TP/3: Dinámica Molecular Dirigida por Eventos:

Movimiento Browniano

Grupo 5

- Golmar, Agustín
- Lobo, Daniel Alejandro

Fundamentos

A destacar...

- Simulación Dinámica molecular: avanza cada cierto delta_t (colisiones)
- Cada partícula posee movimiento, radio y masa.
- Las interacciones son totalmente elásticas.
- Las partículas se mueven a velocidad constante (MRU).
- No hay gravedad.
- Las partículas generadas no pueden estar superpuestas.
- Las actualizaciones provistas durante una colisión/evento se separan en:
  - Colisión contra el recipiente
  - Colisión entre partículas
- Las partículas se esparcen con distribución uniforme.
- Contorno no-periódico.
- Movimiento browniano: partículas de igual masa excepto la partícula distinguida.

Ecuaciones...

- **Se verifica la superposición mediante:** $(x_i - x_j)^2 + (y_i - y_j)^2 > (r_i + r_j)^2$
- **Se determina la colisión contra el recipiente:**
$$0 < v_x \rightarrow x_r - r = x_0 + t\,v_x$$
$$v_x < 0 \rightarrow x_l + r = x_0 + t\,v_x$$
- **Para la colisión entre partículas se utiliza:** $(x_i - x_j)^2 + (y_i - y_j)^2 = (r_i + r_j)^2$
- **El tiempo de colisión resulta ser:** $t_c = -\frac{\Delta v \cdot \Delta r + \sqrt{\Delta}}{\Delta v \cdot \Delta v}$ $\Delta = (\Delta v \cdot \Delta r)^2 - (\Delta v \cdot \Delta v)(\Delta r \cdot \Delta r - \sigma^2)$
- **El impulso durante la colisión es:** $J = (J_x, J_y) = J_m(\Delta x, \Delta y)$ $J_m = \frac{2 m_i m_j (\Delta v \cdot \Delta r)}{\sigma^2 (m_i + m_j)}$

Implementación

Detalles importantes...

- **Formato de archivos:**

  El archivo de salida solo contiene eventos, el estado de las partículas y los ID de las partículas que colisionaron:

  <t0> <id0> ... <idn>

  <x> <y> <vx> <vy>

  ...

  El archivo de animación tiene 4 columnas para ser leído por Ovito: posición X, posición Y, radio, módulo de la velocidad (para colorear).

- **Configuración por JSON**
- **Dos modos de ejecución: GENERATE y SIMULATE**

Modelos

Modelos Relevantes y de Implementación

- **Massive Generator**
- **Massive Particle**
- **Event Driven Simulation**
- **Particle Collider**
- **Collision**

Massive Generator

- Permite agregar partículas ***distinguidas***.

```
public class MassiveGenerator implements Generator {

    protected final List<MassiveParticle> particles;
    protected final Consumer<MassiveParticle> spy;
    protected final double length;
    protected final double radius;
    protected final double speed;
    protected final double mass;
    protected double availableArea;

    public MassiveGenerator(final Builder builder) {

    public static Builder over(final double length) {

    public MassiveGenerator destroy() {

    public List<MassiveParticle> getParticles() {

    public MassiveGenerator create(final int size) {

    public double getLength() {

    protected boolean addWithoutCollision(

    public static class Builder {
}
```

Massive Particle

- Implementa los algoritmos de ***colisión*** y de ***detección***.

```
public class MassiveParticle extends MobileParticle {

    protected final double mass;

    public MassiveParticle(..

    public double getMass() {..

    public MassiveParticle move(final double Δt) {..

    public boolean overlap(final MassiveParticle particle) {..

    public double timeToVerticalCollision(final double length) {..

    public double timeToHorizontalCollision(final double length) {..

    public double timeToCollide(final MassiveParticle particle) {..

    public MassiveParticle bounce(final double vx, final double vy) {..

    public MassiveParticle verticalCollide() {..

    public MassiveParticle horizontalCollide() {..

    public Pair<MassiveParticle> collide(final MassiveParticle particle) {..
}
```

Event Driven Simulation

- Es independiente del ***tipo*** de evento, es decir, es un motor ***genérico*** de simulación.
- Utiliza una ***Priority Queue***.

```
public class EventDrivenSimulation {

    protected final PriorityQueue<Event> events;
    protected final EventSystem<? extends Event> system;
    protected final long maxEvents;
    protected final double maxTime;

    public EventDrivenSimulation(final Builder builder) {..

    public EventDrivenSimulation run() {..

    public static Builder of(final EventSystem<? extends Event> system) {..

    public static class Builder {..
}
```

ParticleCollider

- Genera eventos de tipo ***Collision***.
- Detecta la validés de los mismos.

