INSTITUTO TECNOLÓGICO DE BUENOS AIRES – ITBA
ESCUELA DE INGENIERÍA Y GESTIÓN

Simulación de partículas activas con geometría definida por esfero-polígonos

AUTOR/ES:

Vega, Magdalena (Legajo 55206)

Guido, Sebastian (Legajo 54432)

DOCENTES/TITULARES O TUTORES:

Parisi, Daniel Ricárdo

TRABAJO FINAL PRESENTADO PARA LA OBTENCIÓN DEL TITULO DE INGENIERO EN INFORMÁTICA

Lugar: Ciudad Autónoma de Buenos Aires

Fecha: 7 de abril de 2021

Resumen

Se implementó una plataforma para realizar simulaciones de descargas de partículas autopropulsadas de recintos de geometría arbitraria y con partículas con forma de esferopolígonos que también se pueden definir por el usuario.

Como caso de estudio, se analizó el problema de partículas cuadradas atravesando una apertura angosta. Se estudió la influencia de los principales parámetros del modelo (resistencia a la rotación, amplitud del ruido angular, velocidad de autopropulsión de las partículas y ancho de la apertura del recinto) en el caudal de salida y la distribución de los tiempos de salida entre partículas consecutivas. En particular, a partir de la distribución de estos tiempos de evacuación se alinean los resultados con una ley de potencia, lo que concuerda con observaciones experimentales.

Índice

1. Introducción **4**

2. Modelo Matemático **5**
2.1. Modelo de fuerzas sociales . . . 5
2.2. Dinámicas Regidas por el Paso Temporal . . . 6
2.3. Cálculo de Fuerzas . . . 6
2.3.1. Fuerza de autopropulsión . . . 6
2.3.2. Fuerzas de contacto . . . 7
2.3.3. Visualización de Colisión . . . 9
2.4. Cálculo de Propiedades Físicas . . . 11
2.4.1. Centro de Masa . . . 11
2.4.2. Momento de Inercia . . . 13
2.5. Integración Numérica . . . 13

3. Implementación **15**
3.1. Modelado Computacional . . . 15
3.1.1. Modelado de Partículas . . . 15
3.1.2. Modelado de los Targets . . . 18
3.1.3. Modelado de las Paredes . . . 19
3.2. Modelado de Clases . . . 20
3.3. Preparación del Ambiente . . . 22
3.4. Archivos de Entrada . . . 22
3.4.1. Archivo de Partículas . . . 22
3.4.2. Archivo de Paredes . . . 23
3.4.3. Archivo de Targets . . . 23
3.4.4. Archivo de Constantes . . . 25
3.5. Ejecución del Algoritmo . . . 26
3.5.1. Ejecución de la Simulación . . . 26
3.6. Archivos de Visualización . . . 27

4. Simulaciones **28**

5. Resultados **30**
5.1. Variación del S_{D} . . . 30
5.2. Variación de la apertura de la Salida . . . 33
5.3. Variación de η . . . 37
5.4. Variación de la velocidad . . . 42

6. Conclusiones **47**

1. Introducción

La presencia de embotellamientos o aglomeraciones frente a constricciones en lugares de flujo es algo que se da frecuentemente en la naturaleza. Desde bacterias fluyendo, a personas compitiendo para poder evacuar un recinto ante un incidente, el entendimiento y capacidad para simular estos escenarios ha resultado de interés para distintos ambientes de estudio. [1]

Como un modelo más sencillo se han estudiado medios granulares, aquellos construidas por muchas partículas macroscópicas, que interactúan entre ellas mediante fuerzas tangenciales y normales, siendo altamente disipativas al no recibir energía del exterior como se puede ver en [2] modelando las partículas como esferopolígonos o [3] donde se modelan como esferas.

Esto evolucionó al estudio donde las partículas pasaron a ser autopropulsadas y aportaban energía al sistema. En un comienzo los distintos modelos utilizados para simular estas situaciones trabajaban con partículas circulares sin fuerzas complejas que actuaran entre sí [4]. A causa de esto había limitaciones a la hora de representar fehacientemente sistemas compuestos por geometrías de partículas más complejas. Esto traía problemas como discrepancias en los resultados obtenidos por los resultados de los modelos con los que se veían en la realidad.

En las últimas décadas surgieron, en respuesta a esto, modelos basados en fuerzas (también conocidos como modelos Newtonianos), que están puramente regidos por las fuerzas que cada partícula ejerce con el resto de los miembros del sistema. En este prospecto, el trabajo realizado por [5] fue aceptado y utilizado por su capacidad para simular un gran número de participantes y por ser computacionalmente realizable. A partir de estos nuevos modelos se pudieron obtener resultados que se asemejaban muchísimo mas con lo visto en la realidad.

Con el pasar del tiempo y a partir de nuevos estudios experimentales logrados por [6, 7], se pudo ajustar un modelo para operar con partículas con la forma de píldoras o esferopolígonos. Donde cada partícula esta compuesta por un rectángulo con extremos hemisféricos. Ajustando el modelo para poder funcionar con este tipo de geometría lograron llegar a resultados que se conllevaban con lo visto experimentalmente. De esta forma el modelo planteado por [8] es tomado como base para la extensión que este trabajo propone.

En este informe se busca mostrar los resultados de ajustar el modelo propuesto para habilitar la posibilidad de operar con geometrías de cualquier tipo. Donde la descripción de la forma de las mismas este dada por una concatenación de esferocilíndros rígidos.

Eligiéndose una concatenación de esferocilíndros rígidos, ya que para modelar la curvatura que puede tener una imagen resulta muy costosa por la cantidad de vértices que se tienen que tener para poder generarlo, se ha llegado a estudiar formas óptimas de que se genere tal como se explica en [9].

De esta manera se estudian los requerimientos necesarios para poder ejecutar el modelo con las nuevas partículas complejas, en conjunto con el detalle de implementación y resultados de las distintas simulaciones realizadas.

2. Modelo Matemático

2.1. Modelo de fuerzas sociales

Como su nombre lo indica, el social force model hace hincapié en una mezcla de dos fuerzas, una socio-psicológica que le sucede al individuo al observar su salida y querer llegar a ella, pero a la vez intenta mantener una distancia con los que lo rodean. Y una fuerza física, que se explica como el contacto con otros. Si se encuentran dos individuos que colisionan, estos se van a tener que desprender uno del otro para poder continuar su camino. Tal como fue explicado en [10]

Utilizando la ecuación de Newton:

$$\overrightarrow{F}_{\mathrm{i}} = m_{\mathrm{i}} \times \frac{dv_{\mathrm{i}}}{dt} \tag{1}$$

Para plasmarlo de manera analítica, el modelo plantea un modelo macroscópico en donde cada individuo i está sometido a dos fuerzas, una fuerza de contacto con el resto y una fuerza de autopropulsión por lograr alcanzar el objetivo y escapar.

$$m_{\mathrm{i}}\ddot{r}_{\mathrm{i}} = \overrightarrow{F}_{\mathrm{granular}} + \overrightarrow{F}_{\mathrm{autopropulsión}} \tag{2}$$

Cada una de estas fuerzas es descripta de manera individual para poder llegar a un entendimiento más profundo. Explicándose en primer lugar la fuerza de autopropulsión $F_{\mathrm{autopropulsión}}$ se describe como se muestra en la fórmula a continuación. Donde m_{i} es la masa del individuo, v_{i} la velocidad en el instante y v_{d} la velocidad de autopropulsión del individuo. Este vector se encuentra normalizado en dirección del punto más cercano al objetivo que se quiere llegar. T representa el tiempo característico que le llevaría al individuo llegar a v_{d} si no hubiese ningún obstáculo ni individuos en su camino.

En segundo lugar, la fuerza de contacto se puede pensar como la suma de las fuerzas que los individuos mantienen con las paredes y la fuerza de contacto que se generan con las N partículas vecinas. La forma en que se hicieron estos dos cálculos será explicado más adelante.

$$\overrightarrow{F}_{\mathrm{Gi}} = \sum_{j}^{N_{\mathrm{c}}} \overrightarrow{F}_{\mathrm{ij}} + \overrightarrow{F}_{\mathrm{wi}} \tag{3}$$

$$\overrightarrow{F}_{\mathrm{Di}} = m_{\mathrm{i}} \frac{(\overrightarrow{v}_{\mathrm{d}} - \overrightarrow{v}_{\mathrm{i}})}{\tau} \tag{4}$$

Utilizando este modelo, se puede observar que se adapta de manera muy cohesiva con las dinámicas regidas por el paso temporal, que va a ser explicado a la brevedad pero incorpora todas las fuerzas que puede sentir un individuo en un período de tiempo y al utilizar el sistema de N ecuaciones diferenciales acopladas, se puede predecir su próxima ubicación y así llevar a cabo una simulación para llegar a un resultado. Por último, el movimiento rotacional del modelo queda determinado por las ecuaciones de Euler:

$$\tau_{\mathrm{D_i}} + \sum_{j}^{N_{\mathrm{c}}} \overrightarrow{r}_{\mathrm{ij}} \times \overrightarrow{F}_{\mathrm{ij}} = I_{\mathrm{i}}\dot{\omega}_{\mathrm{i}} \tag{5}$$

En la misma se sumariza el torque generado por la autopropulsión $\tau_{\mathrm{D_i}}$ con aquel generado por cada una de las partículas que hayan colisionado con la partícula en ese instante.

2.2. Dinámicas Regidas por el Paso Temporal

Una simulación dirigida por el paso temporal se compone de N partículas que interactúan mediante fuerzas entre ellas. La simulación está regida por una variable temporal, dt, que marca el paso del tiempo. Para poder predecir y aproximar el movimiento de las partículas, se implementa un método de integración numérica de las ecuaciones de movimiento que da la ubicación pasado una cierta cantidad de tiempo.

Al tener partículas en un espacio acotado, es natural que haya interacciones entre ellas, estas pueden ser de largo alcance, durando varios pasos temporales, o de corto alcance, que duran un paso temporal ya que tienen mucha densidad.

2.3. Cálculo de Fuerzas

El último, y quizás más importante, componente que entra en juego para llevar adelante la simulación es el modelo de fuerzas implementado. El mismo se puede dividir en dos categorías principales:

Fuerza de autopropulsión con ruido

Fuerzas de Contacto

2.3.1. Fuerza de autopropulsión

En primer lugar, se describe la autopropulsión que las partículas tienen por querer llegar hacia sus salidas u objetivos. Cada partícula puede tener asociada una velocidad que marca la intensidad con la que quiere llegar a su objetivo y a partir de ella se establece la intensidad que la fuerza aplica sobre la misma.

$$\vec{F}_{\mathrm{Di}} = m_{\mathrm{i}} \frac{\vec{v}_{\mathrm{d}} - \vec{v}_{\mathrm{i}}}{\tau} \tag{6}$$

La dirección hacia la cual las partículas desean moverse surge de tomar su próximo target, hallar el punto más cercano del segmento al centro de masa de la partícula y elegir ese vector (centro de masa a punto más cercano del target) como vector dirección de la fuerza. La magnitud de esta fuerza surge de calcular la resta del vector recién mencionado con la velocidad actual de la partícula, esto logra que si la partícula se encuentra moviéndose de manera marcadamente distinta a como debería, el cambio sea brusco hasta acercarse al movimiento deseado. Adicionalmente, τ resulta ser el tiempo que le tomaría a la partícula llegar a su velocidad deseada si no hubiera obstáculos en el camino.

En segundo lugar, la autopropulsión genera una rotación en la partícula, ya que se busca que la partícula se posicione y mueva 'mirando' a su objetivo. El torque generado por la autopropulsión se describe en estos tres términos.

$$\tau_{\mathrm{D}} = -S_{\mathrm{D}} \Delta\Theta - \beta\dot{\Theta} + R(t) \tag{7}$$

Se cuenta con el parámetro S_{D}, que regula la intensidad con la que la partícula se resiste a no estar orientada con su objetivo. Modificarlo afecta la velocidad con la que llegaría como también la intensidad de la fuerza que genera su rotación.

Luego se incorpora un término viscoso, reflejado por la dependencia con la velocidad angular, cuyo parámetro β fue utilizado como $4{,}5\sqrt{S_{\mathrm{D}}}$ para garantizar un sistema sobre-amortiguado. Se busca además tener este tipo de comportamiento para obtener resultados que se asemejen lo más que se pueda a

la realidad. Donde uno demora un poco más en alcanzar el estado de equilibrio pero una vez alcanzado el mismo no se oscila.

Por último, se debe incorporar un término de ruido al movimiento. Esto se ve evidenciado más adelante en los resultados, pero el sistema tiende a estancarse y bloquearse por completo sin el mismo. Por lo que se incorpora este componente adicional a la fuerza para simular el movimiento que las partículas, humanos o seres, harían para intentar destrabar al bloquearse. El mismo puede tener cualquier naturaleza, mientras no sea uniforme entre las partículas. En este trabajo el mismo se plantea como un ruido sinusoidal:

$$R(t) = \eta \sin \omega t + \phi_{\mathrm{i}} \tag{8}$$

El ruido está descripto por la amplitud, la frecuencia y la fase ϕ. Para lograr lo mencionado previamente respecto a no tener las partículas sincronizadas, la fase es propia de cada partícula. Esto logra que el movimiento oscilatorio de cada una sea propio y al momento de colisionar o trabarse, cada cual empuje individualmente para poder liberarse.

2.3.2. Fuerzas de contacto

Por otro lado, las fuerzas que sienten las partículas pueden provenir de un ente externo. En este modelo las mismas solamente pueden provenir de la colisión con otra(s) partículas o con una pared. Para detectar si dos partículas están colisionando, se recorren todos los ejes de cada partícula y se verifica contra todos los vértices de otra si hay una colisión.

Un punto P puede caer en alguna de las 3 figuras de la imagen 1 . En caso de caer dentro de A o C, se toma el extremo del segmento más cercano como el más cercano. En cambio, si cae en la figura B, se traza una recta perpendicular al segmento y se la hace pasar por el punto P, para obtener entonces la intersección de este nuevo segmento con el original y así poder determinar el punto del segmento más cercano y la distancia.

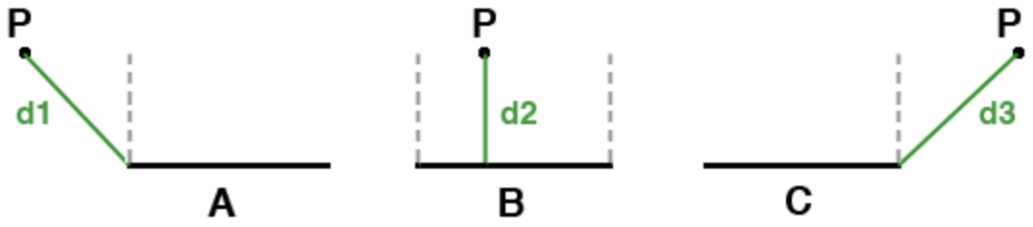


Figura 1: Método para encontrar el punto más cercano a un segmento.

Una vez que se encuentra que hay una colisión, se procede a calcular la fuerza que se aplicará entre cada una de las partículas. Se puede describir entonces a las fuerzas de contacto como:

$$\overrightarrow{F}_{\mathrm{Gi}} = \sum_{j}^{N_{\mathrm{c}}} \overrightarrow{F}_{\mathrm{ij}} + \overrightarrow{F}_{\mathrm{wi}} \tag{9}$$

Donde se tiene que la fuerza total resulta de la suma de las fuerzas entre todas las partículas que están colisionando con la partícula y aquellas generadas por choques con las paredes y la partícula en

cuestión. El modelo de fuerzas que se utiliza para describir las fuerzas normales y tangenciales de contacto es:

$$F^{\mathrm{n}} = -k^{\mathrm{n}}\delta - \gamma^{\mathrm{n}} v^{\mathrm{n}}{}_{\mathrm{rel}} \tag{10}$$

$$F^{\mathrm{t}} = -k^{\mathrm{t}}\xi - \gamma^{\mathrm{t}} v^{\mathrm{t}}{}_{\mathrm{rel}} \tag{11}$$

En cuanto al componente normal de la fuerza de contacto, el mismo se ve compuesto por un componente lineal y uno disipativo, como el visto en la fuerza de autopropulsión. El primer término está dirigido por una constante k_{n} y es directamente proporcional al solapamiento que sufren las dos partículas entre sí d. Para hallar la dimensión del solapamiento es necesario tomar el vértice y eje más cercanos entre las partículas y restarle a la distancia entre ambos la suma de los radios. De ese resultado se obtiene d.

En segundo lugar está el componente disipativo, el cual nuevamente tiene una constante de viscosidad que se opone al movimiento y es directamente proporcional a la velocidad relativa que las dos partículas tienen únicamente en el sentido normal. Esto quiere decir que se debe tomar la velocidad relativa entre las partículas y luego proyectar la misma en el eje perpendicular a aquel que une los dos puntos de contacto.

