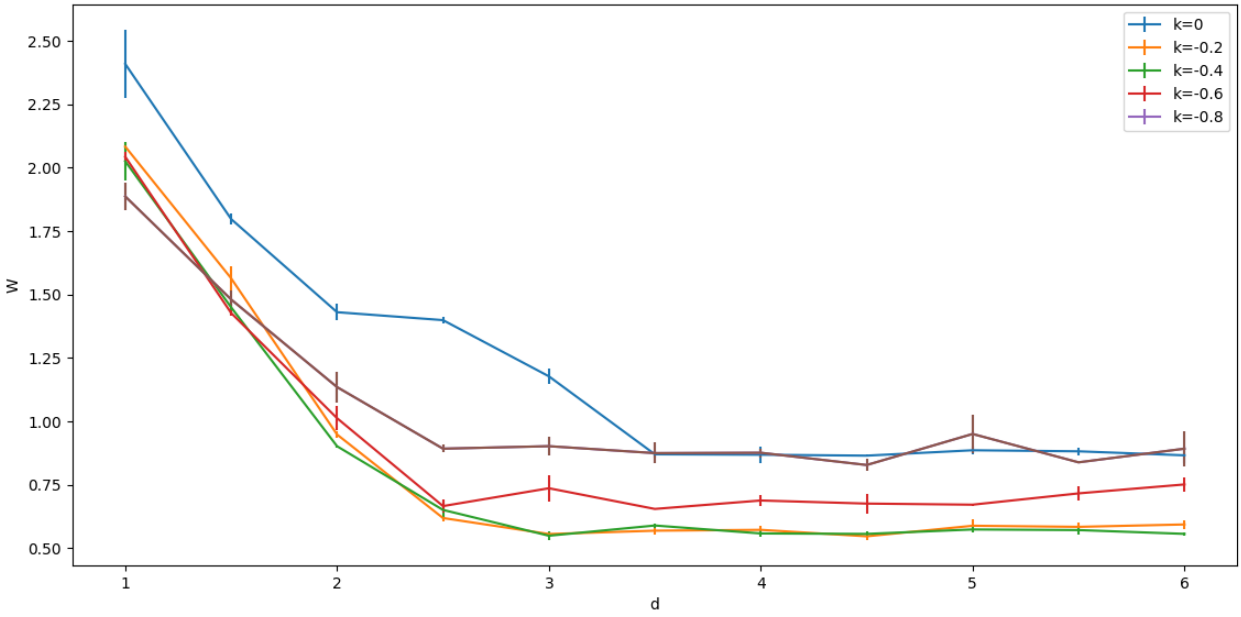

Figura 19:Espesores de la membrana, cada color indica un valor de K.

Están representados los valores de K ∈ [-0.8,0], cada uno de un color. El significado físico es la distancia desde el punto más cercano al centro de masas hasta él más alejado consiguiendo así estimar el espesor.

Se observa que, a medida que aumentamos el valor de la constante d encargada de activar el potencial repulsivo a cortas distancias, el espesor de la membrana disminuye haciéndose constante a un cierto valor. Por otra parte, elementos con un K más negativo presentan un espesor mayor y pueden ser caracterizados por el espesor de saturación a altas d.

A continuación presentamos el mapa de calor completo para este parámetro. Nos permite distinguir entre diferentes estados anulares.