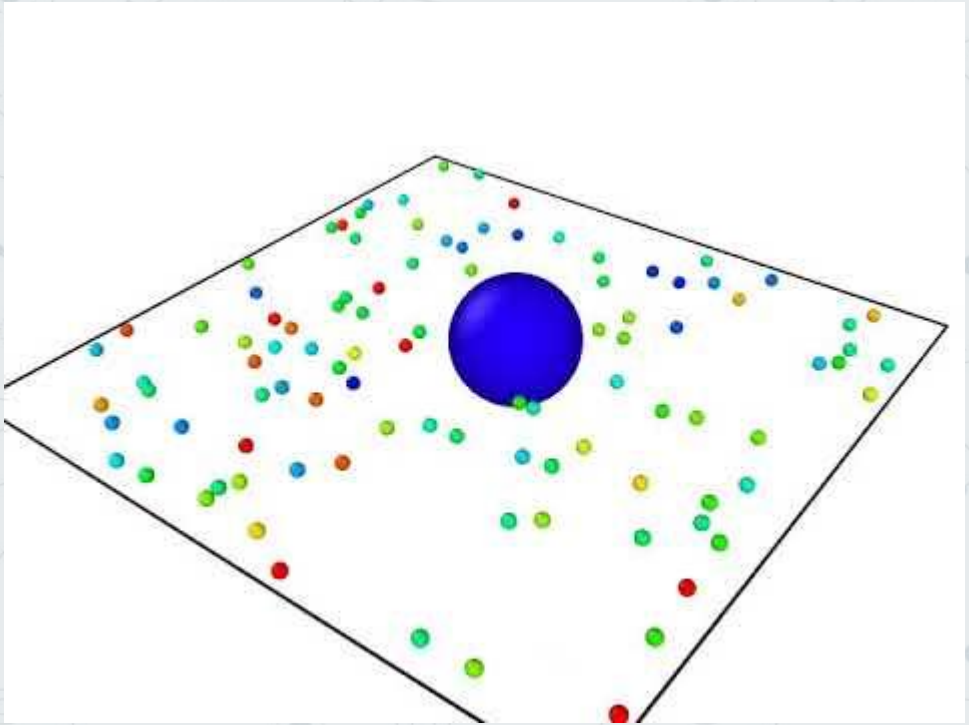
```
public class ParticleCollider implements EventSystem<Collision> {

    protected final long [] wallCollisions;
    protected final long [] particleCollisions;

    protected final BiConsumer<Collision, List<MassiveParticle>> spy;
    protected final List<MassiveParticle> particles;
    protected final Generator generator;
    protected final int size;
    protected final double length;

    public ParticleCollider(final Builder builder) {..

    public List<Collision> bootstrap() {..

    public List<Collision> evolve(final Event event, final double baseTime) {..

    public boolean isValid(final Event event) {..

    protected List<Collision> imminentCollisions(final double baseTime) {..

    protected CollisionType inferType(..

    protected boolean isStale(..

    protected double impactTime(..

    public static Builder of(final int size) {..

    public static class Builder {..
}
```

Collision

- Almacena los IDs de las partículas.
- El tipo de ***colisión***.
- El tiempo.
- La cantidad de colisiones actuales.

```
public class Collision implements Event {

    protected final List<Long> collisions;
    protected final List<Integer> ids;
    protected final CollisionType type;
    protected final double time;
    protected final double baseTime;

    public Collision(final Builder builder) {

    public double getTime() {

    public double getBaseTime() {

    public List<Long> getCollisions() {

    public List<Integer> getIDs() {

    public CollisionType getType() {

    public List<MassiveParticle> collide(

    public static Builder type(final CollisionType type) {

    public static class Builder {
}
```

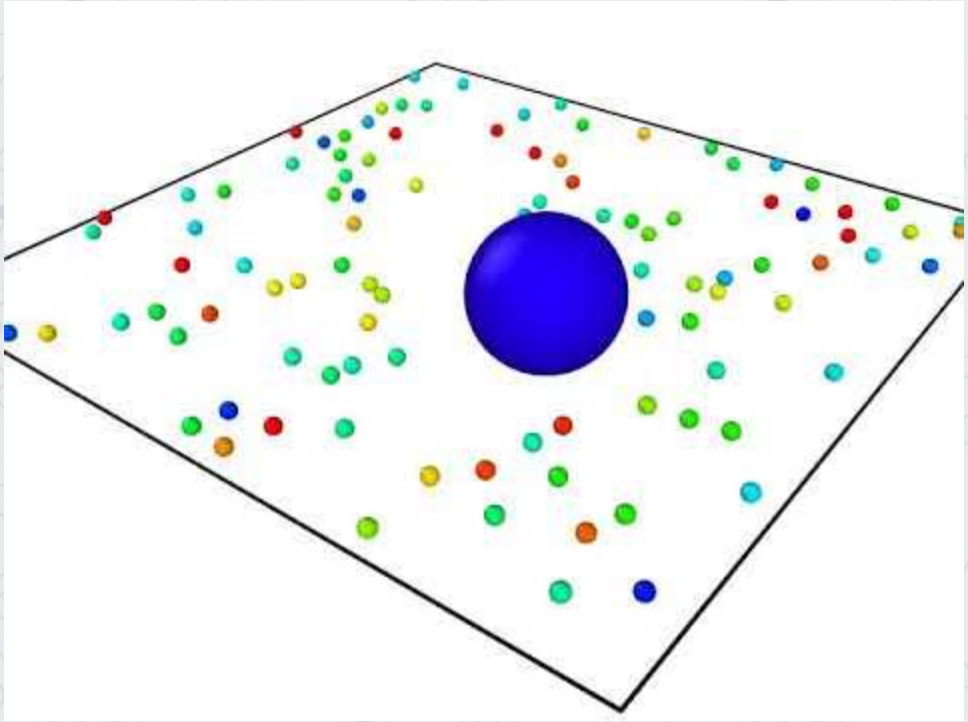
Resultados



```
"n"             : "100",

"events"        : "20000",
"tmax"          : "60.0",
"l"             : "0.5",

"r"             : "0.005",
"speed"         : "0.1",
"mass"          : "0.0001",
"rbig"          : "0.05",
"massbig"       : "0.1",
"xbig"          : "0.25",
"ybig"          : "0.25",

"inputfile"     : "input.txt",
"outputfile"    : "output.txt",
"deltat"        : "0.025"
```