Cabe aclarar que cuando se habla de velocidad relativa entre partículas se habla de la velocidad relativa entre los puntos de contacto. Es decir que se tiene en cuenta tanto el desplazamiento del centro de masa como la velocidad angular y el desplazamiento lineal que esto genera en los puntos de contacto.

Entonces, para calcular esto en primera instancia hay que tener la velocidad del PC_{A} y la velocidad del PC_{B}. Para hacer esto se hace:

$$VPC_{\mathrm{A}} = V_{\mathrm{A}} + \omega_{\mathrm{A}} \times (PC_{\mathrm{A}} - CM_{\mathrm{A}}) \tag{12}$$

. Esto se interpreta como que a la velocidad de desplazamiento del centro de masa hay que agregarle el movimiento que la rotación genera en el punto de contacto. Análogamente se calcula la velocidad del punto de contacto B y finalmente se restan las dos.

Una vez que se cuenta con la velocidad relativa entre los puntos se realiza la operación descripta en la fórmula y entonces se obtiene el módulo de la fuerza normal.

En cuanto a la fuerza tangencial, la misma surge como un reflejo de la elasticidad entre las superficies que se chocan. Cuenta nuevamente con un término disipativo y uno lineal que se comporta como un resorte.

El término que representa el resorte se puede pensar como la resistencia elástica que existe al desplazar dos superficies. Por ejemplo, cuando se apoya una mano encima de la otra y se desliza una en un sentido, la piel se tiende a estirar siguiendo el movimiento pero oponiéndose al mismo. Cuanto más estira, mayor es la fuerza que se hace para volver. Por esto, cuando dos partículas colisionan y hay un desplazamiento tangencial relativo entre los puntos de contacto, se crea un resorte imaginario que comienza a estirarse y lo hará siempre y cuando las partículas continúan en contacto. Tan pronto como las partículas dejen de colisionar el estiramiento del resorte (y por lo tanto la fuerza que aplica) vuelve a ser nulo.

En segundo lugar nuevamente se cuenta con un término de amortiguamiento proporcional a la velocidad relativa tangencial entre los puntos de contacto, que se haya de manera análoga a la relativa norma mencionada anteriormente pero con sentido tangencial a las superficies que se chocan.

2.3.3. Visualización de Colisión

Para mejor comprender el funcionamiento de los distintos componentes que entran en juego cada vez que se colisiona entre dos partículas se puede ver la siguiente secuencia de imágenes que dan un mejor entendimiento a cómo se implementa este modelo de fuerzas de contacto.

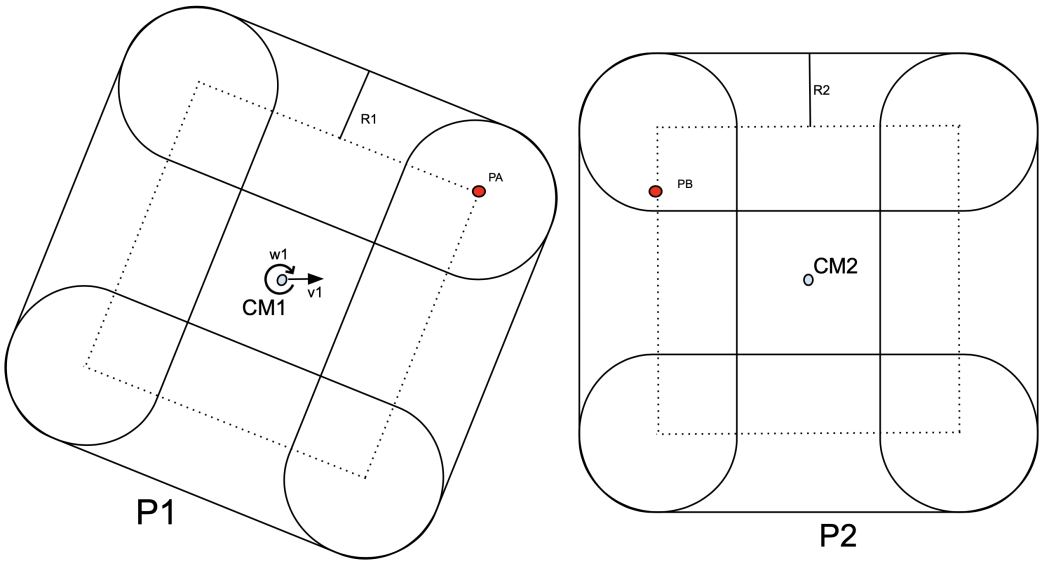
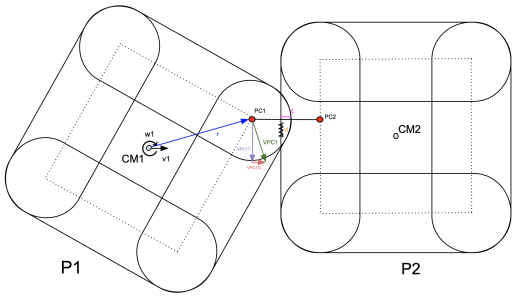


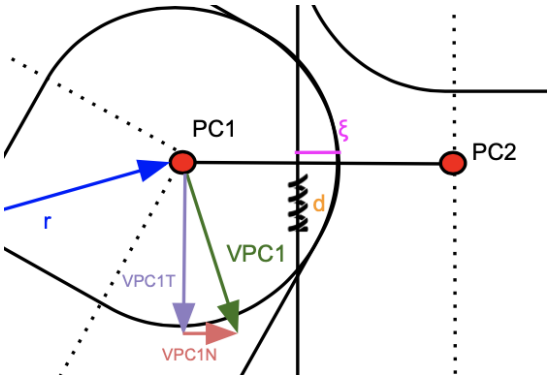
Figura 2: Comienzo de una colisión entre P1 y P2

En primer lugar, viendo la (figura 2) se pueden visualizar dos partículas distintas P_1 y P_2. Cada una de ellas cuenta con sus respectivos 4 esferocilíndros describiendo un cuadrado con sus respectivos radios. Adicionalmente P_1 cuenta con su velocidad lineal v_1 y una velocidad de rotación angular w_1 como se describe en la misma.

Como primer iteración se intenta ver si están colisionando y, dado que la distancia entre los dos puntos más cercanos (P_A y P_B) no es menor a la suma de los radios, no se hay colisión y se continua con la simulación.



(a) Comienza la colisión entre las dos partículas.



(b) Foco en el área de la colisión para visualizar los distintos componentes.

Figura 3: Fotograma de los distintos componentes que participan en una colisión

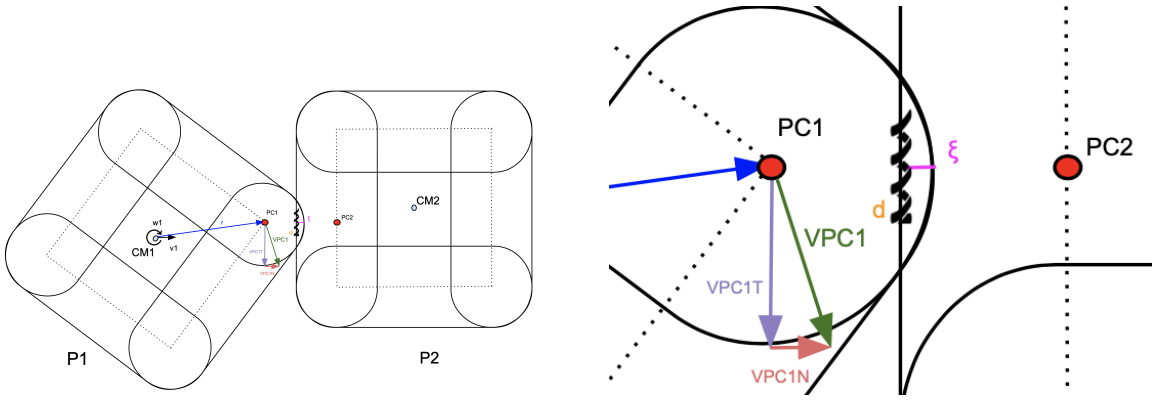
En el siguiente instante de la simulación se llega a un estado donde P_1 y P_2 avanzan a las posiciones mostradas en la (figura 3(a)). Poniendo foco en el area que muestra el roce entre las 2 partículas, como se muestra en la (figura 3(b)) se puede ver que hay un solapamiento entre las dos partículas.

Este solapamiento resulta de la distancia entre los puntos de contacto mas cercanos entre las dos partículas, PC_1 y PC_2. El valor de este solapamiento resulta ξ, visualizado en rosa. El mismo es el solapamiento δ visualizado en (10) y determinará la magnitud de la fuerza normal que separa las partículas.

Como siguiente punto es necesario hallar las velocidades relativas entre los puntos que están colisionando, pero lograr hacer eso es necesario antes encontrar la velocidad de cada uno de los puntos de contacto PC_1 y PC_2. Debido a que P_2 no cuenta con ningún tipo de velocidad, la velocidad de PC_2 también resultará nula. En el caso de PC_1, es necesario tomar en cuenta V_1 y la rotación w_1 de la partícula. Finalmente, siguiendo lo planteado en (12) es posible obtener V_{PC_1}.

Luego, descomponiendo V_{PC_1} en los componentes tangenciales y normales al eje de choque (aquel representado por el vector PC_1 a PC_2), se obtiene el componente tangencial y el normal de la velocidad, que es utilizado en (11) y (10) respectivamente.

Por último, dado que hay un componente no nulo de velocidad relativa tangencial entre las dos partículas, se estira un resorte imaginario representando cierta resistencia al roce entre las dos superficies. El estiramiento del mismo se puede ver como d y la magnitud del mismo es el resultante de multiplicar la velocidad relativa tangencial por el diferencial de tiempo del paso de la simulación.



(a) Continua la colisión entre las 2 partículas.

(b) Foco en el área de interés. El resorte se extiende partiendo del estiramiento adquirido del instante previo.

Figura 4: Continuación de la colisión en el próximo instante

Continuando en el siguiente instante de la simulación, se arriba a la (figura 4). En este caso se repite el mismo proceso desarrollado previamente hasta el momento en el que se debe trabajar con el resorte imaginario entre las dos superficies rozándose. Aquí, al continuar habiendo una velocidad tangencial relativa no nula, el estiramiento del mismo se agregará a lo que se había originado en la iteración previa.

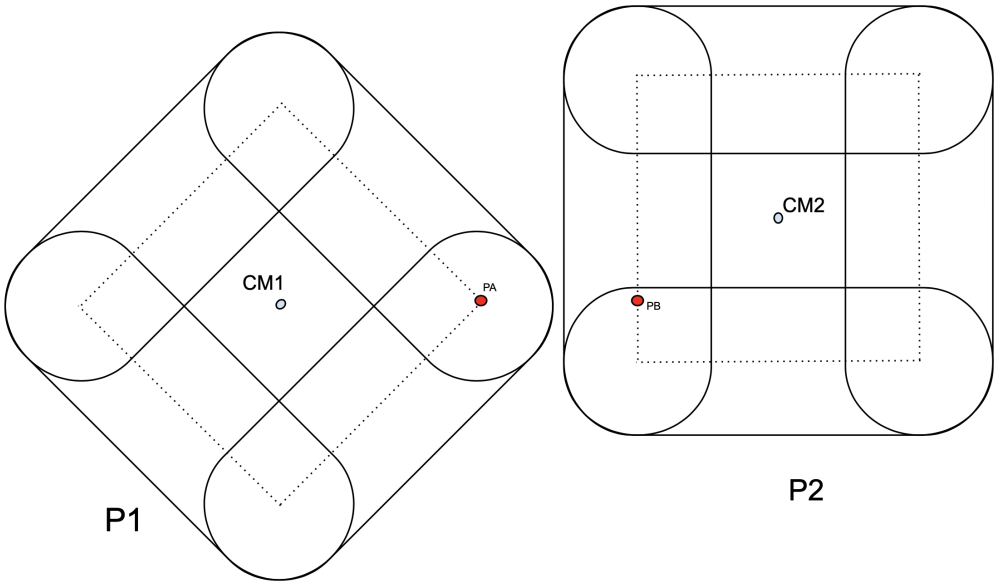


Figura 5: Finalización de la colisión al no haber contacto.

Una vez que las partículas dejan de colisionar, se restituye el estiramiendo del resorte entre las mismas a 0 y las fuerzas que se ejercen desaparece.

2.4. Cálculo de Propiedades Físicas

Para lograr simular fehacientemente la interacción entre las distintas partículas, es necesario poder determinar el centro de masa y momento de inercia de las mismas. Estas dos características son fundamentales en el cálculo y predicción de cómo se manifiesta el movimiento de traslación y rotacional de un objeto.

2.4.1. Centro de Masa

En primer lugar se calcula el centro de masa de la partícula. El mismo tiene en cuenta la distribución de la masa sobre toda su área. En este trabajo se considera que la distribución es homogénea, por lo que cualquier punto dentro del cuerpo aportará la misma cantidad de masa. Para encontrar las coordenadas del centro de masa es necesario aplicar la fórmula de:

$$x_{\mathrm{cm}} = \frac{\int_0^M x dm}{M} \tag{13}$$

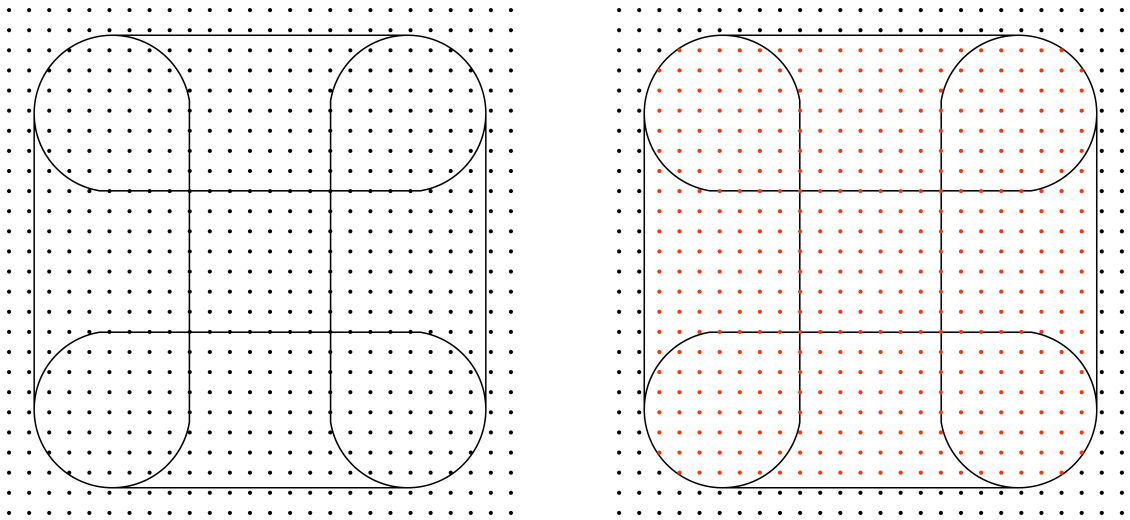
pero al trabajar con partículas descriptas por segmentos se decide discretizar el proceso en una sumatoria de la forma:

$$x_{\mathrm{cm}} = \frac{\sum_{i=1}^{N} m_{\mathrm{i}} x_{\mathrm{i}}}{M} \tag{14}$$

Para computar estos valores es necesario discretizar el espacio en una grilla de tamaño muy pequeño 0.001 que contenga los extremos cartesianos de la geometría en el trabajo. En segundo lugar se itera para todos los puntos de esa grilla que estén dentro de la geometría y se los suma para obtener el valor final.

Para un mejor entendimiento del funcionamiento de este algoritmo se puede visualizar la (figura 6). En la misma se puede apreciar como se proyecta la grilla de puntos sobre todo el área conteniendo

a la geometría (figura 6(a)). Luego, punto por punto se averigua si el mismo forma parte del interior de la geometría y finalmente se suman las coordenadas de todos los puntos dentro de la figura (aquellos representados de color rojo en la (figura 6(b))) y se procede a utilizar cada uno de los mismos como si fuera un diferencial de masa y se promedian sus posiciones.



(a) Grilla de puntos proyectada sobre la geometría. **(b)** Puntos dentro de la geometría seleccionados.

Figura 6: Cálculo de propiedades físicas mediante la suma de puntos internos a la geometría en una grilla.