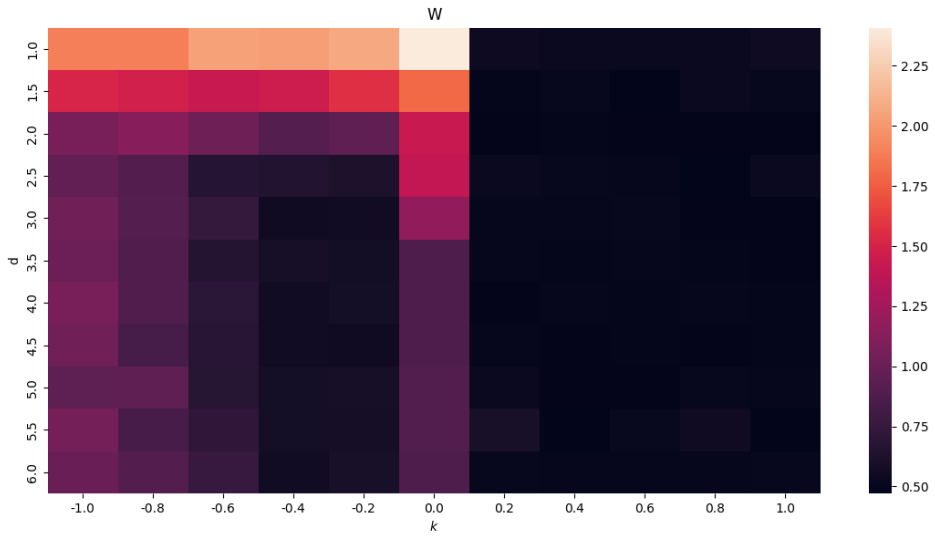

Figura 20:Mapa de calor de la constante W. Para valores de $K \in [-1,0]$ *estima el espesor del disco; para* $K \in (0,1]$ *estima el radio de la estructura.*

Se puede comprobar que el espesor de las estructuras anulares W del sistema es mayor a medida que disminuimos los valores del parámetro K y del parámetro d. Como regla general para los estados síncronos W es el radio de la estructura, como vemos es prácticamente constante.

4 Conclusiones

En definitiva, se ha profundizado en el estudio de los sistemas autoorganizados tipo “swarmalators” a través de varios modelos para obtener una visión general del campo en cuestión. Se ha propuesto un modelo propio que modifica el modelo de O’keeffe[10] al reducir la función de interacción repulsiva a cortas distancias, y al igual que en los modelos estudiados, en el nuestro se observan propiedades y estructuras emergentes. Sin embargo, a diferencia de los demás

modelos, nuestra función repulsiva es discontinua en contraste con la gaussiana propuesta en el estudio [14] de Jiménez-Morales, y restringe de forma discreta el número de partículas que intervienen en el término repulsivo.

Se ha realizado un análisis de las propiedades emergentes en función de los valores de los parámetros que rigen nuestro modelo . Como resultado se han obtenido estructuras anulares que recuerdan capas lipídicas propias de sistemas biológicos. Además se han obtenido estados con alta sincronización a partir de la aleatoriedad, claro ejemplo de comportamiento colectivo en sistemas sociales.

Se ha puesto de manifiesto la importancia de los métodos numéricos para la simulación de los sistemas autoorganizados y se ha usado el lenguaje de programación Python como lenguaje de alto nivel. Este tipo de lenguaje permite una integración rápida de las ecuaciones de evolución que implican la resolución de problemas con alto número de variables.

Finalmente, durante el desarrollo del programa usado para la simulación de los sistemas tratados, han sido exploradas las capacidades de cómputo y uso de memoria a la hora de simular fenómenos de tal magnitud.