Usando los datos del ejemplo

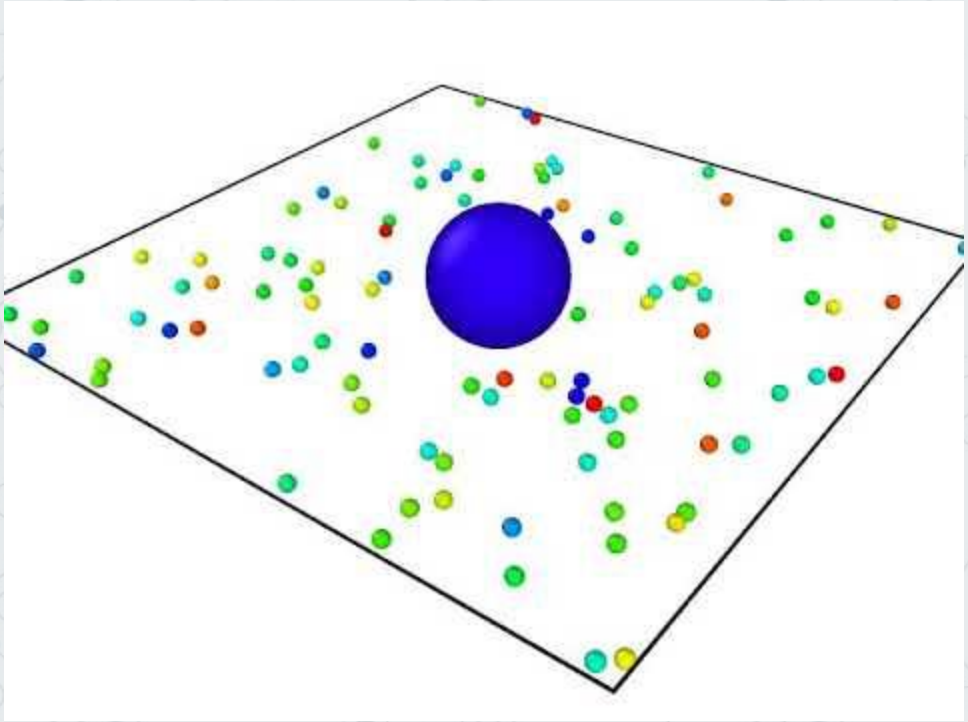
```
"events"        : "20000",
"tmax"          : "60.0",
"l"             : "0.5",

"inputfile"     : "input.txt",
"outputfile"    : "output.txt",
"deltat"        : "0.025"
```



```
"n"                 : "100",

"events"            : "20000",
"tmax"              : "60.0",
"l"                 : "0.5",

"r"                 : "0.005",
"speed"             : "0.3",
"mass"              : "0.0001",
"rbig"              : "0.05",
"massbig"           : "0.1",
"xbig"              : "0.25",
"ybig"              : "0.25",

"inputfile"         : "input.txt",
"outputfile"        : "output.txt",
"deltat"            : "0.025"
```

```
"events"            : "20000",
"tmax"              : "60.0",
"l"                 : "0.5",

"inputfile"         : "input.txt",
"outputfile"        : "output.txt",
"deltat"            : "0.025"
```

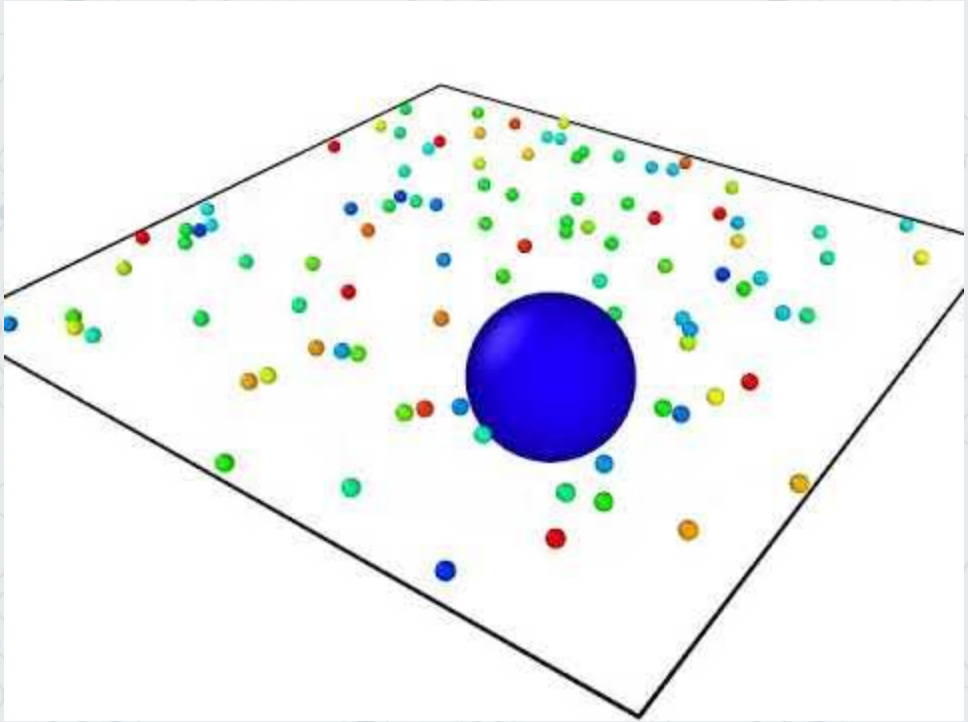
Triple velocidad



La mitad de la velocidad

```
"n"             : "100",

"events"        : "20000",
"tmax"          : "60.0",
"l"             : "0.5",

"r"             : "0.005",
"speed"         : "0.05",
"mass"          : "0.0001",
"rbig"          : "0.05",
"massbig"       : "0.1",
"xbig"          : "0.25",
"ybig"          : "0.25",

"inputfile"     : "input.txt",
"outputfile"    : "output.txt",
"deltat"        : "0.025"
```

```
"events"        : "20000",
"tmax"          : "60.0",
"l"             : "0.5",

"inputfile"     : "input.txt",
"outputfile"    : "output.txt",
"deltat"        : "0.025"
```