La consulta por pertenencia de un punto al área de la geometría se realiza trazando una línea horizontal partiendo del punto que cruce por completo la grilla y que pase por el punto en cuestión. Luego se contabilizan la cantidad de veces que esa línea cruza los ejes de la geometría o si se encuentra a una distancia menor a su radio de alguno de los bordes (ver figura 7)

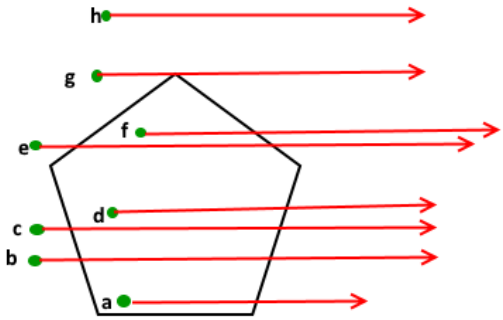


Figura 7: Método para calcular si un punto yace dentro de la geometría descripta.

Basándose en [11], se considera que un punto dado pertenece al área de la geometría si alguna de las siguientes condiciones es verdadera:

- El punto se encuentra sobre un eje
- El número de ejes que la linea horizontal proyectada cruza es impar.

Una vez calculado el centro de masa, se puede predecir el eje de rotación de las partículas y se puede proceder con la siguiente cualidad física.

2.4.2. Momento de Inercia

Para poder predecir nuevamente la forma en la cual el objeto adquirirá movimiento rotacional es necesario conocer su propiedad de centro de inercia. Nuevamente la forma de calcular dicha propiedad resulta de la integral:

$$I = \int_m r^2 dm \tag{15}$$

Para facilitar los cálculos se procede a aplicar el mismo proceso de simplificación que se hizo al hallar el centro de masa. Se discretiza el espacio en una grilla de puntos que contenga como límites a todos los vértices de la partícula, se recorre dicha matriz sumando y consultando para cada punto si el mimo está en el interior de la partícula. La fórmula final sobre la que se termina trabajando es :

$$I = \sum m_{\mathrm{i}} {r_{\mathrm{i}}}^2 \tag{16}$$

Una vez finalizado el cálculo de esta propiedad ya se poseen todos los valores necesarios para representar fehacientemente el comportamiento y reacción de la misma ante choques y fuerzas aplicadas.

2.5. Integración Numérica

Para poder realizar la simulación, se tiene que encontrar una forma de actualizar las posiciones de las partículas a medida que pasa el tiempo. Cada una de ellas experimenta fuerzas como las del social force model que fue explicado anteriormente, para ello se discretizan las ecuaciones de movimiento utilizando su desarrollo de Taylor.

$$r_{\mathrm{i}}(t+\Delta t) = r_{\mathrm{i}}(t) + \Delta t v_{\mathrm{i}}(t) + \frac{\Delta t^2}{2m_{\mathrm{i}}} f_{\mathrm{i}}(t) + \frac{\Delta t^3}{3!} \dddot{r}_{\mathrm{i}}(t) + \mathcal{O}(\Delta t^4) \tag{17}$$

$$v_{\mathrm{i}}(t+\Delta t) = v_{\mathrm{i}}(t) + \frac{\Delta t}{m_{\mathrm{i}}} f_{\mathrm{i}} + \frac{\Delta t^2}{2} f_{\mathrm{i}}(t) + \frac{\Delta t^2}{2} \ddot{v}_{\mathrm{i}}(t) + \frac{\Delta t^3}{3!} \dddot{v}_{\mathrm{i}}(t) + \mathcal{O}(\Delta t^4) \tag{18}$$

Basándose en este principio, surgen diferentes modelos que aproximan la próxima posición y/o velocidad de un elemento dado una suma ΔT. Cada uno puede variar en dificultad de ejecución, dependiendo el grado de precisión que se necesite, se elige el que más se indique al problema.

Se tomaron en consideración dos algoritmos, uno para poder calcular el valor del centro de masa en la partícula un instante antes de que empiece la simulación y otro para poder utilizar como método de integración durante la simulación.

Para poder calcular el valor del centro de masa un instante antes de que empiece la simulación, se utilizó el método de euler. Este es el más sencillo de implementar, pero el menos preciso. Se seleccionó ya que no es necesario tener tanta precisión, la única razón por la que se necesita que haya un valor es que una partícula vaya desde (0,0) hacia su posición inicial.

$$r_{\mathrm{i}}(t+\Delta t) = r_{\mathrm{i}}(t) + \Delta t v_{\mathrm{i}}(t) + \frac{\Delta t^2}{2m_{\mathrm{i}}} f_{\mathrm{i}}(t) + \mathcal{O}(\Delta t^3) \tag{19}$$

$$v_{\mathrm{i}}(t+\Delta t) = v_{\mathrm{i}}(t) + \frac{\Delta t}{m_{\mathrm{i}}} f_{\mathrm{i}}(t) + \mathcal{O}(\Delta t^2) \tag{20}$$

Luego para utilizar durante toda la simulación, se eligió el algoritmo de Verlet como método de integración [12], ya que es sencillo de implementar, no requiere cómputo y no es necesario computar las velocidades para obtener las trayectorias.

Para calcular las ecuaciones de verlet, se utiliza el desarrollo de Taylor para Δt y $-\Delta t$ y se suman haciendo (21) + (22) obteniendo (23) Despejando en un caso posición en $t + \Delta t$ y en otro caso v_{t} haciendo (24) - (25) obteniendo (26).

$$r_{\mathrm{i}}(t+\Delta t) = r_{\mathrm{i}}(t) + \Delta t v_{\mathrm{i}}(t) + \frac{\Delta t^2}{2m_{\mathrm{i}}} f_{\mathrm{i}}(t) + \frac{\Delta t^3}{3!} \dddot{r}_{\mathrm{i}}(t)) + \mathcal{O}(\Delta t^4) \tag{21}$$

$$r_{\mathrm{i}}(t-\Delta t) = r_{\mathrm{i}}(t) - \Delta t v_{\mathrm{i}}(t) + \frac{\Delta t^2}{2m_{\mathrm{i}}} f_{\mathrm{i}}(t) - \frac{\Delta t^3}{3!} \dddot{r}_{\mathrm{i}}(t)) + \mathcal{O}(\Delta t^4) \tag{22}$$

$$r_{\mathrm{i}}(t+\Delta t) = 2r_{\mathrm{i}}(t) - r_{\mathrm{i}}(t-\Delta t) + \frac{\Delta t^2}{m_{\mathrm{i}}} f_{\mathrm{i}}(t) + \mathcal{O}(\Delta t^4) \tag{23}$$

$$r_{\mathrm{i}}(t+\Delta t) = r_{\mathrm{i}}(t) + \Delta t v_{\mathrm{i}}(t) + \frac{\Delta t^2}{2m_{\mathrm{i}}} f_{\mathrm{i}}(t) + \frac{\Delta t^3}{3!} \dddot{r}_{\mathrm{i}}(t)) + \mathcal{O}(\Delta t^4) \tag{24}$$

$$r_{\mathrm{i}}(t-\Delta t) = r_{\mathrm{i}}(t) - \Delta t v_{\mathrm{i}}(t) + \frac{\Delta t^2}{2m_{\mathrm{i}}} f_{\mathrm{i}}(t) - \frac{\Delta t^3}{3!} \dddot{r}_{\mathrm{i}}(t)) + \mathcal{O}(\Delta t^4) \tag{25}$$

$$v_{\mathrm{i}}(t) = \frac{r_{\mathrm{i}}(t+\Delta t) - r_{\mathrm{i}}(t-\Delta t)}{2\Delta t} + \mathcal{O}(\Delta t^3) \tag{26}$$

3. Implementación

3.1. Modelado Computacional

El modelado de la evacuación de esferopolígonos genéricos de un recinto fue implementado de tal manera que todos los parámetros para ingresar al sistema sean lo más genéricos posibles, esto facilita su diversificación y será de gran ayuda al comenzar con ejecuciones y poder llegar a resultados.

El programa toma como punto de partida 4 archivos de configuración que ayudan a modelar todas las partes del algoritmo, cada uno será explicado más en detalle a continuación.

Estos modelados son, cada una de las partículas, los elementos que generan que el espacio sea acotado, es decir las paredes, el objetivo de cada una de estas partículas para poder ser libre, llamado targets, como también los parámetros que forman parte de todo el sistema.

3.1.1. Modelado de Partículas

Al comenzar a plantear una partícula genérica, no era simplemente usar geometrías existentes, ya que habría que hacer cada una por separada y no se podrían agregar nuevas, sino que se tenían que modelar de alguna manera que todo fuera aceptado. Luego de varios intentos se planteó, como puntapié inicial, una serie de puntos para poder generar la partícula. Por razones de simplicidad la masa está homogéneamente distribuida y los puntos tienen un orden de creación, en este caso de izquierda a derecha, con una dirección apuntando al norte. Tal como se puede observar en las figuras ejemplificadas a continuación donde se identifican 3 partículas, una estrella, un pájaro y una flecha (figura 8). Todas estás partículas tenderían a orientarse como estas lo hacen hacia el norte para poder avanzar.

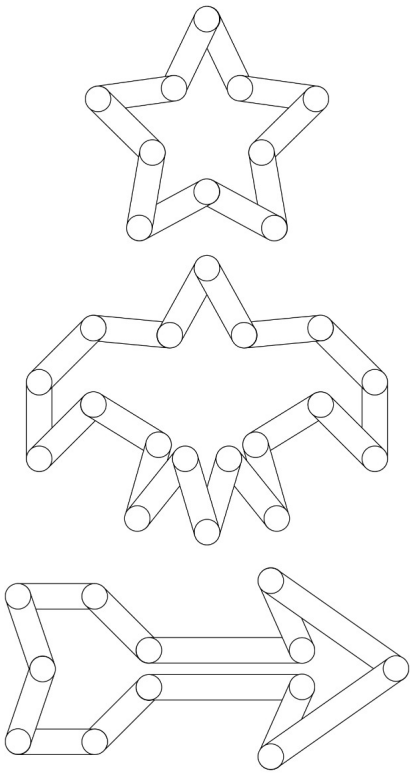


Figura 8: Partículas con diferentes geometrías

Este modelo fue el correcto en elegir, pero, sin algunos cambios, genera muchos conflictos al querer desplazar a la partícula y poder calcular sus interacciones.

Para ello se calculó el centro de masa y cada uno de los puntos asignados fueron anclados a este. Esto se realizó teniendo un eje de referencia y reemplazando cada punto por el ángulo que lleva desde el norte donde se encuentra la partícula a la recta que generan el centro de masa al punto y su longitud.

Tal como fue explicado anteriormente el centro de masa es calculado a partir de la desratización del espacio en una grilla de puntos que contenta como límite a todos los vértices de la partícula y se puede visualizar en el código a continuación. El método poligonBounds es el responsable de devolver un arreglo de 4 elementos con los extremos que encapsulan la geometría en cuestión. Iterando por todos los polígonos que lo generan y eligiendo aquellos que cumplen con lo que se busca.

```
Point calcularCentroDeMasa(poligon, precision, radius){
    acum = 0;
    xAcum = 0;
    yAcum = 0;
    double[] bounds = poligonBounds(poligon, radius);
    for (i; cada iteracion los extremos cartesianos en el eje x)
        for (j; cada iteracion los extremos cartesianos en el eje y)
            if (esta dentro de la geometria)
                acum++;
                xAcum+=i;
                yAcum+=j;

    double x =xAcum/acum;
```

```
        double y = yAcum/acum;
        new Point(x,y);
    }
```

Una vez hecho este cálculo se necesita generar el objeto de cada una de las partículas como se observa a continuación, por cada partícula que se quiere ingresar en la simulación, se utilizan la lista de puntos ingresada como parámetro inicial en conjunto con una masa y un radio aleatorio, para poder generarla. A su vez, el método calculateAngularPoints utiliza el centro de masa previamente calculado y ancla cada partícula a su eje.

```
    for (int j = 0; j < TOTAL_PARTICLES; j ++) {
        double mass = getRandomUniformMass();
        double radius = getRandomRadius(rMin, rMax);
        Point massCenter = calculateMassCenter(points, precision, radius );
        double inertiaMoment = calculateInertiaMoment(points, massCenter, precision, radius);
        List<AngularPoint> ap = calculateAngularPoints(massCenter, points);
        inertiaMoment *= mass;
        double phase = Math.random() * Math.PI * 2;
        Particle p = new Particle(particles.size(), type, mass, ap, massCenter, 0, radius, desiredVel, new Point(0,0),
                0 , 0, inertiaMoment, phase);
        p.setColor(255,255,255);
        particles.add(p);
        }

    calculateAngularPoints(Point massCenter, Point[] points) {
        List<AngularPoint> ap = new ArrayList<>();
        for (int i = 0; i < points.length; i++) {
            double angle = Utils.getAngle(massCenter, points[i]);
            double length = massCenter.distanceBetween(points[i]);
            ap.add(new AngularPoint(angle, length));
        }
        return ap;
    }
```

Al hacer este paso, no solamente se hace mucho más sencillo desplazar a las partículas, sino también dibujarla en el espacio y poder calcular las fuerzas sociales y de autopropulsión que está experimentaba.

Para poder calcular las fuerzas de contacto, se necesita calcular el momento de inercia como se explicó en la sección anterior, esto ayuda que cada partícula pueda obtener movimiento rotacional y así con cada contacto pueda rotar y no solamente desplazarse de manera lineal. Tal como se muestra en el pseudocódigo a continuación, y como se realizo para calcular el centro de masa, se itera por cada uno de los puntos que se encuentra dentro del polígono.

```
    inertiaMoment(poligon, relative, precision, radius){
        points = 0.0;
        inertia = 0;
        double[] bounds = poligonBounds(poligon, radius);
        double[] bounds = poligonBounds(poligon, radius);
        for (i; cada iteracion los extremos cartesianos en el eje x)
            for (j; cada iteracion los extremos cartesianos en el eje y)
                if (esta dentro de la geometria)
                    points += 1.0;
                    double relX = i - relative.getX();
                    double relY = j - relative.getY();
                    inertia += relX * relX + relY * relY;
        return inertia / points;
```

Cada partícula tiene una masa, un radio, una orientación y la velocidad a la que desea moverse.

Cada uno de estos parámetros se necesitan para poder llevar a cabo la simulación de la manera más real posible y van a ser utilizados, ya sea en las fórmulas y cálculos que se explican a lo largo de todo el informe.

Para que la simulación sea lo más real posible, y teniendo en cuenta el modelo de integración numérica elegido, no tenía sentido dejar la posición del centro de masa anterior igual a 0, entonces, basándonos en la teoría se decidió que cuando se crean las partículas por primera vez se cargue la posición anterior utilizando el método de euler pero con un Δ de $-\Delta T$; tal y como como se explicó anteriormente y se ve en el pseudocódigo a continuación.

```
eulerPosition(double delta) {
    nextX = getMassCenter().getX() + delta * getVel().getX() + (delta * delta * getForce().getX()) / (2 * getMass
        ());
    nextY = getMassCenter().getY() + delta * getVel().getY() + (delta * delta * getForce().getY()) / (2 * getMass
        ());
    new Point(nextX, nextY);
```

3.1.2. Modelado de los Targets

Cada partícula tiene que tener uno o más objetivos que tiene que atravesar para poder evacuar, ya sea que el recinto tuviese varios lugares para salir, uno solo, o solamente se las quisiera mover a un lado por alguna razón y luego a otros. Se comenzó con pensar cómo se modelaría el objetivo, ya que puede ser una apertura en un recinto, de tamaño desconocido o solamente un punto en el espacio donde se tiene que llegar para por ejemplo, agarrar algo. Las posibilidades son enormes.

Finalmente se decidió modelar los targets como un segmento con la orientación que uno quiera, ya sea horizontal, vertical o en ángulo. Con solamente atravesar algún punto de la superficie del segmento, se tomaría como alcanzado por la partícula y se pasaría al próximo. En cada momento de la simulación se calcula el punto más cercano entre la orientación de la partícula y el segmento, como se muestra en el pseudocódigo a continuación para poder ser más precisos y calcular la fuerza de autopropulsión de manera más eficiente, utilizando el método planteado por [13].

Siendo (x1, y1), (x2, y2) los dos extremos del segmento ab, y (x3, y3) las coordenadas del punto p.

```
public static Point completeClosestPoint(Segment ab, Point p) {
        Point lineVec = new Point(x2-x1, y2-y1);
        Point pVec = new Point(x3-x1, y3-y1);

        double lineLength = lineVec.module();
        Point lineUnitVec = lineVec.get_unit_vector();

        Point pointVecScaled = pVec.multiplyBy(1/lineLength);

        double t = lineUnitVec.dotProduct(pointVecScaled);

        if (t<0.0)
            t = 0.0;
        else if (t>1.0) {
            t=1.0;

        Point nearest = new Point(lineVec.getX() * t, lineVec.getY() * t);
        nearest.add(ab.getP1());
        return nearest;
    }
```

Ya que una partícula puede tener más de un objetivo, se necesitaba modelar realmente cuando una partícula había logrado evacuar el recinto o no, para ello se agregó una variable que identifica si un objetivo es final. Esta propiedad se ingresa en los archivos de configuración que serán explicados más adelante. Se agregó también un color diferente si es final para que al visualizar una simulación puede ser más claro.