5 Anexo: Código fuente en Python

5.1 Herramienta principal para la simulación de un estado

```
1. import numpy as np
2. from matplotlib import pyplot as plt
3. from scipy import integrate
4. import time
5. import swarmalator_tanh_func
6. import numba
7.
8. # Simulation params swarmalators acoplados
9. inicio = time.time()
10.
11.     n = 100
12.     phase_0 = 2 * np.pi * np.random.rand(n)
13.     x_0 = 10 * np.random.rand(n)
14.     y_0 = 10 * np.random.rand(n)
15.     initial = np.concatenate((phase_0, x_0, y_0), axis=0)
16.     "args=(J,delta,k,w,n)"
17.     J = 1
18.     delta = 1.5
19.     K =0.2
20.     w = 0 * np.random.rand(n)
21.
22.     args = (J, delta, K, w, n)
23.
24.     # Integrator params
25.     Tf = float(200)
26.     t_span = (0.0, Tf)
27.     t_eval = np.linspace(0, Tf, 200000)
28.     max_step = 0.1
29.
30.     # Integrator module uses RK45
31.
32.     INT = integrate.solve_ivp(method='RK45',
33.                               fun=swarmalator_tanh_func.swarmala
   tor, y0=initial, t_span=t_span
34.                               , args=args, dense_output=True,t_e
   val=t_eval)
35.
36.     # Extracción de las variables
37.     """variable_out_t, cada  columna es el vector de estados del
   sistema de partículas para cada una de las variables
38.     la fila hace refrencia al indice de la particula, los
   indices de laa columnas los relacionan con el tiempo"""
```

```
39.
40.     sol = INT.y
41.     phi_out_t = sol[0:n]
42.     x_out_t = sol[n:2 * n]
43.     y_out_t = sol[2 * n:3 * n]
44.     t = INT.t
45.
46.
47.     #  Funciones auxiliares en complejos
48.     def to_complex(phase):
49.         tau = complex(np.cos(phase), np.sin(phase))
50.         return tau
51.
52.
53.     to_complex_vectorice = np.vectorize(to_complex)
54.
55.     phi_out_complex = to_complex_vectorice(phi_out_t)
56.
57.     # Contenedores módulo de analisis
58.     S = np.zeros(len(t), dtype=complex)
59.     R = np.zeros(len(t), dtype=complex)
60.     T_kin = np.zeros(len(t))
61.     p = np.zeros(len(t))
62.
63.     x_cm = np.zeros(len(t))
64.     y_cm = np.zeros(len(t))
65.     # Módulo de análisis
66.     for k in range(len(t)):
67.         # calculo del centro de masas
68.         x_cm = sum(x_out_t[:, k]) / n
69.         y_cm = sum(y_out_t[:, k]) / n
70.         x_pos = x_out_t[:, k] - x_cm
71.         y_pos = y_out_t[:, k] - y_cm
72.         # calculo de s
73.         x = np.arctan(y_pos / x_pos)
74.
75.         Rmas = sum(to_complex_vectorice(phi_out_t[:, k] + x))
76.         Rmin = sum(to_complex_vectorice(phi_out_t[:, k] - x))
77.         S[k] = max(abs(Rmas), abs(Rmin))
78.         R[k] = (1 / n) * sum(phi_out_complex[:, k])
79.         # Calculo de las velocidades y energías cinéticas
80.         imagen = np.concatenate((phi_out_t[:, k], x_out_t[:, k],
    y_out_t[:, k]), axis=0)
81.         omega = swarmalator_tanh_func.swarmalator(1, imagen, J,
   delta, K, w, n)
82.         vx = omega[n:2 * n]
83.         vy = omega[2 * n:3 * n]
84.
85.         T_kin[k] = (sum(vx ** 2) + sum(vy ** 2)) * (1 / (2 * n))
86.         p[k] = sum(vx + vy) * (1 / n)
87.
88.     S = (1 / n) * abs(S)
```

```
89.      R = abs(R)
90.
91.      plt.figure('Estado final')
92.      x = x_out_t[:, -1]
93.      y = y_out_t[:, -1]
94.      phi = np.sin(phi_out_t[:, -1])
95.      plt.scatter(x, y, c=phi, s=50, cmap="plasma")
96.      plt.axis('equal')
97.      fin = time.time()
98.      time = fin - inicio
99.      print(time)
```