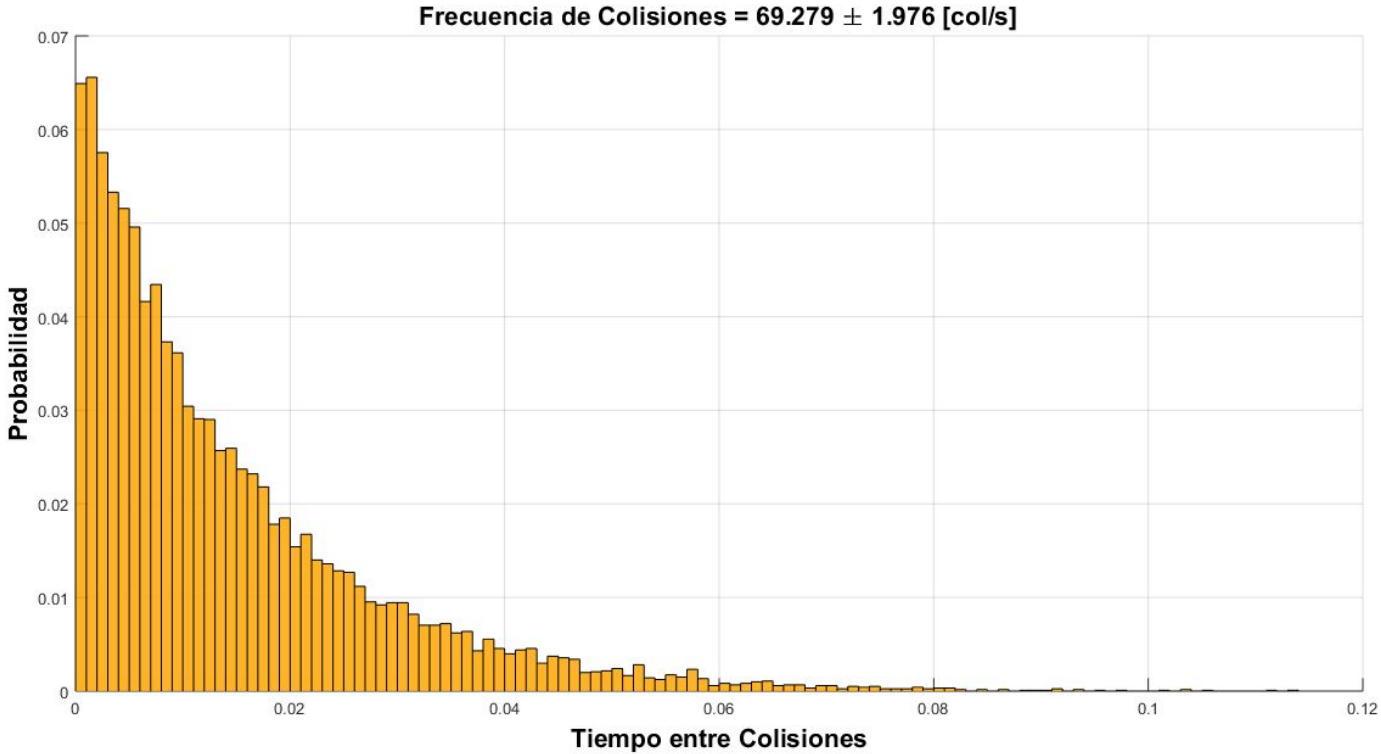
El quíntuple de la velocidad

```
"n"             : "100",

"events"        : "20000",
"tmax"          : "60.0",
"l"             : "0.5",

"r"             : "0.005",
"speed"         : "0.5",
"mass"          : "0.0001",
"rbig"          : "0.05",
"massbig"       : "0.1",
"xbig"          : "0.25",
"ybig"          : "0.25",

"inputfile"     : "input.txt",
"outputfile"    : "output.txt",
"deltat"        : "0.025"
```

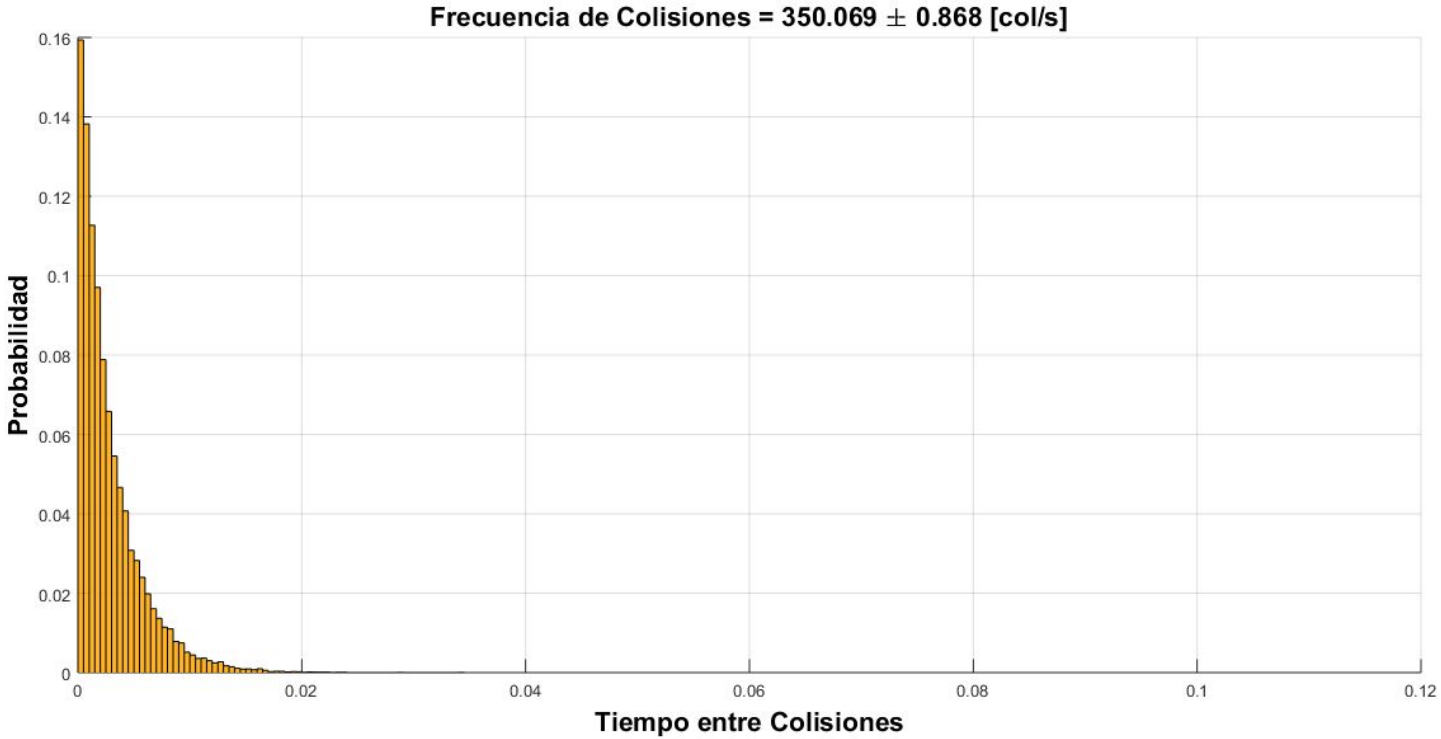
```
"events"        : "20000",
"tmax"          : "60.0",
"l"             : "0.5",

"inputfile"     : "input.txt",
"outputfile"    : "output.txt",
"deltat"        : "0.025"
```