Para poder crear un target se utilizó la clase Target que contiene el segmento, si es final y su centro de masa, como se muestra a continuación.

```
Target(Segment segment, boolean end, Point massCenter) {
    this.s = segment;
    this.end = end;
    if (end) setFinal();  else setMiddle();
    place(massCenter);
}
```

La actualización de los targets, parece sencilla pero contiene mucha lógica, ya que todo depende cual es el próximo paso para cada partícula.

Si se llegó a un target pero este no es el final entonces solamente se actualiza la dirección de autopropulsión al próximo objetivo. En cambio, si este target es el último, entonces se imprime el tiempo en que esta partícula llegó a su objetivo y se tienen dos opciones basándose en las constantes. Si se elige que las partículas vuelvan a ingresar al recinto entonces se hace un reseteo de la partícula, sus targets, su posición, sus fuerzas, su velocidad, para que en la próxima iteración pueda interactuar de nuevo. Si en cambio se eligió que no vuelva a ingresar, entonces se la remueve de las partículas del arreglo y se continúa.

```
if (p.objectiveReached()){
    if (! p.lastTarget()){
        p.nextTarget();
    } else {
        if (! continuous) {
            particles.remove(p);
        } else {
            p.resetTargets();
            p.resetPosition();
        }
    }
}
```

3.1.3. Modelado de las Paredes

Un componente esencial en las simulaciones son las paredes, es decir los bloques inamovibles que evitan que las partículas encuentren otro lugar para irse. Se pensó en modelarlas como un componente diferente, pero se notaron muchas coincidencias con las partículas matemáticamente hablando, son obstáculos en el camino que contienen una masa, un centro de masa, un radio, una serie de puntos en el espacio, pero se diferencian en que no se van a mover, sin importar la fuerza que se les aplique y no están sujetas a formas genéricas, sino que se componen de un segmento.

Este modelado facilitó la interacción física entre partículas, ya que las colisiones se podían detectar de la misma manera, la única diferencia se notaba en la fuerza que se generaba si existe alguna interacción, esto será explicado más adelante.

Las paredes fueron implementadas con la misma clase Particle pero con la única diferencia que con-

tiene una variable que especifica que es una pared y cada vez que se tienen que calcular las fuerzas, solamente se agregan fuerzas a aquellas que no son paredes.

```
isWall = True
```

3.2. Modelado de Clases

El algoritmo y su simulación fue realizado en JAVA8 utilizando maven como manejador de paquetes y para poder llevarlo a cabo se generaron 21 clases, donde cada una de ellas cumple una función especifica. El uml de las clases, sus métodos y variables se puede visualizar en la (figuras 9, 10, 11).

Para poder leer cuales son los archivos de entrada se utiliza la clase Parameter que contiene los nombres pasados por línea de comando luego para leer estos archivos y como se componen se implementaron las clases WallType, TargetType, ConstantType y ParticleType que mapean el formato en que se van a ingresar los parámetros, a su vez Type maneja los tipos de archivos que pueden ingresar, en este caso serian partículas, paredes, targets y constantes. Como también cada uno de los elementos que se generan siendo Particle, Target y Data.

Modelar cada una de las diferentes geometrías implica hacer uso de puntos angulares, segmentos, puntos y quaterniones, como también todas las operaciones que se puedan generar entre ellos, existiendo así AngularPoint, Point, Segment y Quaternion.

Por ultimo, y como punto más importante, está como se maneja la simulación y todo lo que conlleva. Se tienen Input y Output que manejan los archivos de entrada y salida. Simulator que se encarga de llevar a cabo la simulación. Collider maneja las colisiones entre las partículas, como también Utils donde se guardan todos los métodos que calculan propiedades físicas y cálculos genéricos y Main que es el punto de entrada.

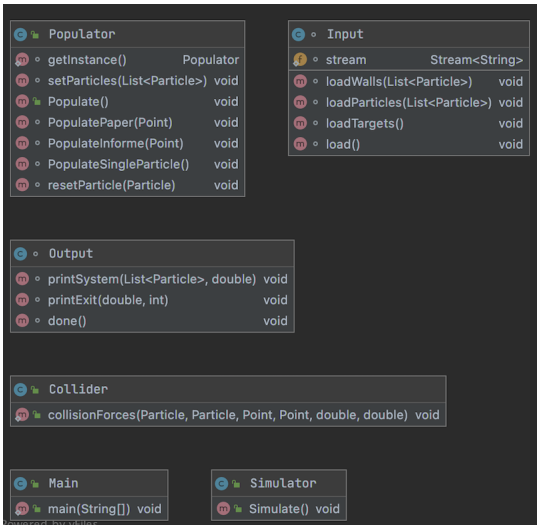


Figura 9: Uml de clases principales con sus métodos

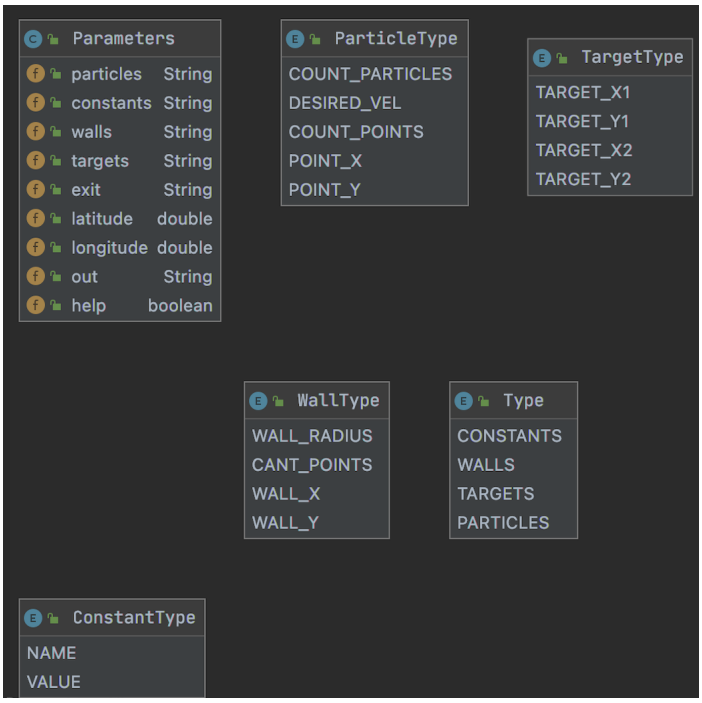


Figura 10: Uml de las clases encargadas de manejar entrada con sus variables

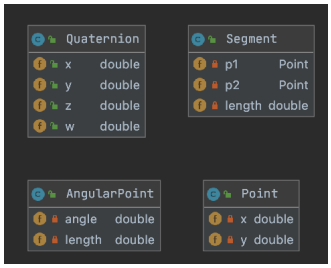


Figura 11: Uml de las clases encargadas de manejar geometrías con sus variables

3.3. Preparación del Ambiente

Tomando en consideración todo lo antes explicado, las ecuaciones de verlet, el social force model, modelado de las partículas, paredes y targets y el cálculo de las fuerzas. Se implementó el siguiente algoritmo para llevar a cabo la simulación.

En primer lugar, se leen los archivos de configuración que van a marcar el ambiente, cuáles son los valores de las variables, cuántas partículas van a haber y de que tipo, cual va a ser la distribución de las paredes, y que targets van a existir. Para que el desarrollo sea más sencillo, todas las partículas siguen los mismos objetivos.

Luego, se diagrama cuales van a ser las paredes iniciales, obteniendo el área que componen, y a cada partícula se la crea, se le asigna una posición y orientación aleatoria en el espacio y se les asignan una lista de los objetivos que van a tener que cumplir, esto se hace de esta manera ya que no todas las partículas van a llegar a los objetivos en determinado tiempo.

Y así, una vez que está todo asignado es como comienza la simulación.

```
loadConstants();
readParticleFile();
readTargetFile();
readWallFile();
loadTargets();
loadWalls();
loadParticles();
createExitFiles();
particles = populateParticles();
particles.populateTargets();
simulation.start();
```

3.4. Archivos de Entrada

Como se explico anteriormente se necesito una forma de poder ingresar y poblar el ambiente de una manera sencilla pero que también deje lugar a toda la complejidad que se quiera.

Para poder leer los archivos de entrada se implemento un parser en la clase Input, que lee cada archivo por separado y en base al formato de cada uno de los archivos va creando cada elemento en particular. Se realizó de esta manera para que sea sencillo tanto para el que esta queriendo correr la simulación como para implementar.

3.4.1. Archivo de Partículas

Para poder definir las partículas de cualquier tipo de geometría que van a ser parte de la simulación, se tiene que especificar cuantos tipos de partículas se van a necesitar.

Un tipo de partícula es aquel que comparte una geometría determinada en cuanto a tamaño, estas pueden aparecer en cualquier lugar del espacio que se define con las paredes, se decidió implementarlo de esta manera para no tener que ingresar N veces cada partícula si lo que se quería eran que sean del mismo formato y que pueda ser legible con tan solo ver el archivo.

El formato para cada tipo de partículas es:

```
cantidadDeParticulasDeEseFormato velocidadDeseada cantidadDePuntos x1 y1 ... xn yn
```

Los puntos tienen que estar en un orden determinado, si esto no sucede puede interferir el funcionamiento de la simulación, en el sistema de referencias que se prefiera, uniéndose el primer punto con el último. El último punto ingresado genera un segmento con el primer punto ingresado,dando la libertad de poder generar partículas de cualquier forma y tamaño dependiendo de lo que se necesite.

Por ejemplo, para poder generar una partícula en forma de cuadrado de lado 3 x 3 se imprimirán los puntos con velocidad deseada 2 m/s, tal como se muestra en la (figura 12) y se muestra a continuación.

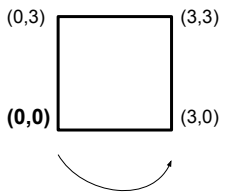


Figura 12: Orden de puntos para partícula en forma de cuadrado

```
1  2  4  0  0  3  0  3  3  0  3
```

Esto es solo un ejemplo de un tipo de partícula, se podría elegir estrellas, rombos hasta figuras extrañas con el fin que se lo desee.

3.4.2. Archivo de Paredes

Para poder definir las paredes, se puede diferenciar dos tipos de paredes, aquellas en donde se van a instanciar partículas y aquellas que no, ya sea inicialmente o al reinsertarse una vez que hayan salido si se elijen condiciones periódicas de contorno.

Este tipo de paredes comenzarían con una I al principio para que se pueda poblar a la partícula en ese área, esto se determinó de esta manera para poder, si es que se necesitara, tener la posibilidad de plasmar diagramas complejos.

Luego, el formato para definir cada pared es:

```
esInicial  radioPared cantidadDePuntos x1 y1 ... xn yn
```

Por ejemplo si se quisiera plasmar el sistema inicial que utilizaremos en la simulaciones se deberían plantear 5 paredes para poder generar un recinto de 12 m x 12 m con una apertura en la pared inferior de 1 m, como todas las partículas van a comenzar en esta área, se las inicio a todas con I y radio 0.05.

```
5
I   0.05   2   0   0   5   0
I   0.05   2   7   0   12  0
I   0.05   2   12  0   12  12
I   0.05   2   12  12  0   12
I   0.05   2   0   12  0   0
```

3.4.3. Archivo de Targets

Cada partícula tiene objetivos a los cuales tiene que llegar en la simulación, todas tienen los mismos, como se mencionó anteriormente.

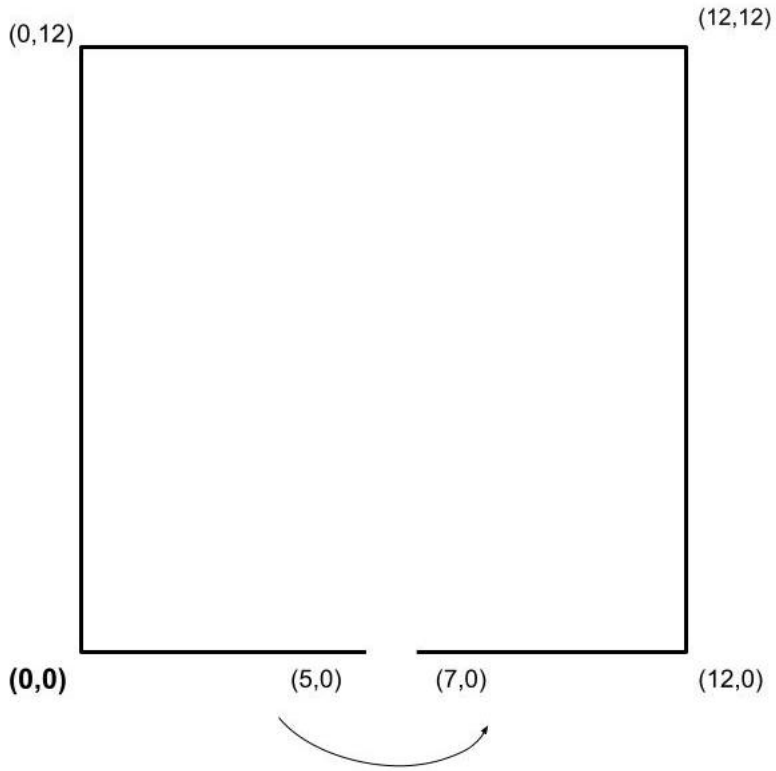


Figura 13: Orden de puntos para paredes iniciales

El archivo consta de varias líneas, en primer lugar se tiene que comenzar con cuantos targets se va a tener. Luego, por cada uno, se necesitará una nueva línea.

Si el target es el último, se tiene que poner una 'F' delante. Se tienen 4 valores con los x1 y1 x2 y2 que son los dos puntos que unen al segmento del objetivo. Vale aclarar que si se pone mas de un target 'F' o targets intermedios luego de uno final, ninguna partícula llegara a ese target ya que luego de llegar al primero se marcará como completado.

Ejemplo: En este ejemplo se muestra que las partículas tienen 2 objetivos, como se plasmo en las simulaciones, uno ubicado en el segmento construido por los puntos (5.7,0) y (6.3,0) y luego el objetivo final ubicado en (3.5, -0.5) a (8.5, -0.5)

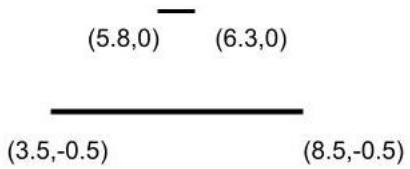


Figura 14: Orden de puntos para targets iniciales

```
2
5.7 0    6.3 0
F    3.5 −0.5     8.5 −0.5
```

3.4.4. Archivo de Constantes

El archivo de constantes enumera una lista (clave valor), la separación tiene que ser hecha con un tabulador, si no se agrega un valor, se toma el valor defecto expresado mas adelante.

Los valores que toma son:

- M_{Max}

La masa máxima de cada partícula, valor predeterminado = 99.5 kg

- M_{Min}

La masa minima de cada partícula, valor predeterminado = 59.5 kg

- totalTime

Tiempo total de la simulacion, valor predeterminado = 1000 seg

- d_{T}

Intervalo de tiempo de integración, valor predeterminado = 0.00025 seg

- printTime

Intervalo de tiempo de impresión, valor predeterminado = 0.05 seg

- continuous

Define si las partículas se tienen condiciones periódicas de contorno, es un valor booleano que por defecto es Verdadero

- R_{Max}

Radio máximo de cada partícula, valor predeterminado = 0.165 m

- M_{Min}

Radio mínimo de cada partícula,valor predeterminado = 0.12 m

- k_{t}

Constante del resorte en la dirección tangencial, valor predeterminado = $0{,}088 \times 10^6 N/m$

- K_{n}

Constante del resorte en dirección normal, valor predeterminado = $2{,}2 \times 10^6 N/m$

- grav

Valor de la gravedad, valor predeterminado = 10 m/s

- β

coeficiente de amortiguamiento, valor predeterminado = $4{,}5\sqrt{SD} = 4{,}5\sqrt{15}$

- S_{D}

Fuerza direccional, valor predeterminado = 15 Nm

- η

Ruido del sistema ,valor predeterminado = 0.75

- μ

Coeficiente de fricción microscópica, valor predeterminado = 0.5

3.5. Ejecución del Algoritmo

Para poder manejar la simulación, se parametriza un Δt y un tiempo máximo, que genera un ritmo a la simulación.