5.2 Función auxiliar para el integrador

```
1. import numpy as np
2. import numba
3.
4. __author__ = "Fabian Hormigo Pereila"
5. __copyright__ = "Copyright (C) 2022 Minervo"
6. __license__ = "Creative Commons (by-nc)"
7.
8.
9. @numba.jit(nopython=True,cache=True )
10.      def swarmalator_matrix_element_atractivo(x_j, x_i, y_j, y_i,
    phi_j, phi_i, J, delta):
11.          """Calcula el elemento de matriz para el modelo de
   swarmalator, potencial atractivo
12.          Devuelve el elemento de matriz para cada grado de
   libertad,necesaria para evaluar cada componente en el integrador
13.          scipy.integrate.solve_ivp"""
14.          r_ij = ((x_i - x_j) ** 2 + (y_i - y_j) ** 2) ** 0.5
15.
16.          auxiliar_element = (1 + J * np.cos(phi_j - phi_i)) /
   r_ij
17.          # componente x
18.          elementx = (x_j - x_i) * auxiliar_element
19.
20.          # componente y
21.          elementy = (y_j - y_i) * auxiliar_element
22.
23.          # componente phi
24.          elementphi = np.sin(phi_j - phi_i) / r_ij
25.          return [elementphi, elementx, elementy]
26.
27.
28.
29.      @numba.jit(nopython=True,cache=True )
30.      def swarmalator_matrix_element_repulsivo(x_j, x_i, y_j, y_i,
    phi_j, phi_i, J, delta):
```

```
        """Calcula el elemento de matriz para el modelo de
  swarmalator con potencial repulsivo
        Devuelve el elemento de matriz para cada grado de
  libertad,necesaria para evaluar cada componente en el integrador
        scipy.integrate.solve_ivp"""
        r_ij = ((x_i - x_j) ** 2 + (y_i - y_j) ** 2) ** 0.5
        if r_ij < delta:


            d =1

            # componente x
            elementx = -(x_j - x_i) * ((1 / r_ij) ** 2)

            # componente y
            elementy = -(y_j - y_i) * ((1 / r_ij) ** 2)

            # componente phi
            elementphi = np.sin(phi_j - phi_i) / r_ij
        else:
            d=0
            elementx = 0
            elementy = 0
            elementphi = 0

        return [elementphi, elementx, elementy,d]



    def swarmalator(t, imagen, J, delta, k, w, n):
        """Funcion introducida en el integrador
  scipy.integrate.solve_ivp para simulación,
       necesaria para integrar el modelo
            Params:
            t: necesario para el integrador dtype:none
            y:matriz con el cojunto de estados dtype:np array
  dimension:(n,3) estructura (n,[phi,x,y])
            delta: parámetro interno de la simulación,regula el
  potencial repulsivo escalón dtype:float
            J:parámetro interno de la simulación, constante de
  acoplo espacial dtype:float
            K:parámetro interno de la simulación,constante de
  acoplo fasorial dtype:float
            n:número de osciladores dtype:float
            """
        # Extraccion de las variables
        phi = imagen[0:n]
        x = imagen[n:2 * n]
        y = imagen[2 * n:3 * n]
        # matrices a sobreescribir de salida
        phi_out = np.zeros_like(x)
```

```
76.         x_out = np.zeros_like(x)
77.         y_out = np.zeros_like(x)
78.         for i in range(len(x)):
79.
80.             # Acumuladores auxiliares suma en j
81.             A_a=np.zeros_like(x)
82.             B_a = np.zeros_like(x)
83.             C_a= np.zeros_like(x)
84.             A_r = np.zeros_like(x)
85.             B_r = np.zeros_like(x)
86.             C_r = np.zeros_like(x)
87.             N=np.zeros_like(x)
88.             for j in range(len(x)):
89.                 if i == j:
90.                     [A_a[j], B_a[j], C_a[j]] = [0, 0, 0]
91.                     [A_r[j], B_r[j], C_r[j]] = [0, 0, 0]
92.                 else:
93.                     [A_a[j], B_a[j], C_a[j]] = swarmalator_matri
   x_element_atractivo(x[j], x[i], y[j], y[i], phi[j], phi[i],
94.
                       J, delta)
95.                     [A_r[j], B_r[j], C_r[j], N[j]] = swarmalator
   _matrix_element_repulsivo(x[j], x[i], y[j], y[i], phi[j],
96.
                           phi[i], J, delta)
97.
98.
99.                 # componentes i extraidas
100.            norm = sum(N)+1
101.            phi_out[i] = (k / n) * sum(A_a)
102.            x_out[i] = sum(np.array((1 / n) *
   B_a)+np.array((1/norm)*B_r))
103.            y_out[i] = sum(np.array((1 / n) *
   C_a)+np.array((1/norm)*C_r))
104.
105.            phi_out = phi_out + w
106.            x_out = x_out
107.            y_out = y_out
108.
109.        return np.concatenate((phi_out, x_out, y_out), axis=0)
```