Gráficos

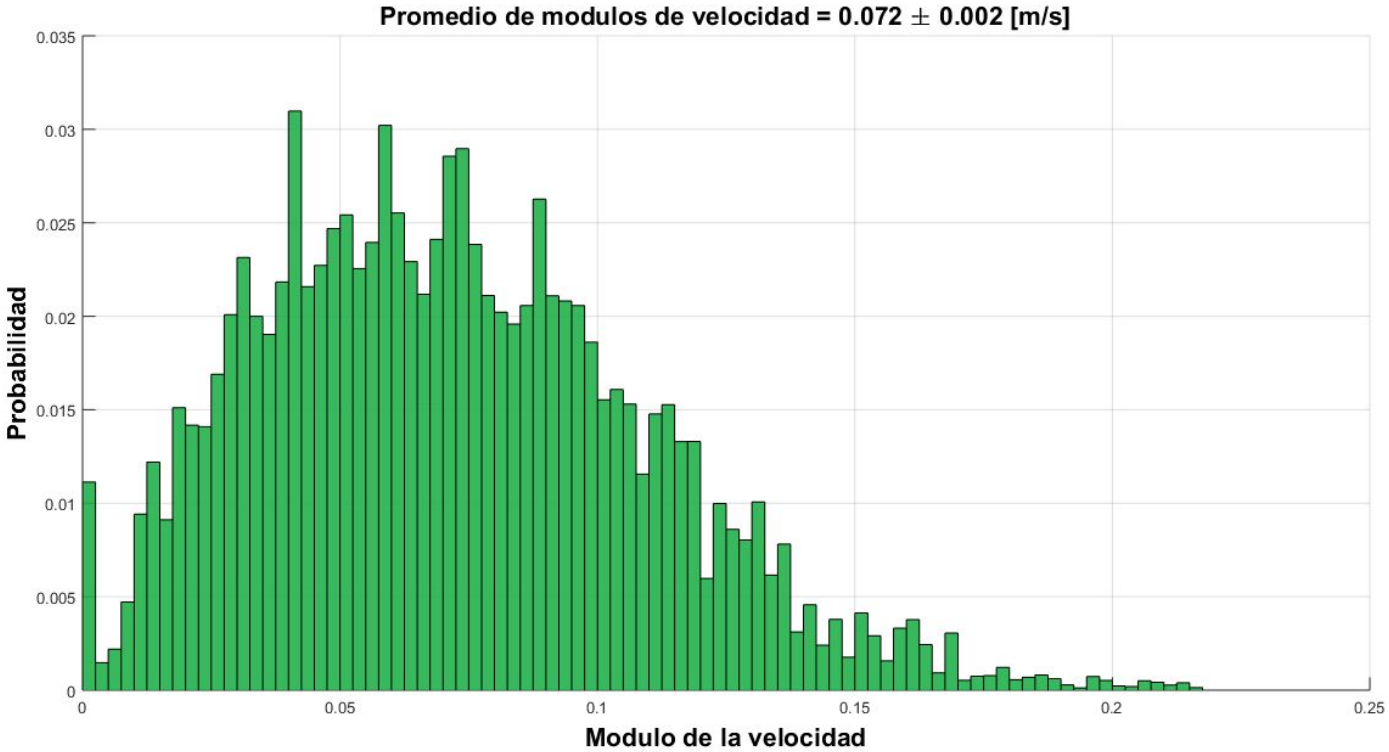
3 Simulaciones (N = 100 | V = 0.1 | 1 min)



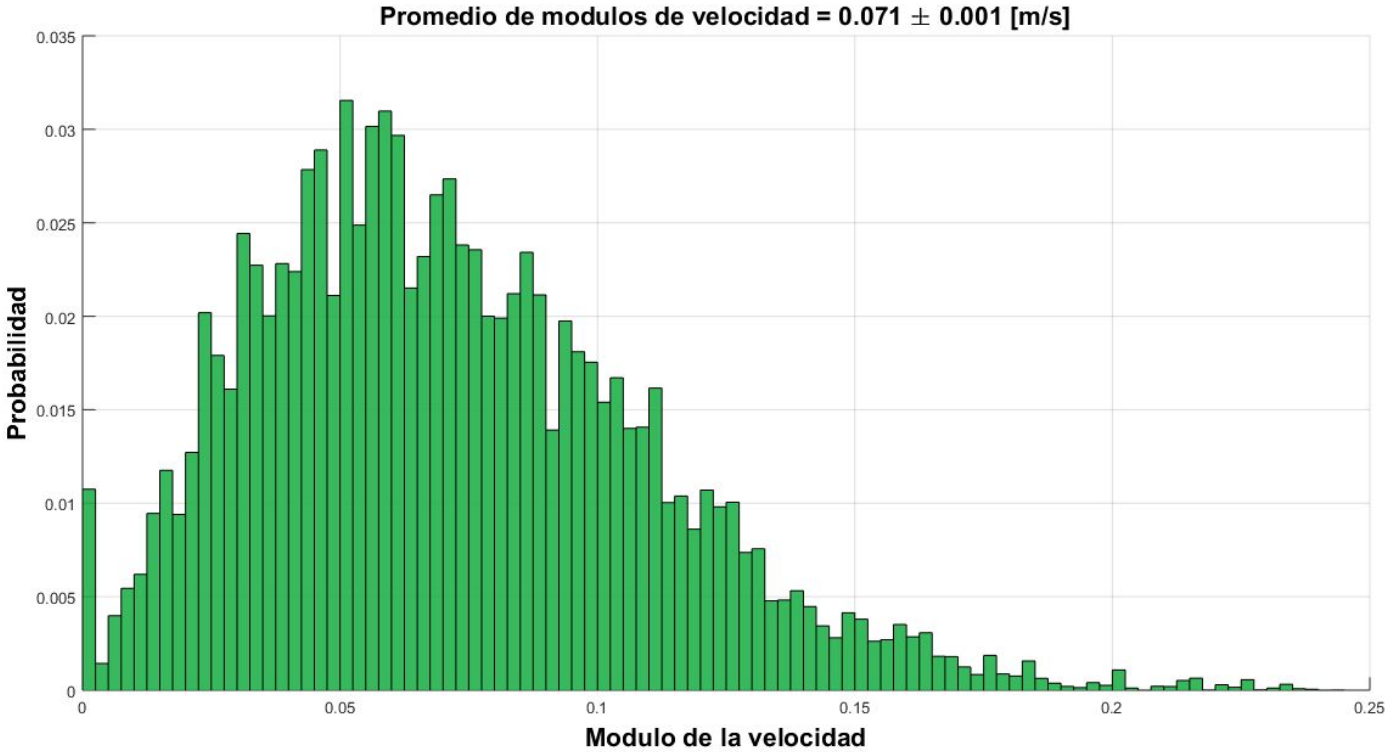
3 Simulaciones (N = 100 | V = 0.5 | 30 seg)