Teniendo en cuenta los conceptos explicados anteriormente, el algoritmo planteado fija una iteración por cada paso temporal, donde se calcula las fuerzas que cada elemento particular tiene con su entorno, a través de las fórmulas del social force model, y luego actualizar sus posiciones dado las fuerzas que cada una de ellas siente y su nuevo lugar de destino si esto fuese necesario. Cada vez que alguna partícula logra evacuar el recinto, la misma es reinsertada nuevamente al sistema para mantener el balance de la cantidad de entidades participando.

```
while (time < TOTAL_TIME) {
    guardarParticulasAnteriores();
    calcularFuerzas();
    for (particle : particles) {
        if (! particle.isWall) {
            particle.actualizarPosiciones();
            particle.actualizarObjetivos();
        }
    time += DELTA_TIME;
    }
}
```

3.5.1. Ejecución de la Simulación

La simulación consta de un ciclo while que corre desde 0 hasta el tiempo total señalado en las constantes.

En cada momento se imprime el sistema en un archivo si el tiempo coincide con la impresión. Luego, como las partículas pueden atascarse se agrego una condición de que si hace más de $t_{\max}$ segundos no salió ninguna partícula, entonces se termina la simulación.

El próximo paso será calcular las fuerzas que cada partícula siente, con los métodos explicados anteriormente, para que no haya errores, se guardan las partículas anteriores en un arreglo ya que cada vez que se calcula una nueva fuerza, se reemplaza su valor anterior. Se decidió crear un arreglo separado para que no se estén calculando fuerzas de partículas con valores incorrectos.

Una vez que se obtienen las fuerzas, entonces se actualizan las posiciones, velocidades y orientaciones, nuevas y anteriores utilizando el método de verlet. El método de verlet no incluye orientaciones, pero fue esencial ya que si no fuese por esto, las partículas no rotarían en base a lo que experimentan.

Al saber cual es la nueva posición, se puede saber si realmente la partícula llegó al target o no, es así como se actualizan los targets, se actualiza el tiempo y se hace todo el proceso de vuelta.

Si se llegó al final de la simulación, se imprime el caudal para poder evaluar, aunque también están los archivos que darán más información.

```
while (time < TOTAL_TIME) {
    if ( hay_que_imprimir) {
        printSystem();
    }
    if (total_time - last_particle_exit > 20) {//Si pasaron mas de 20 segs desde que salio una particula
        end_simulation();
    }
    savePreviousParticles();
```

```
    calcularFuerzas();
    for (particle: particles) {
        if (! particle.isWall) {
            particle.updatePosition();
            particle.updateTargets();
        }
    time += DELTA_TIME;
    }
}
print_caudal();
```

3.6. Archivos de Visualización

Para poder medir resultados de la ejecución del algoritmo, en cada cierto tiempo $T_{\text{impresión}}$ se plasman resultados parciales en 2 archivos. Uno que muestra las posiciones de todo el sistema en un determinado tiempo y otro que, marca el tiempo exacto de salida de cada partícula.

Cada uno tiene un objetivo por el cual se realiza, el primero ayuda para poder dibujar el sistema y poder tener una clara representación visual de que paso cada cierto tiempo, no tendría sentido imprimir en cada dt ya que se cargaría mucho el sistema, para ello se eligió un $T_{\text{impresión}}$ que hace que el archivo pueda ser visualizado correctamente. Se utilizó el programa ovito para visualizar las diferentes simulaciones, ya que tiene la opción de generar una secuencia dadas visualizaciones de momentos específicos. Se imprimió el formato que este recibe, que es un formato .xyz. El documento tiene que estar diagramado de la siguiente manera: En la primera línea la cantidad de puntos a enumerar, en la segunda el tiempo de ese instante y luego una línea por cada punto. Se decidió, para que sea claro, imprimir las paredes en color rojo, los objetivos en amarillo y luego las partículas en blanco.

El segundo archivo ayuda para poder sacar conclusiones, cuál es el efecto de los parámetros en el caudal, graficar partículas vs. tiempo de salida y poder comparar con otros resultados. Para esto, cada vez que una partícula egresa de su último objetivo, se incrementa el contador de cuántas partículas egresaron y se computa el tiempo y el valor de este contador.

4. Simulaciones

Luego de todo lo explicado, se decidió evaluar los 4 parámetros más importantes para poder entender el sistema, y ver como realmente los parámetros afectan al sistema. Para encontrar puntos de quiebre y patrones.

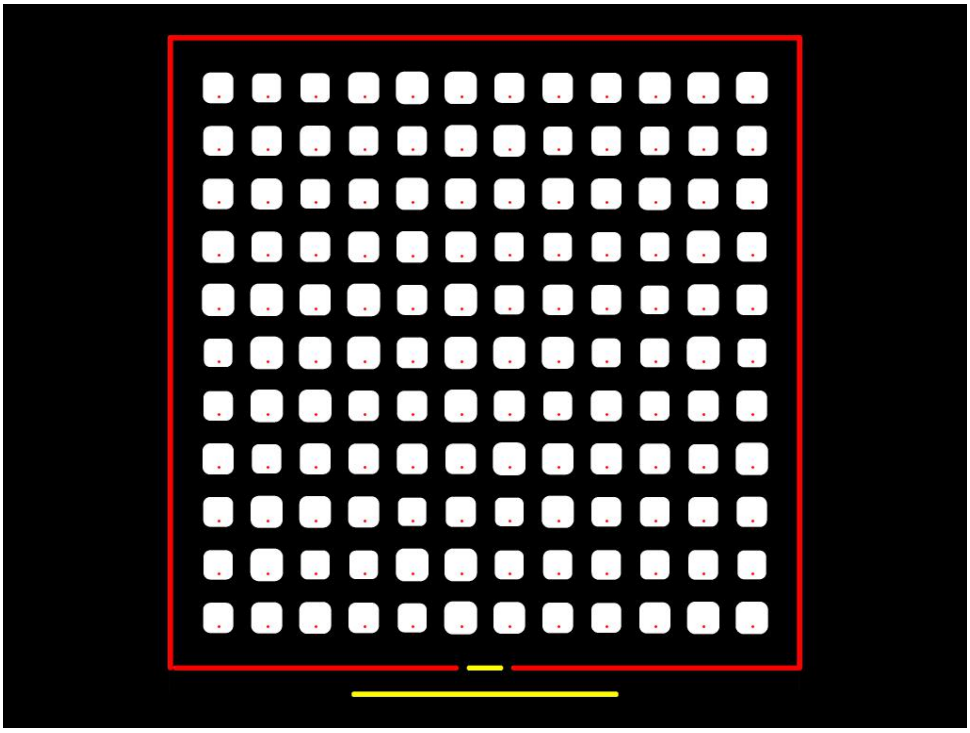


Figura 15: Estado inicial al comienzo de la simulación. Se puede visualizar los miembros a evacuar como los cuadrados blancos, la orientación de los mismos determinada por el punto rojo. Las paredes se muestran como lineas rojas mientras que los distintos objetivos que deben ser alcanzados para evacuar se muestran en color amarillo

Se eligió un sistema homogéneo compuesto por 132 partículas en forma de cuadrado de lado 0,3 m comprendidas simétricamente en un recinto cuadrado de 12 m de lado (ver figura 15). Cada partícula cuenta con una masa comprendida entre $[59,5kg - 99,5kg]$ y un radio asociado a los esferocilíndros que lo componen $[0,12m - 0,165m]$ con una orientación inicial hacia la pared donde se encuentra la salida.

Con una apertura de 1 metro en el centro de la pared inferior y 2 objetivos, uno directamente en la apertura y otro 0.5 metros más abajo. Esto se implementó para poder evitar que algunas partículas intenten atravesar las paredes cuando esto es imposible.

Vale aclarar que el sistema tenia condiciones periódicas de contorno, es decir que al salir una partícula, se volvía a reinsertar en el sistema para poder ver como el sistema se comporta pasando un tiempo grande.

El tiempo total de simulación fue de 1000 segundos con un $\Delta t = 2{,}368 \times 10^{-5}$ s y un $t_{\text{impresión}} = 0{,}05s$

El valor de $k_{\text{t}} = 2{,}2 \times 10^6 N/m$ y $k_{\text{n}} = \dfrac{k_{\text{t}}}{25} N/m$

Por defecto para todas las simulaciones se mantuvo constante en los casos donde no se estaba estudiando el parámetro: ancho de la apertura (W) 1m, $S_{\text{D}} = 25Nm$, $\eta = 0{,}75$, $v_{\text{d}} = 1{,}5m/s$

Se eligió estudiar 4 parámetros

S_{D} - la intensidad con la que una partícula deseaba alinearse con su objetivo.

η - la intensidad del ruido generado por las partículas al moverse

v_{d} - velocidad de autopropulsión de las partículas

W o apertura - apertura de la salida inferior

Tomando 5 valores diferentes, 2 por arriba de la constante y 2 por debajo, corriendo cada una de las configuraciones 5 veces y tomando un promedio de ellas.

Como parámetros a evaluar se estudió el caudal móvil, que tomando un N fijo marca una "ventana"de la cantidad N partículas que salieron sobre el tiempo que transcurrió tal como se muestra en la fórmula (27). Se denomina a esta ventana móvil como $\mathrm{V_N}$, donde el valor que N tomó se varió hasta definir el valor adecuado.

Se analizaron los gráficos para $\mathrm{V_{N=50}}$, $\mathrm{V_{N=100}}$, $\mathrm{V_{N=150}}$, $\mathrm{V_{N=200}}$ y $\mathrm{V_{N=250}}$, eligiéndose $\mathrm{V_{N=150}}$m, ya que ventanas más chicas generaban curvas muy sensible a cambios y ventanas grandes mostraban el efecto contrario, curvas que se mantenían estables pese a cambios que ocurrían. Analizando los gráficos se decidió elegir un $\mathrm{V_{N=150}}$, ya que se visualizó un balance entre las diferentes curvas.

$$Q_{\mathrm{x}} = \frac{\Delta N_{\mathrm{fijo}}}{T(X + \Delta N) - T(X)} \tag{27}$$

Adicionalmente se estudia la función de supervivencia de los tiempos de salida entre partículas. Se busca analizar si los mismos respetan el comportamiento y distribución de una ley de poder y si es posible ajustar los tiempos entre salidas con el método presentado por [14].

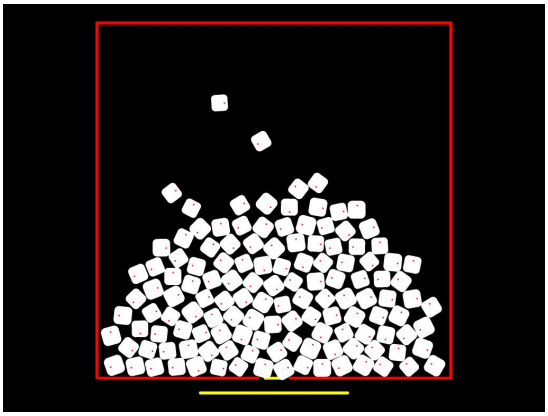
5. Resultados

5.1. Variación del S_{D}

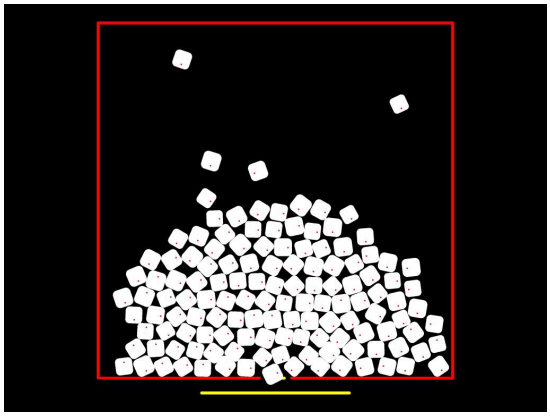
Se varió S_{D}, o la intensidad con la que una partícula deseaba alinearse con su objetivo, entre los valores mostrados en la tabla a continuación (Tabla 1)

S_{D}	η	v_{d}	W
0 Nm	0.75	1.5 m/s	1 m
12.5 Nm	0.75	1.5 m/s	1 m
25 Nm	0.75	1.5 m/s	1 m
37.5 Nm	0.75	1.5 m/s	1 m
50 Nm	0.75	1.5 m/s	1 m

Tabla 1: Valores estudiados en cada una de las simulaciones



(a) Sistema en equilibrio, $S_{\mathrm{D}} = 0$ Nm visualizada en https://youtu.be/Isdd4Ff_Dds



(b) Sistema en equilibrio, $S_{\mathrm{D}} = 50$ Nm visualizada en https://youtu.be/LTl9OOIA1WI

Figura 16: Comparación del sistema en equilibrio para variaciones de S_{D}

Analizando los casos extremos de los valores estudiados y viendo la disposición de las partículas una vez alcanzado el equilibrio (Ver figura 16) se puede determinar parte del impacto que la intensidad de S_{D} tiene sobre el modelo. En primer lugar, a partir de las imágenes y animaciones, la falta de preferencia en la orientación genera un desorden en las corridas que no permiten llegar a una estructura en donde las partículas estén bien pegadas las unas a las otras. A diferencia de esto, los valores más elevados consiguen un poco más de cercanía entre las distintas partículas próximas a la salida.

Esta des-estructuración de las partículas es algo también visto al elevar el valor de η y dado que S_{D} se opone al ruido o movimiento aleatorio generado por η, los efectos de la variación de una también impacta su contra parte.

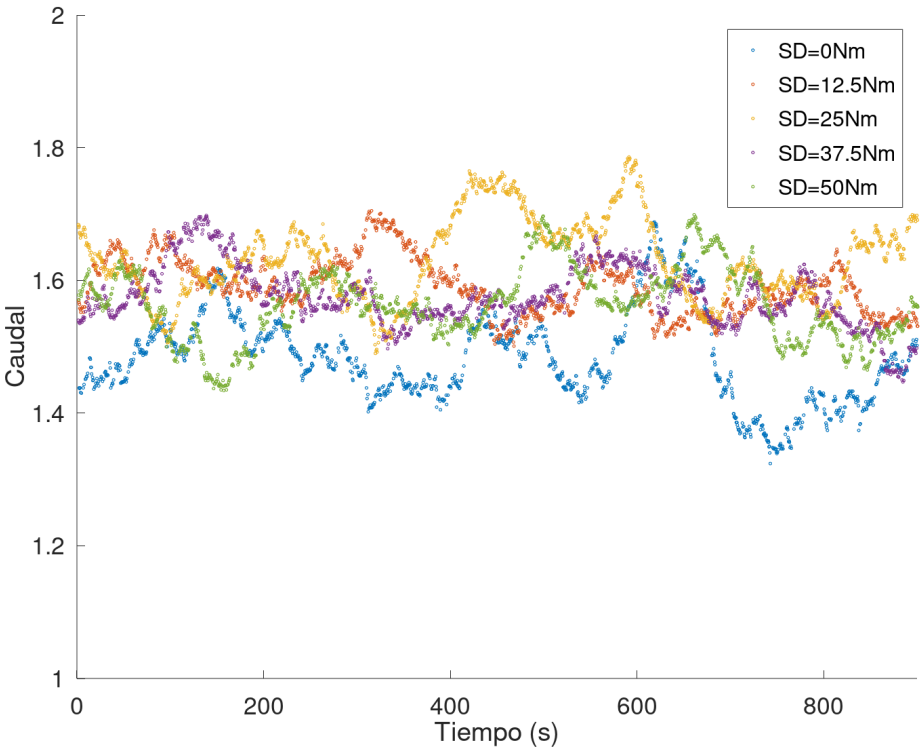


Figura 17: Caudal calculado por medio de una $V_{N=150}$ para los distintos valores de S_D

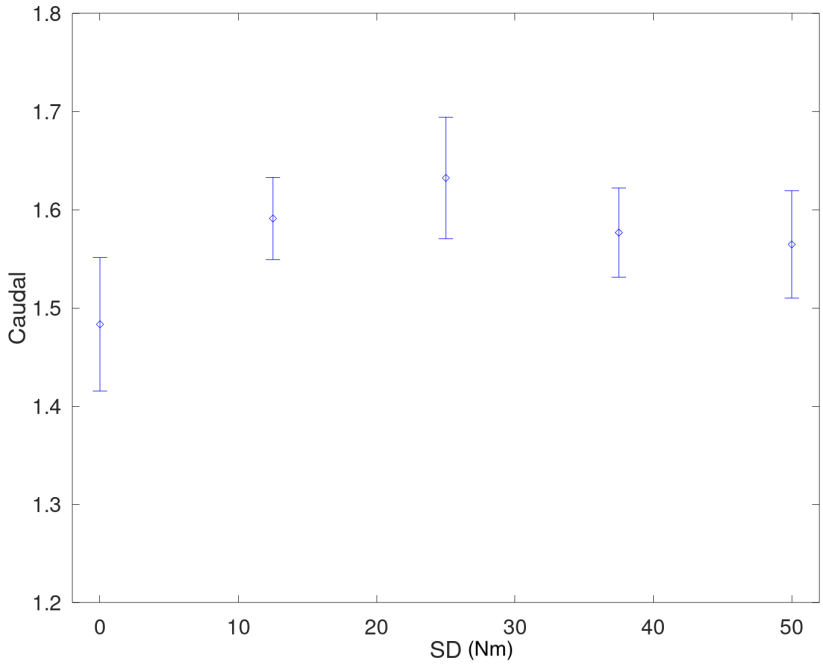


Figura 18: Caudal promedio para cada valor de S_D donde la barra de error representa un desvió estándar.