5.3 Script para el barrido de estados variación de: K y d

```
1. import numpy as np
2. from scipy import integrate
3. import swarmalator_tanh_func
4. import pandas as pd
5. import numba
6. np.random.seed(2)
7. n = 100
```

```
8. phase_0 = 2 * np.pi * np.random.rand(n)
9. x_0 = 10 * np.random.rand(n)
10.      y_0 = 10 * np.random.rand(n)
11.      initial = np.concatenate((phase_0, x_0, y_0), axis=0)
12.      "args=(J,delta,k,w,n)"
13.      J = 1
14.      delta = np.linspace(1, 3, 5)
15.      K = np.linspace(-1, 1, 11)
16.      w = 0 * np.random.rand(n)
17.      # Integrator params
18.      Tf = float(200)
19.      t_span = (0.0, Tf)
20.      t_eval = np.linspace(0, Tf, 200000)
21.      rtol = 0.1
22.
23.
24.      #  Funciones auxiliares en complejos
25.
26.      def to_complex(phase):
27.          tau = complex(np.cos(phase), np.sin(phase))
28.          return tau
29.
30.
31.      to_complex_vectorice = np.vectorize(to_complex)
32.
33.
34.      # bucle
35.      # funcion sweep
36.
37.      def sweep_states(arg):
38.          args = tuple(arg)
39.
40.          # Integrator module uses RK45
41.          INT = integrate.solve_ivp(method='RK45',
42.                                    fun=swarmalator_tanh_func.swar
   malator, y0=initial, t_span=t_span
43.                                    , args=args, dense_output=True
   )
44.          # Extracción de las variables
45.          """variable_out_t, cada  columna es el vector de estados
   del sistema de partículas para cada una de las variables
46.          la fila hace refrencia al indice de la particula, los
   indices de laa columnas los relacionan con el tiempo"""
47.          sol = INT.y
48.          phi_out_t = sol[0:n]
49.          x_out_t = sol[n:2 * n]
50.          y_out_t = sol[2 * n:3 * n]
51.          # Contenedores módulo de analisis
52.          phi_out_complex = to_complex_vectorice(phi_out_t)
53.          S = np.zeros(len(INT.t), dtype=complex)
54.          R = np.zeros(len(INT.t), dtype=complex)
55.          T_kin = np.zeros(len(INT.t))
```

```
56.         p = np.zeros(len(INT.t))
57.
58.         x_cm = np.zeros(len(INT.t))
59.         y_cm = np.zeros(len(INT.t))
60.         # Módulo de análisis
61.         for k in range(len(INT.t)):
62.             # calculo del centro de masas
63.             x_cm[k] = sum(x_out_t[:, k]) / n
64.             y_cm[k] = sum(y_out_t[:, k]) / n
65.             x_pos = x_out_t[:, k] - x_cm[k]
66.             y_pos = y_out_t[:, k] - y_cm[k]
67.             # calculo de s
68.             x = np.arctan(y_pos / x_pos)
69.
70.             Rmas = sum(to_complex_vectorice(phi_out_t[:, k] +
   x))
71.             Rmin = sum(to_complex_vectorice(phi_out_t[:, k] -
   x))
72.             S[k] = max(abs(Rmas), abs(Rmin))
73.             R[k] = (1 / n) * sum(phi_out_complex[:, k])
74.             # Calculo de las velocidades y energías cinéticas
75.             imagen = np.concatenate((phi_out_t[:, k], x_out_t[:,
    k], y_out_t[:, k]), axis=0)
76.             omega = swarmalator_tanh_func.swarmalator(1, imagen,
    J, delta[j], K[i], w, n)
77.             vx = omega[n:2 * n]
78.             vy = omega[2 * n:3 * n]
79.
80.             T_kin[k] = (sum(vx ** 2) + sum(vy ** 2)) * (1 / (2 *
   n))
81.             p[k] = sum(vx + vy) * (1 / n)
82.
83.         return [sum((1 /
   n) * abs(S)) / len(INT), sum(abs(R)) / len(INT), sum(p) / len(INT
   ),
84.                 sum(T_kin) / len(INT), sol[:, -1]]
85.
86.
87.     # acumulador bucle
88.     T_medio = {}
89.     R_medio = {}
90.     p_medio = {}
91.     S_medio = {}
92.     States = {}
93.     # bucle extraccion de parámetros
94.     for i in range(len(K)):
95.         for j in range(len(delta)):
96.             arg = [J, delta[j], K[i], w, n]
97.
98.             [S_medio[K[i], delta[j]], R_medio[K[i], delta[j]], p
   _medio[K[i], delta[j]], T_medio[K[i], delta[j]],
99.              States[K[i],delta[j]]] = sweep_states(arg)
```

```
100.
101.     # data
102.     np.save('R.npy', R_medio)
103.     np.save('S.npy', S_medio)
104.     np.save('T.npy', T_medio)
105.     np.save('P.npy', p_medio)
106.     np.save('Simulation.npy', States)
```

5.4 Visualizador para el barrido de estados

```
1. import numpy as np
2. from matplotlib import pyplot as plt
3. import matplotlib
4.
5. # carga de archivos
6. x = np.load('Simulation.npy', allow_pickle=True)
7. x = x.tolist()
8. color_map = np.linspace(-1,1,100)
9. n = 100
10.      delta = np.linspace(1, 3, 5)
11.      K = np.linspace(-1, 1, 11)
12.      fig, ax = plt.subplots(len(delta), len(K))
13.
14.
15.
16.      for i in range(len(delta)):
17.          for j in range(len(K)):
18.              # unpack
19.
20.              sol = x[K[j], delta[i]]
21.              phi_out_t = sol[0:n]
22.              x_out_t = sol[n:2 * n]
23.              y_out_t = sol[2 * n:3 * n]
24.
25.              x_cm = sum(x_out_t) / len(x_out_t)
26.              y_cm = sum(y_out_t) / len(y_out_t)
27.              x_out_t = x_out_t - x_cm
28.              y_out_t = y_out_t - y_cm
29.              [x_min,x_max]=[min(x_out_t)-0.2,max(x_out_t)+0.2]
30.              [y_min, y_max] = [min(y_out_t)-
   0.2, max(y_out_t)+0.2]
31.
32.              ax[i, j].scatter(x_out_t, y_out_t, c=phi_out_t, cmap
   ='twilight',s=10)
33.              ax[i, j].set_xlim(x_min, x_max)
34.              ax[i, j].set_ylim(y_min, y_max)
35.
36.
37.              #ax[i, j].set_xticks([])
38.              #ax[i, j].set_yticks([])
```

```
39.             ax[i,j].tick_params(axis='both',labelsize=5)
40.
41.             ax[i, j].axis('equal')
42.
43.
44.     sm=matplotlib.cm.ScalarMappable(cmap='twilight')
45.     fig.colorbar(sm,values=color_map,ax=ax,orientation='vertical
   ',aspect=100,ticks=[])
```