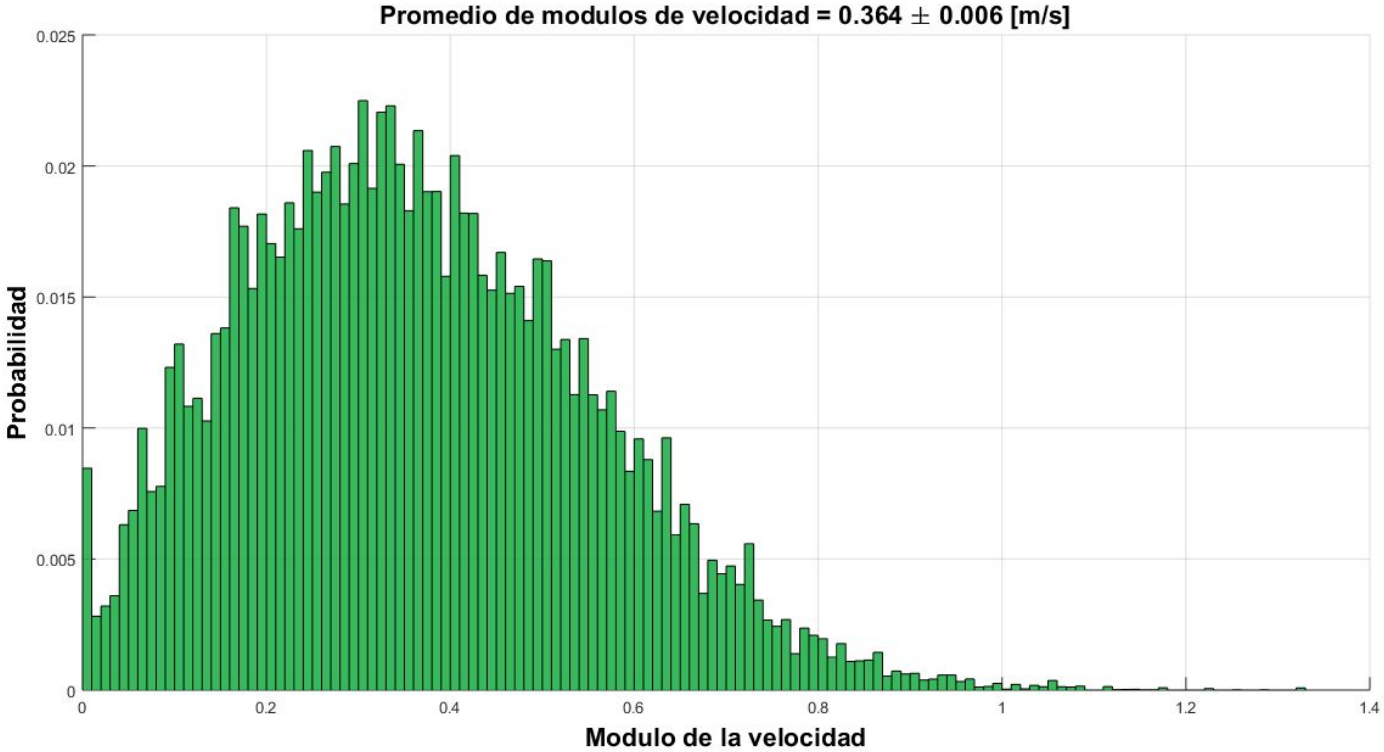
3 Simulaciones (N = 100 | V = 0.1 | 1 min): Estado Inicial (1er Tercio)