Incluso con la diferencia que se visualiza en los fotogramas de la simulación, a partir de la figura 18 se puede ver que el impacto final que se tiene en la simulación es muy bajo, aumentando únicamente en 10 % el porcentaje de caudal entre el mejor y peor caso para las simulaciones.

Como dato adicional el desvío estándar representado como las barras de error en la (figura 18) se mantienen constantes para los diferentes valores de S_{D}.

Pero se puede ver una sutil diferencia observando el caudal móvil en la (figura 17) donde hay un claro pico cuando $S_{\mathrm{D}} = 25$ Nm y un mínimo cuando $S_{\mathrm{D}} = 0$ Nm. Se observa que aumentar $\mathrm{S_D}$ a valores mayores que 25 Nm se entorpece la salida de las partículas, generando así un mayor grado de roce y fuerzas entre las mismas.

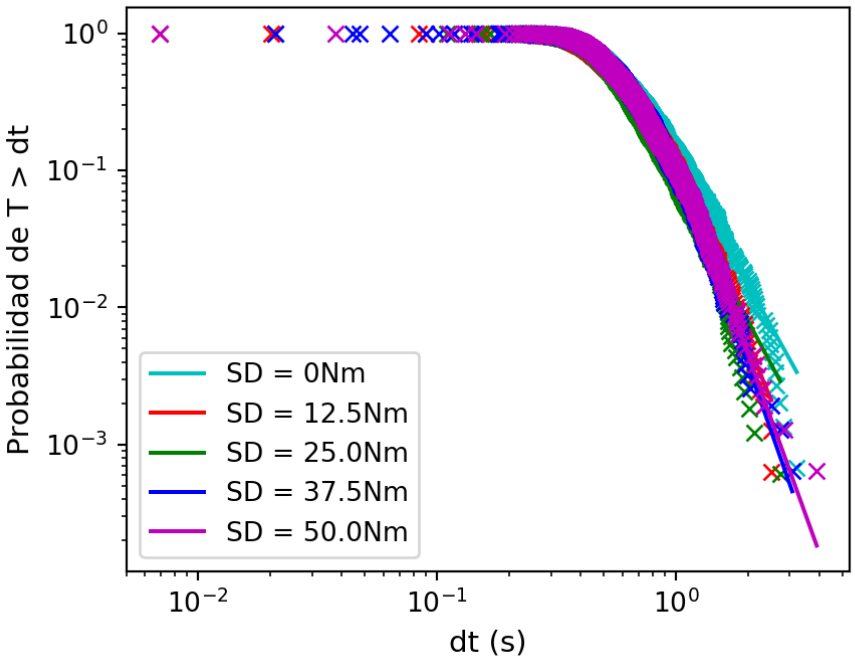


Figura 19: Función de supervivencia del tiempo de salida entre dos partículas para los valores de S_{D}. Se visualiza la función de supervivencia marcada con cruces y el ajuste de la ley de potencia obtenido mediante el método de Clauset-Shalizi-Newman [14] como las respectivas lineas rectas continuas. Los exponentes de los ajustes son $\alpha_1 = 4{,}13 \pm 0{,}18$, $\alpha_2 = 5{,}27 \pm 0{,}33$, $\alpha_3 = 4{,}26 \pm 0{,}13$, $\alpha_4 = 5{,}98 \pm 0{,}49$ y $\alpha_5 = 5{,}83 \pm 0{,}37$

Visualizando las curvas de supervivencia de la (figura 19) se ve como los tiempos de salida entre partículas respeta una distribución de cola pesada la cual fue ajustada como se describe en la imagen. Se destaca que únicamente el último tramo de la función se puede ajustar al respetar el modelo que una ley de poder presentaría. Todos estos ajustes se obtuvieron con un valor p >0.1, asegurando que la aproximación de los datos al modelo no es invalida. Se destaca una mayor probabilidad de encontrar tiempos de pasaje mayores al trabajar sin $\mathrm{S_D}$, alineándose con lo que se espera del menor caudal final. Para una mejor visualización de los parámetros del ajuste obtenido se ver la (figura 20).

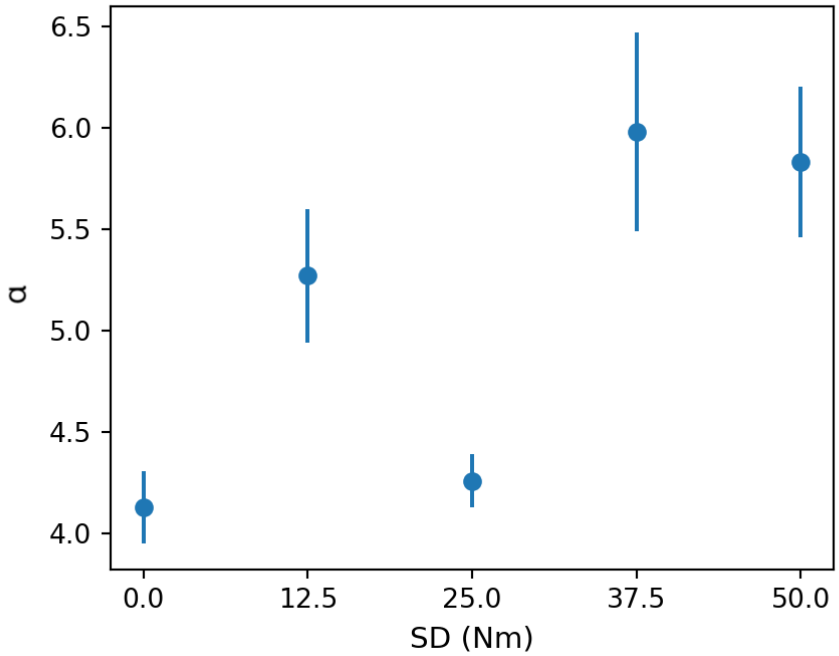


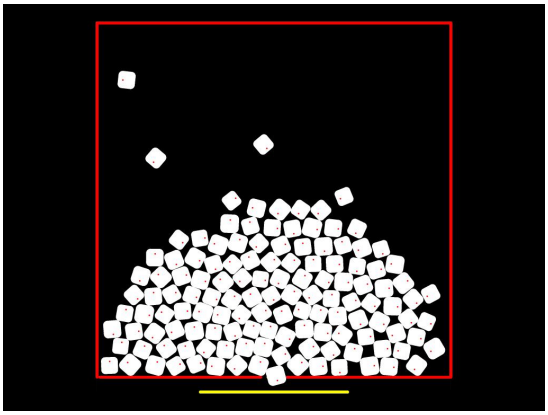
Figura 20: α resultantes de los ajustes a las leyes de potencia para cada uno de los valores de S_D. Las barras de error representan el desvío estandar.

5.2. Variación de la apertura de la Salida

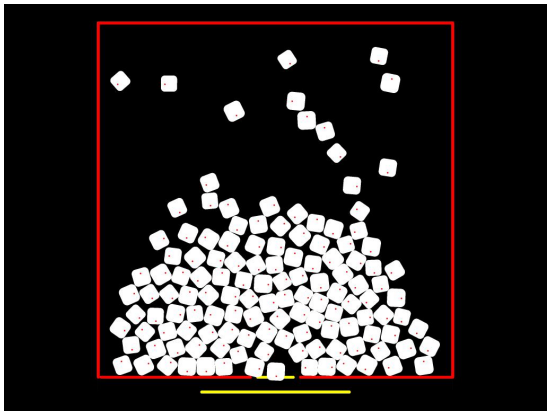
Se varió la apertura de la salida (W) con los valores mostrados en la tabla a continuación (Tabla 2)

S_D	η	v_d	W
25 Nm	0.75	1.5 m/s	0.8 m
25 Nm	0.75	1.5 m/s	1 m
25 Nm	0.75	1.5 m/s	1.2 m
25 Nm	0.75	1.5 m/s	1.4 m
25 Nm	0.75	1.5 m/s	1.6 m
25 Nm	0.75	1.5 m/s	1.8 m

Tabla 2: Valores estudiados en cada una de las simulaciones



(a) Sistema en equilibrio, apertura de 0.8 m visualizada en https://youtu.be/hhRgA5QuVto



(b) Sistema en equilibrio, apertura de 1.6 m visualizada en https://youtu.be/Z_8yBeMz1ZI

Figura 21: Comparación del sistema en equilibrio para variaciones de W

Visualizando fotogramas del sistema una vez alcanzado el equilibrio (figura 21) y visualizando las animaciones indicadas, se puede notar cómo la diferencia principal en el sistema se encuentra en la inmediatez a la salida. Alrededor de la misma se forman bloqueos en forma de pequeños arcos que proceden a derrumbarse y liberar a las partículas mas próximas a salir.

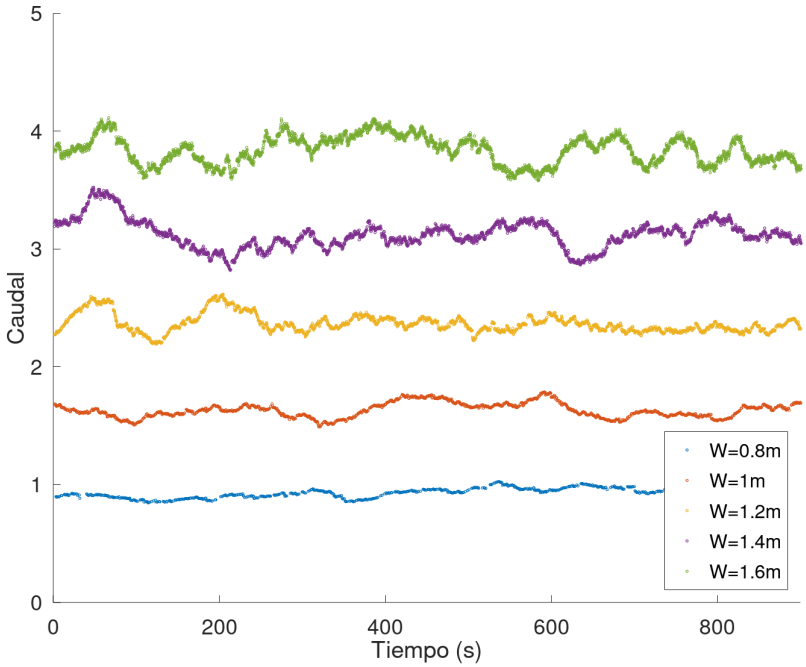


Figura 22: Caudal móvil calculado con una $V_{N=150}$ para los distintos valores de W.

Se puede ver en la (figura 22) cómo la variación en la apertura afecta el caudal móvil a lo largo de la simulación. El mismo crece gradualmente a medida que se abre la apertura. Pero un factor que resulta interesante es cómo a medida que crece el caudal, también el mismo fluctúa en el tiempo más abruptamente (esto se puede evidenciar en la misma figura visualizando el caudal móvil correspondiente a las aperturas más grandes). Este fenómeno se da debido a restricciones propias del modelo de fuerzas, donde las mismas partículas generan bloqueos que hasta no ser liberados, bloquean momentá-

neamente el egreso del recinto generando las bajas en el caudal.

Para mayor claridad sobre el aumento en la variabilidad del caudal a lo largo de las simulaciones es mejor observar la (figura 23), donde el desvío estándar representado como las barras de error crece consistentemente al aumentar la apertura.

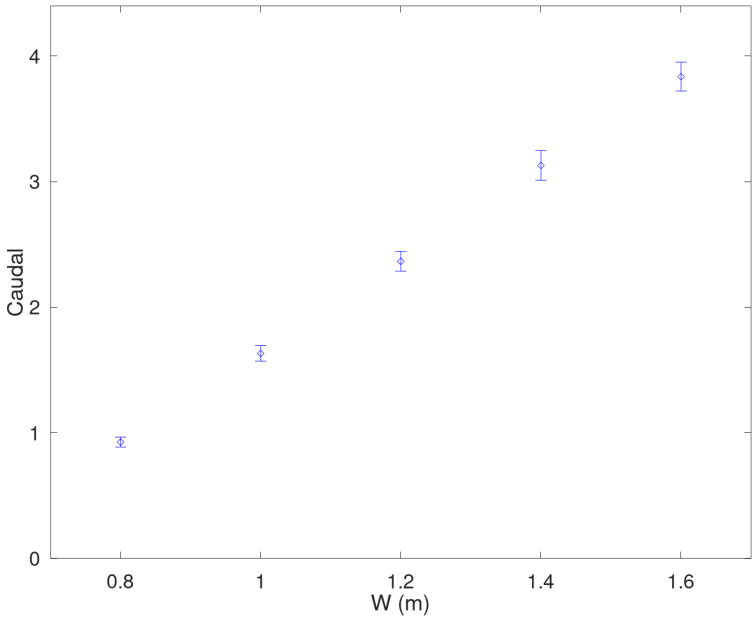


Figura 23: Caudal de la simulación vs la variación de la apertura (W).

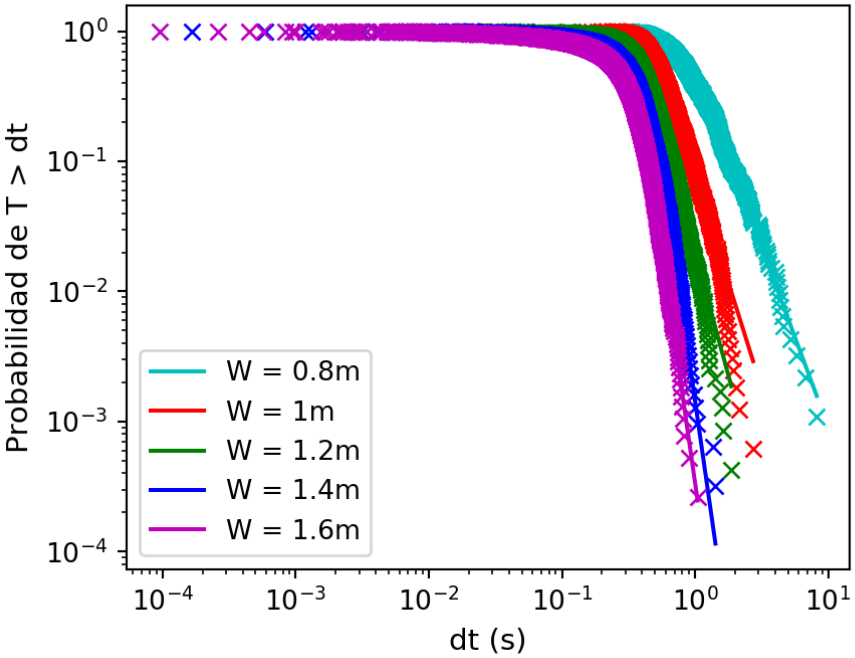


Figura 24: Función de supervivencia del tiempo de salida entre dos partículas para los valores de W. Se visualiza la función de supervivencia marcada con cruces y el ajuste de la ley de potencia obtenido mediante el método de Clauset-Shalizi-Newman [14] como las respectivas lineas rectas continuas. Los exponentes de los ajustes son $\alpha_1 = 3{,}75 \pm 0{,}19$, $\alpha_2 = 4{,}26 \pm 0{,}13$, $\alpha_3 = 4{,}74 \pm 0{,}13$, $\alpha_4 = 8{,}16 \pm 0{,}64$ y $\alpha_5 = 8{,}12 \pm 0{,}44$

Alineándose con lo visto en los resultados para las pruebas de S_D, se puede realizar el ajuste de los datos a una ley de potencia para la caída final visualizada en la función de supervivencia de la (figura 24). Para todos los valores de W se pudo realizar el ajuste con valor p superior a 0.1, nuevamente mostrándolo como un modelo adecuado. Adicionalmente se denota la marcada caída en los tiempos de pasaje al incrementar el ancho de la salida, algo que se evidencia en especial con la curva de la apertura más pequeña y analizando la (figura 25). En la misma es muchísimo más probable encontrarse con tiempos de pasaje o atascos más elevados, debido al poco espacio y alta competitividad que hay por lograr conseguir alinearse con el mismo con aperturas pequeñas. Se puede notar una clara tendencia a la suba de los valores de α y por lo tanto una baja en la probabilidad de encontrar estancamientos prolongados con aperturas mayores.