6 Referencias

[1] A. Etxeberria y L. Bich, «Auto-organización y autopoiesis», *Diccionario Interdisciplinar Austral*, 2017.

[2] S. Tuckett, «p_il_fishschool.jpg (1200×670)», 2009. https://feelthebrain.files.wordpress.com/2017/02/p_il_fishschool.jpg (accedido 11 de abril de 2023).

[3] «Desvelan uno de los secretos de la naturaleza que aún no tiene explicación: qué patrones matemáticos siguen las abejas para fabricar sus perfectos panales - Canal UGR». https://canal.ugr.es/noticia/desvelan-uno-de-los-secretos-de-la-naturaleza-que-aun-no-tiene-explicacion-que-patrones-matematicos-siguen-las-abejas-para-fabricar-sus-perfectos-panales/ (accedido 12 de abril de 2023).

[4] J. Soldevila Estrada, «6233608b6d666.r_d.1903-1229-2278.jpeg (948×465)». https://www.lavanguardia.com/files/image_948_465/files/fp/uploads/2022/03/17/6233608b6d666.r_d.1903-1229-2278.jpeg (accedido 12 de abril de 2023).

[5] M. L. Antequera Gómez, «Biofilm de Bacillus cereus - Sociedad Española de Microbiología». https://www.semicrobiologia.org/imagen-banco/biofilm-de-bacillus-cereus (accedido 11 de abril de 2023).

[6] T. C. Schelling, «Dynamic models of segregation», *J Math Sociol*, vol. 1, n.º 2, 1971, doi: 10.1080/0022250X.1971.9989794.

[7] R. V. Carreon, «estudio teórico y preparación de nanopartículas de plata autoensambladas mediante evaporación térmica sobre líquidos iónicos para aplicaciones plasmónicas», 2021, (Master's thesis)

[8] A. T. Winfree, «Biological rhythms and the behavior of populations of coupled oscillators», *J Theor Biol*, vol. 16, n.º 1, pp. 15-42, 1967, doi: 10.1016/0022-5193(67)90051-3.

[9] Y. Kuramoto, «Self-entrainment of a population of coupled non-linear oscillators BT - International Symposium on Mathematical Problems in Theoretical Physics», en *International Symposium on Mathematical Problems in Theoretical Physics*, 1975.

[10] K. O'Keeffe y C. Bettstetter, «A review of swarmalators and their potential in bio-inspired computing», 2019. doi: 10.1117/12.2518682.

[11] K. P. O'Keeffe, H. Hong, y S. H. Strogatz, «Oscillators that sync and swarm», *Nat Commun*, vol. 8, n.º 1, 2017, doi: 10.1038/s41467-017-01190-3.

[12] H. K. Lee, K. Yeo, y H. Hong, «Collective steady-state patterns of swarmalators with finite-cutoff interaction distance», *Chaos*, vol. 31, n.º 3, 2021, doi: 10.1063/5.0038591.

[13] J. P. Euzéby y J. P. Euzéby, «Bacillus entry in LPSN», *Int J Syst Bacteriol*, vol. 47, n.º 2, pp. 590-2, 1997, doi: 10.1099/00207713-47-2-590.

[14] F. Jiménez-Morales, «Oscillatory behavior in a system of swarmalators with a short-range repulsive interaction», *Phys Rev E*, vol. 101, n.º 6, 2020, doi: 10.1103/PhysRevE.101.062202.

[15] P. Japón, F. Jiménez-Morales, y F. Casares, «Intercellular communication and the organization of simple multicellular animals», *Cells and Development*, vol. 169, 2022, doi: 10.1016/j.cdev.2021.203726.