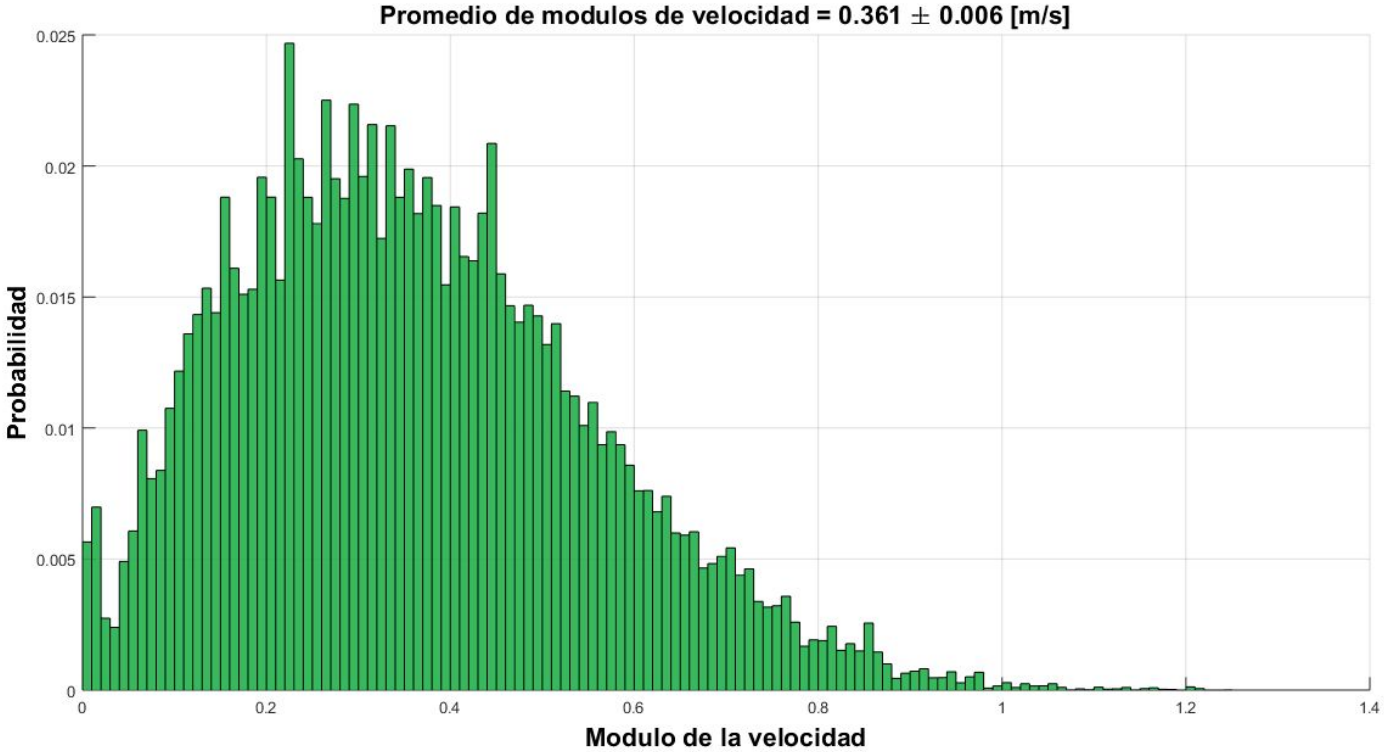
3 Simulaciones (N = 100 | V = 0.1 | 1 min): Estado Final (3er Tercio)



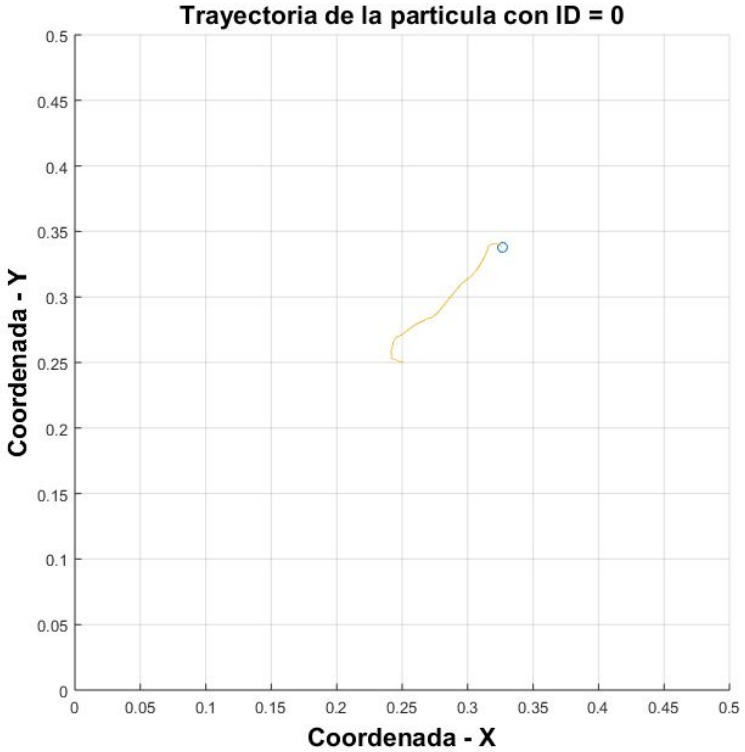
3 Simulaciones (N = 100 | V = 0.5 | 30 seg): Estado Inicial (1er Tercio)



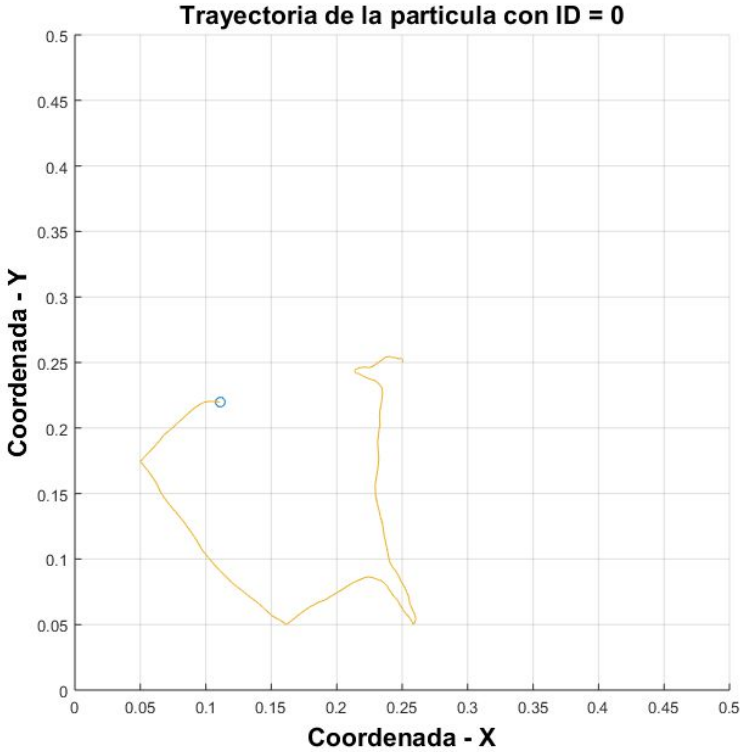
3 Simulaciones (N = 100 | V = 0.5 | 30 seg): Estado Final (3er Tercio)