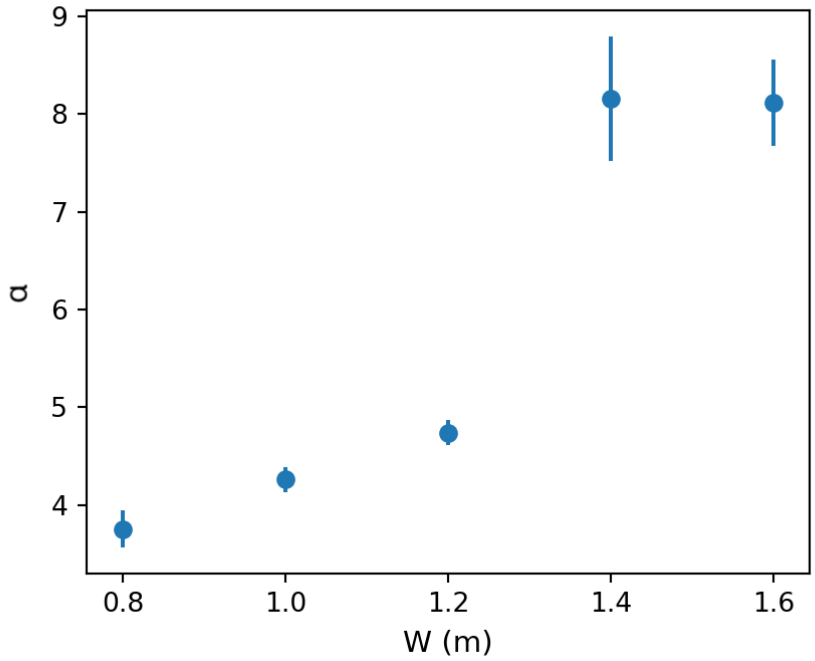


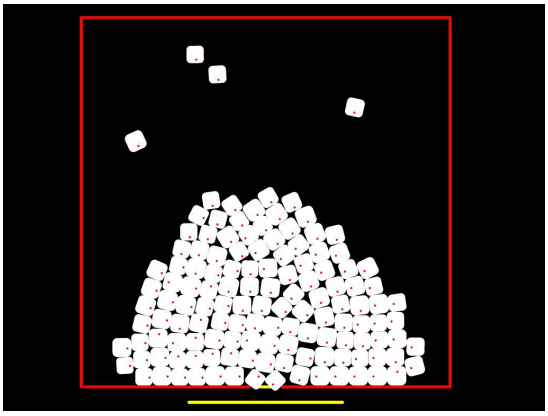
Figura 25: α resultantes de los ajustes a las leyes de potencia para cada uno de los valores de W. Las barras de error representan el desvío estándar.

5.3. Variación de η

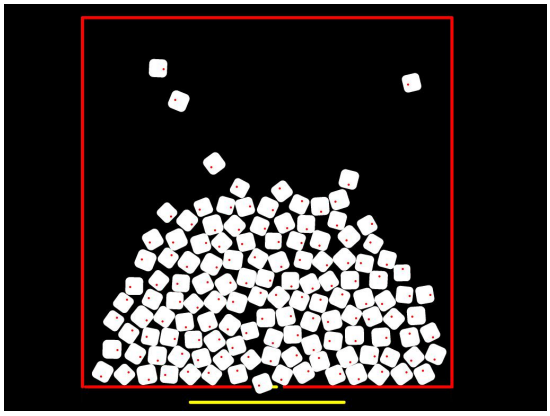
Se varió el η, el ruido del sistema, con los valores que se muestran en la tabla a continuación (Tabla 3)

S_D	η	v_d	W
25 Nm	0	1.5 m/s	1 m
25 Nm	0.375	1.5 m/s	1 m
25 Nm	0.75	1.5 m/s	1 m
25 Nm	1.125	1.5 m/s	1 m
25 Nm	1.5	1.5 m/s	1 m

Tabla 3: Valores estudiados en cada una de las simulaciones



(a) Sistema en equilibrio, $\eta = 0$ visualizada en https://youtu.be/hhRgA5QuVto



(b) Sistema en equilibrio, $\eta = 1.5$ visualizada en https://youtu.be/Sthfx-z301g

Figura 26: Comparación del sistema en equilibrio para variaciones de η

A partir de las imágenes de la simulación (figura 26) y las animaciones indicadas resulta evidente el efecto disruptivo que el ruido aporta al sistema. La ausencia del mismo genera una estructura en la que todas las partículas se encuentran chocando contra varias a la vez, sintiendo una suma de fuerzas que las tiende a trabar.

De manera opuesta, al aumentar la intensidad del ruido se genera el efecto adverso, donde las partículas giran violentamente y no permiten la llegada a una estructura con un orden. En este caso se ve como continuar aumentando η aporta bajas al caudal final, puesto que los choques entre las partículas rotando tan fuertemente no permiten el flujo del sistema.

Observando la relación que hay entre el caudal de las diferentes variantes del η se puede ver que con $\eta = 0$, es decir que no hay ruido, el sistema se estanca y el flujo de partículas que sale es mínimo tal como se muestra en la (figura 27), se puede entender que es algo necesario a tener presente durante la simulación.

Con un η bajo se puede ver que ayuda a des-trabajar el sistema y sigue creyendo como se muestra en el caudal promedio en la (figura 28) hasta un pico en $0,75$ pero al continuar elevando los valores se genera una reducción en el flujo ya que las fuerzas entre partículas terminan siendo muy fuertes y entorpeciendo el flujo.

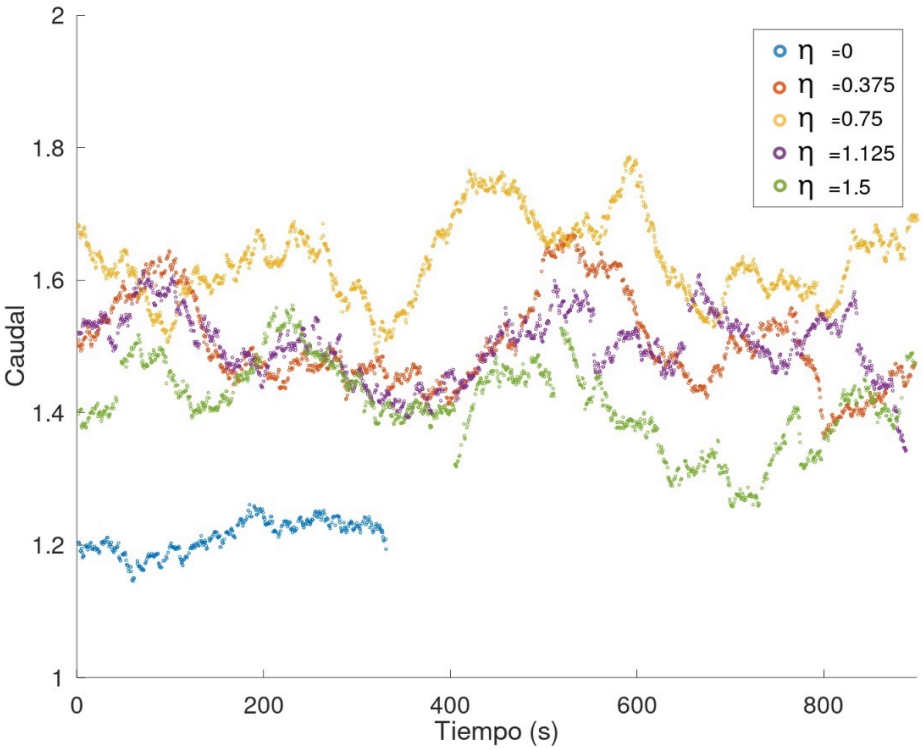


Figura 27: Caudal móvil tomado a partir de una $V_{N=150}$ para los distintos valores de η.

Analizando el gráfico del caudal móvil en la (figura 27) se nota como las simulaciones corriendo sin ruido no logran finalizar. Puesto que pasa un 20 % del tiempo de las mismas sin que alguna partícula evacue. Se nota la necesidad del sistema por contener un componente no nulo de ruido para ayudarlo a salir de situaciones de estancamiento como las vistas en la)figura 26).

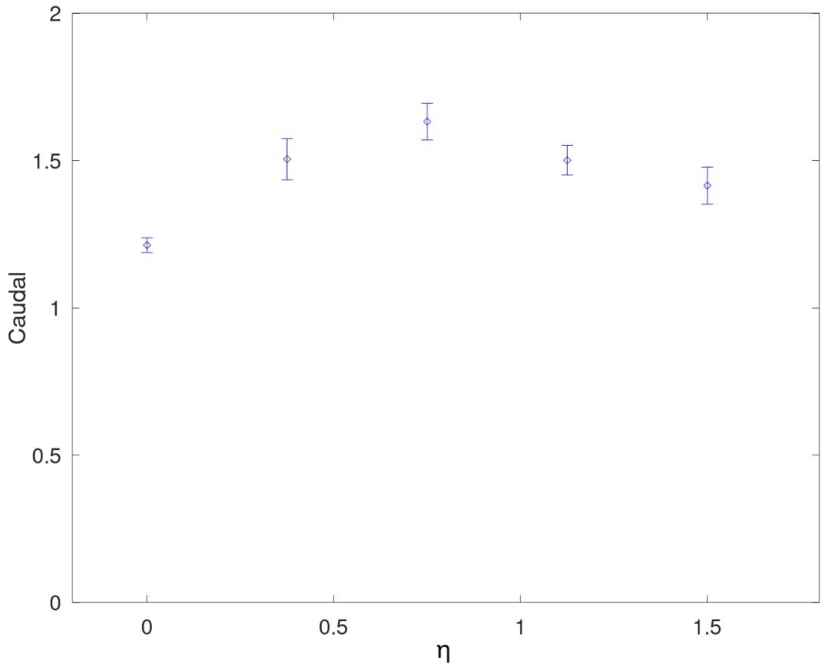


Figura 28: Caudal de la simulación vs la variación de η donde las barras de error representan una desviación estándar.

A partir de la (figura 27) queda claro el impacto que el ruido tiene y aunque establecido que el mismo resulta ser fundamental para poder llevar adelante el modelo, excederse con la intensidad del mismo trae reducciones al flujo del sistema por aportar tanto desorden

Se puede observar a su vez, que el desvío estándar representado como las barras de error en la (figura 28) muestra que no hay mucha variación a medida que se varía el valor de η en comparación con otros parámetros.

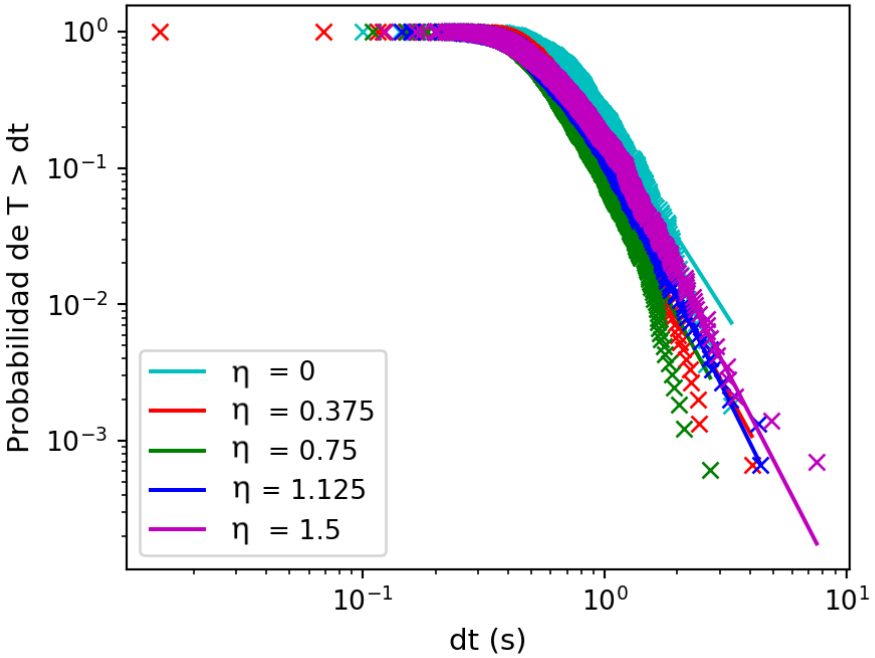


Figura 29: Función de supervivencia del tiempo de salida entre dos partículas para los valores de η. Se visualiza la función de supervivencia marcada con cruces y el ajuste de la ley de potencia obtenido mediante el método de Clauset-Shalizi-Newman [14] como las respectivas lineas rectas continuas. Los exponentes de los ajustes son $\alpha_1 = 3{,}77 \pm 0{,}16$, $\alpha_2 = 4{,}22 \pm 0{,}14$, $\alpha_3 = 4{,}26 \pm 0{,}13$, $\alpha_4 = 4{,}58 \pm 0{,}26$ y $\alpha_5 = 4{,}42 \pm 0{,}32$

Visualizando las curvas de supervivencia de la (figura 29) nuevamente se pueden obtener ajustes a una distribución de una ley de potencia para los datos con un nivel de significancia alto. Resulta interesante ver que para el los valores extremos de η la curva muestra una mayor probabilidad para estancamientos más largos. Para el caso más bajo esto se vuelve a explicar como la incapacidad del sistema para destrabarse, mientras que para $\eta = 1{,}5$ se explica como un exceso de ruido que no permite orden y alineación con la salida. Observando aparte la (figura 30) se puede ver más clara la tendencia de aumento que hay en α en los ajustes. Se puede observar como a mayores valores de η se ve una tendencia a menores estancamientos prolongados, sugiriendo que a pesar de haber un mayor nivel de desorden, los estancamientos resultan inferiores.

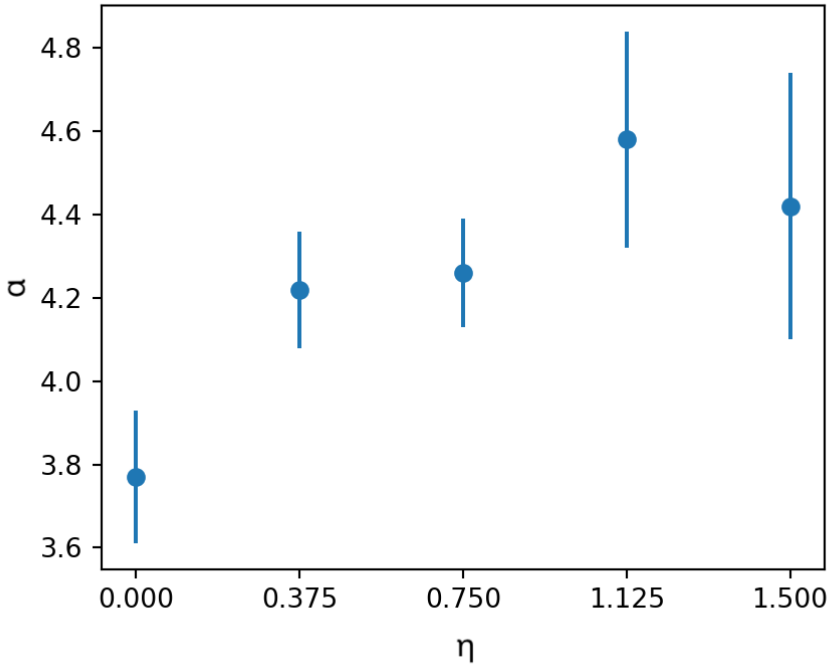


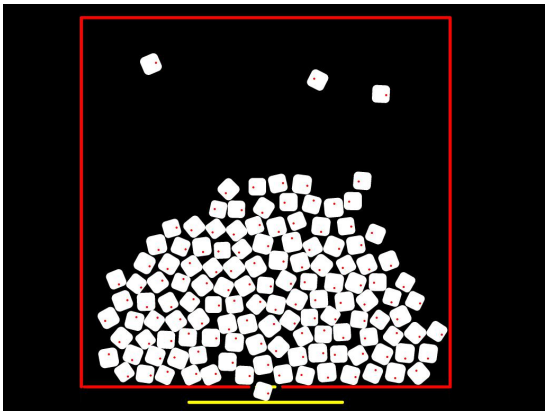
Figura 30: α resultantes de los ajustes a las leyes de potencia para cada uno de los valores de η. Las barras de error representan el desvío estándar.

5.4. Variación de la velocidad

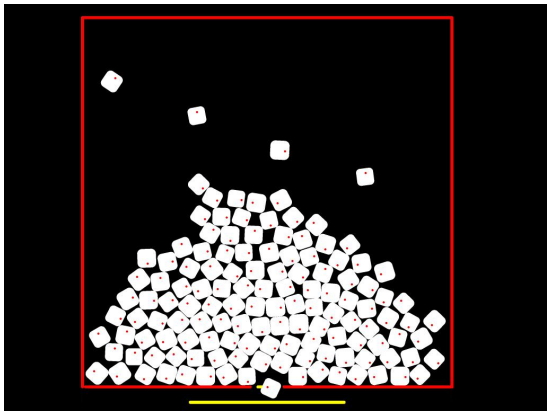
Se vario la velocidad de autopropulsión de las partículas V_{d} con los valores que se muestran en la tabla a continuación (Tabla 4).

S_{D}	η	v_{d}	W
25 Nm	0.75	0.5 m/s	1 m
25 Nm	0.75	1 m/s	1 m
25 Nm	0.75	1.5 m/s	1 m
25 Nm	0.75	2 m/s	1 m
25 Nm	0.75	2.5 m/s	1 m

Tabla 4: Valores estudiados en cada una de las simulaciones



(a) Sistema en equilibrio, $V_d = 0.5$ m/s visualizado en https://youtu.be/SsrCH18wgjc



(b) Sistema en equilibrio, $V_d = 2.5$ m/s visualizado en https://youtu.be/zd19TYP0aGw

Figura 31: Comparación del sistema en equilibrio para variaciones de V_d

Como se puede esperar de los resultados, al querer moverse mas rápido, el caudal crece. Pero se puede observar que los incrementos de los valores más pequeños generan aumentos mucho mayores, que al aumentar las velocidades el caudal apenas se ve mejorado.

Esto se debe a que al haber una autopropulsión mayor por moverse mas rápido, el sistema deja de fluir tan libremente y se ven las partículas mas apretadas, ya que las fuerzas de contacto no terminan de permitir el libre movimiento. Esto se evidencia en la (figura 31) y en las animaciones indicadas, donde al auto propulsarse con mayor intensidad y moverse más rápido se genera un aglomeramiento mucho más compacto en donde las fuerzas entre partículas terminan deteniendo la libre circulación y controlando la velocidad general del sistema para evacuarse.

El tiempo que se demora en alcanzar el equilibrio se ve creciente a medida que aumenta la velocidad. Quitando del medio el pico inicial que hay debido a las partículas que llegan a egresar sin obstáculos, el tiempo que se demora en alcanzar el mismo asciende consistentemente. Se evidencia también que a medida que el caudal aumenta debido a velocidades más altas, el mismo se vuelve menos consistente, evidenciando fluctuaciones muy fuertes(Ver figura 32).

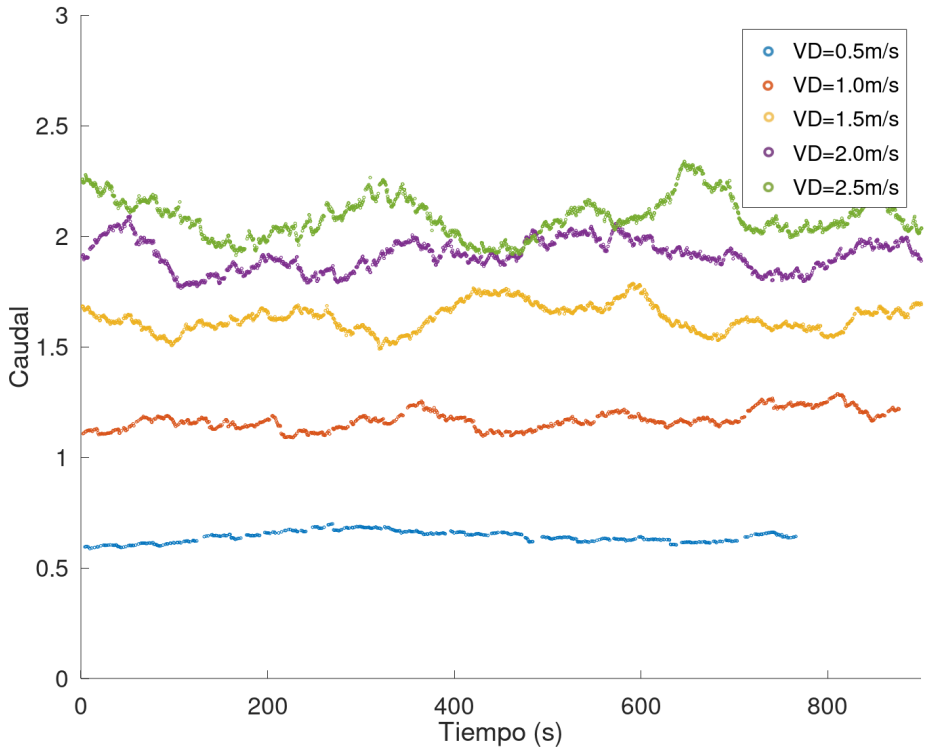


Figura 32: Caudal móvil tomado a partir de una $V_{N=150}$ para los distintos valores de V_d.

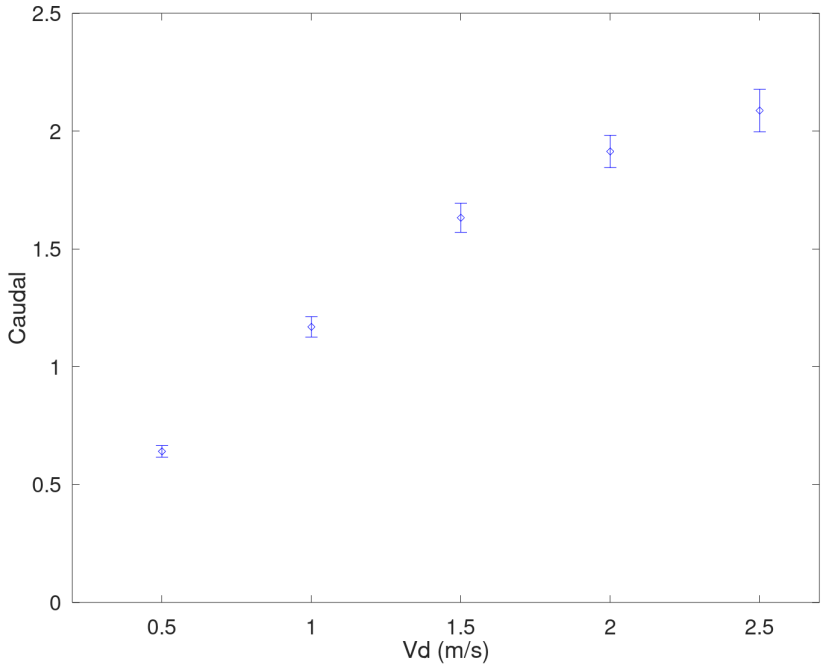


Figura 33: Caudal promedio de la simulación vs. la variación de V_d donde las barras de error representan una desviación estándar.

Adicionalmente, observando la (figura 33) se puede observar como se estanca el crecimiento en el caudal promedio generado por los aumentos en la velocidad deseada de propulsión. Nuevamente se ve reflejado cómo las fuerzas que las partículas se ejercen entre sí terminan generando bloqueos y obstruyendo el libre flujo. Esto es algo que casi no se visualiza en las velocidades más bajas, donde al haber una autopropulsión mucho más leve, los roces y fuerzas ofrecen un menor grado de restricción al movimiento.

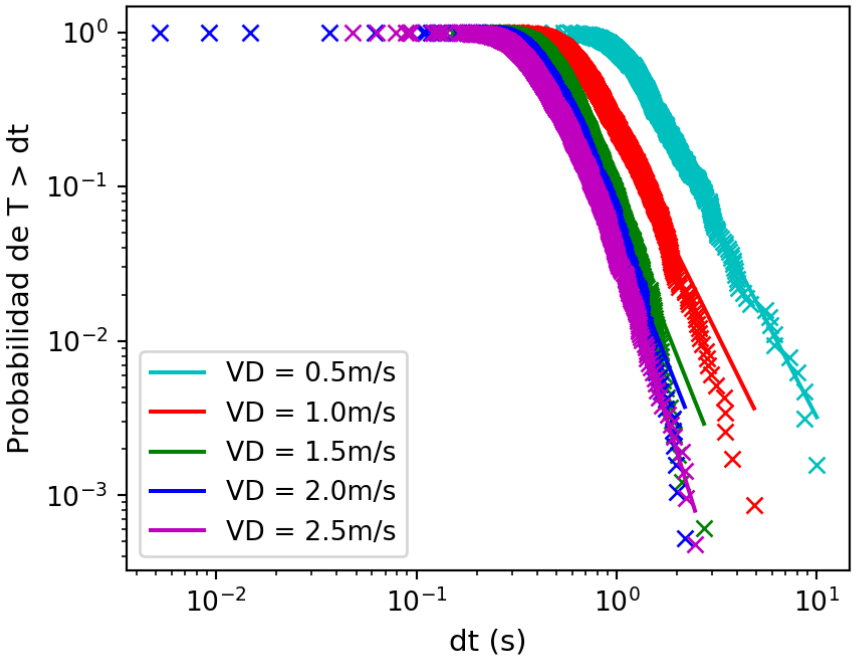


Figura 34: Función de supervivencia del tiempo de salida entre dos partículas para los valores de V_d. Se visualiza la función de supervivencia marcada con cruces y el ajuste de la ley de potencia obtenido mediante el método de Clauset-Shalizi-Newman [14] como las respectivas lineas rectas continuas. Los exponentes de los ajustes son $\alpha_1 = 3{,}48 \pm 0{,}12$, $\alpha_2 = 3{,}61 \pm 0{,}11$, $\alpha_3 = 4{,}26 \pm 0{,}13$, $\alpha_4 = 4{,}29 \pm 0{,}14$ y $\alpha_5 = 5{,}25 \pm 0{,}32$

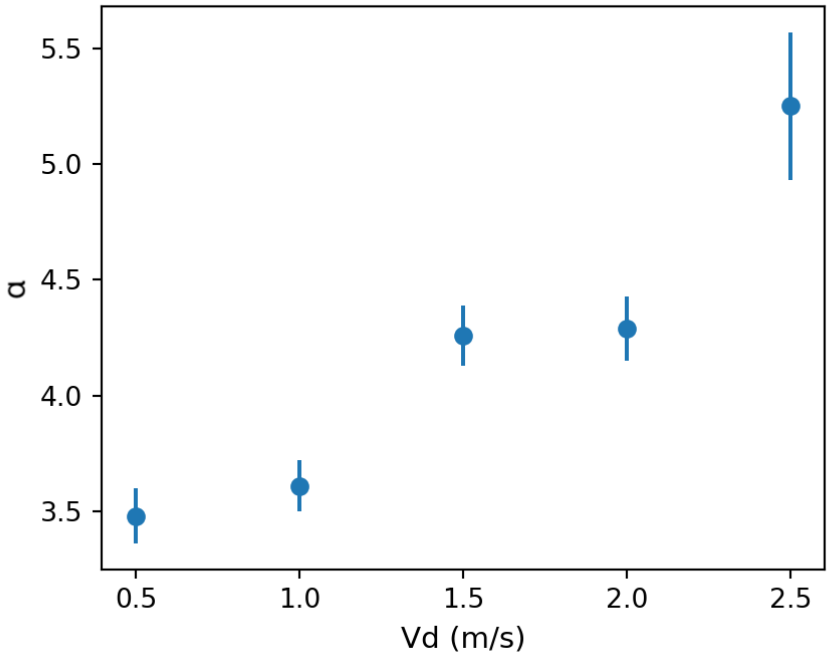


Figura 35: α resultantes de los ajustes a las leyes de potencia para cada uno de los valores de V_D. Las barras de error representan el desvío estándar.

Por último, visualizando la (figura 34) se ven los ajustes alcanzados. Los mismos nuevamente se consiguieron con un valor de significancia p superior a 0.1, permitiendo validar la aproximación a esta distribución. Se puede ver como para valores elevados de V_d las curvas se asemejan mucho para los limites de las colas. Mostrando que aumentar la velocidad no disminuye fuertemente la cantidad de bloqueos o estancamientos que hay una vez que las partículas compiten con velocidades de autopropulsión elevadas. Adicionalmente al ver la (figura 35) se ve la tendencia de suba en los α, volviendo a marcar que a mayores velocidades la probabilidad de estancamientos muy prolongados se reduce rápidamente.

6. Conclusiones

Se desarrolló un sistema altamente configurable, con el potencial para simular un sin fin de diferentes escenarios donde haya partículas autopropulsadas con geometrías arbitrarias. La posibilidad de elegir el tipo de geometrías, parámetros de las fuerzas y distintas variables del sistema con facilidad vuelven a este desarrollo una potente herramienta que pueda ser utilizada en futuros trabajos de investigación.

Se observó que eliminar el ruido por completo en las condiciones estudiadas lleva a un estancamiento completo del sistema, que imposibilita llevar adelante las simulaciones. Así también el incremento de η por encima de 0.75 resulta contraproducente, puesto que el desorden generado disminuyó los valores del caudal. Esto se ve alineado con el valor propuesto en [15] a pesar de las diferentes geometrías estudiadas.

Se concluyó que para geometrías como las estudiadas, con aspecto cuadrangular y simétricas, la intensidad por orientarse (S_D) aportó variaciones del caudal menores al 10 %. La variación de S_D impactó la simulación en menor medida que el resto de los parámetros estudiados. Tanto la ausencia por una preferencia como la intensidad máxima probada para esta alineación demostraron reducir el flujo levemente, pero no de una manera muy significativa.

Se vio un incremento constante y marcado en el caudal al incrementar la apertura (W) de la salida, pero así también un aumento en la variabilidad del caudal. A menor apertura el nivel del mismo se registró estable, mientras que para aperturas amplias el mismo fluctuó.

Los tiempos de salida entre las distintas partículas presentaron una distribución de cola pesada como las vistas en una ley de potencia. Se pudo aproximar la distribución de los mismos para la cola a esta ley con un valor $p > 0.1$, tomando como válida esta aproximación. Estos resultados se alinean con estudios empíricos realizados por Pastor et al. [6] para la distribución de los mismos. Sin embargo no se vio manifestado el efecto “faster is slower” presente en el mismo, que no se vio presente en el estudio bajo estos parámetros y geometrías estudiadas.

Referencias

[1] Hankin, B. D., and Richard A. Wright.*Passenger Flow in Subways* Journal of the Operational Research Society 9.2 (1958): 81-88.

[2] Alonso-Marroquin, F. *Spheropolygons: A new method to simulate conservative and dissipative interactions between 2D complex-shaped rigid bodies* EPL (Europhysics Letters) 83.1 (2008): 14001.

[3] Luding, Stefan. *Theory and Simulation of Granular Materials with Hard Spheres* Granul Matter 148 (2003): 202-219.

[4] Alonso-Marroquin, Fernando, et al. *Simulation of counterflow pedestrian dynamics using spheropolygons* Physical Review E 90.6 (2014): 063305.

[5] Kirchner, Ansgar, and Andreas Schadschneider. *Simulation of evacuation processes using a bionics-inspired cellular automaton model for pedestrian dynamics*, Physica A: statistical mechanics and its applications 312.1-2 (2002): 260-276.

[6] Pastor, José M., et al. *Experimental proof of faster-is-slower in systems of frictional particles flowing through constrictions* , Physical Review E 92.6 (2015): 062817.

[7] Garcimartín, Ángel, et al.*Flow of pedestrians through narrow doors with different competitiveness* Journal of Statistical Mechanics: Theory and Experiment 2016.4 (2016): 043402.

[8] Hidalgo, R. Cruz, Daniel R. Parisi, and Iker Zuriguel. *Simulating competitive egress of noncircular pedestrians* Physical Review E 95.4 (2017): 042319.

[9] Dobrohotoff, Peter B., et al. *Optimal description of two-dimensional complex-shaped objects using spheropolygons* Granular Matter 14.5 (2012): 651-658.

[10] Helbing, Dirk, Illés Farkas, and Tamas Vicsek. *Simulating dynamical features of escape panic* Nature 407.6803 (2000): 487-490.

[11] 11 Dec, 2020 GeekForGeeks *How to check if a given point lies inside or outside a polygon?* https://www.geeksforgeeks.org/how-to-check-if-a-given-point-lies-inside-a-polygon/

[12] Grubmüller, Helmut, et al. *Generalized Verlet Algorithm for Efficient Molecular Dynamics Simulations with Long-range Interactions* Molecular Simulation 6.1-3 (1991): 121-142.

[13] 2002- Malcolm Kesson. *Shortest Distance from a Point to a Line* http://www.fundza.com/vectors/point2line/index.html

[14] Clauset, Aaron, Cosma Rohilla Shalizi, and Mark EJ Newman *Powerlaw distributions in empirical data* SIAM review 51.4 (2009): 661-703.

[15] Parisi, Daniel R., Raúl Cruz Hidalgo, and Iker Zuriguel *Active particles with desired orientation flowing through a bottleneck* Scientific reports 8.1 (2018): 1-9.