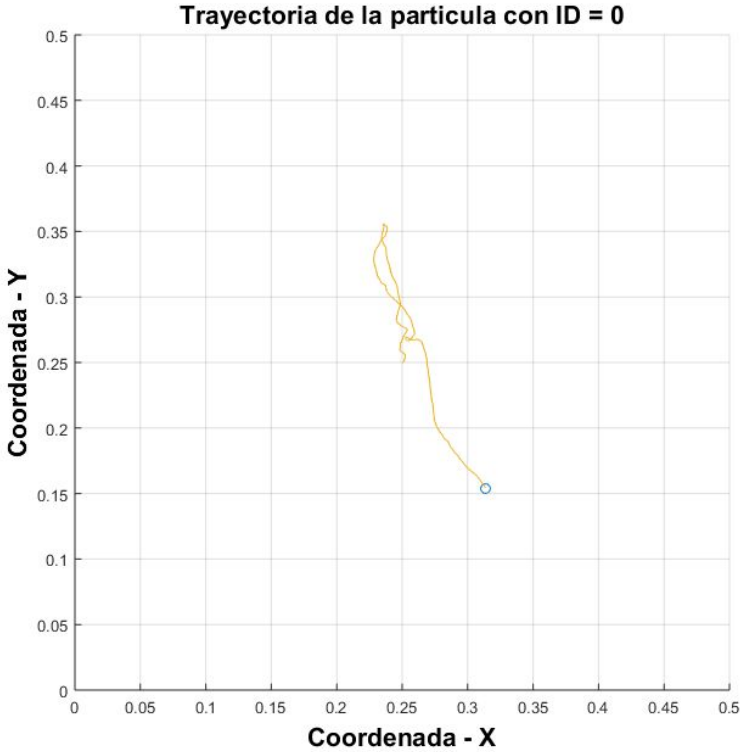
Trayectoria (N = 100 | V = 0.1 m/s | 2 min): Partícula Distinguida



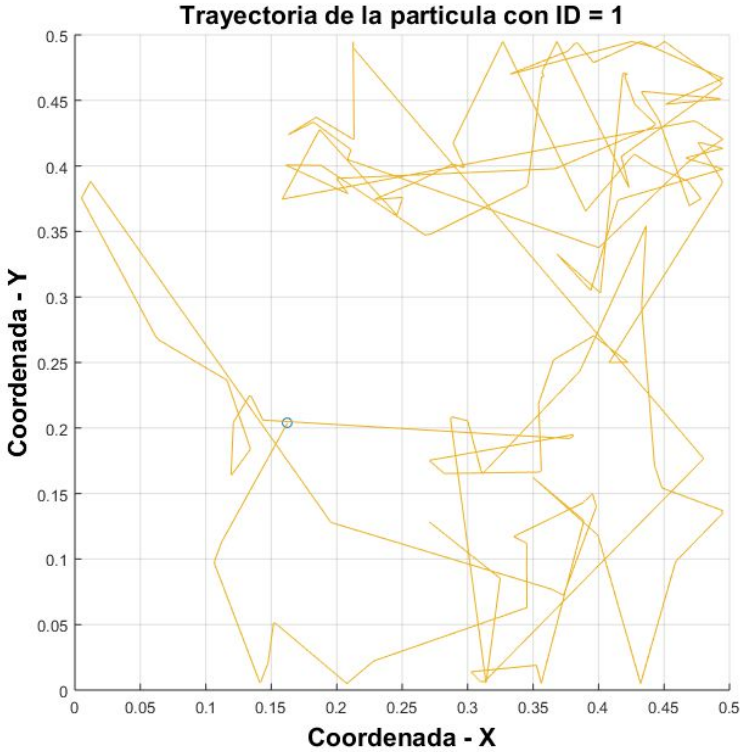
Trayectoria (N = 100 | V = 0.5 m/s | 1 min): Partícula Distinguida



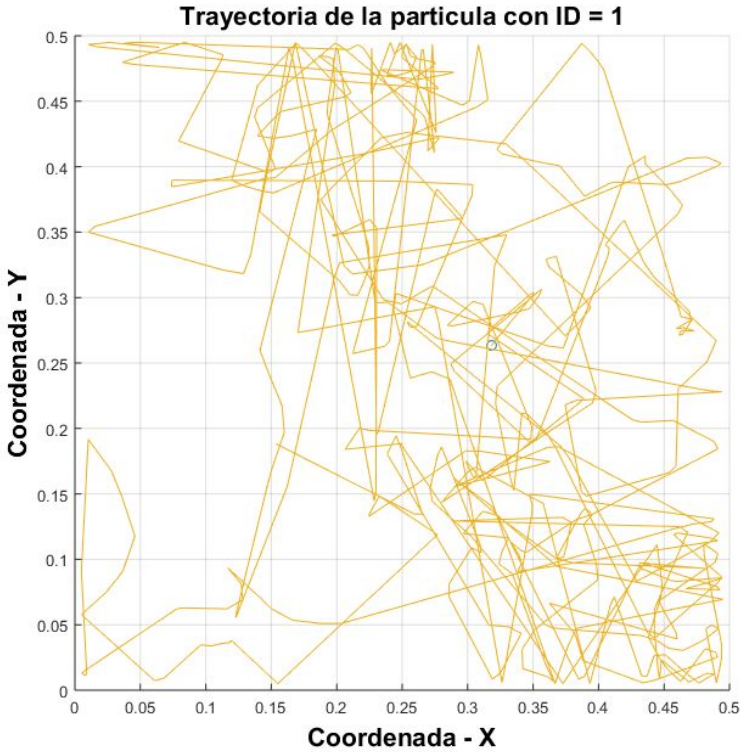
Trayectoria (N = 100 | V = 1.0 m/s | 30 seg): Partícula Distinguida



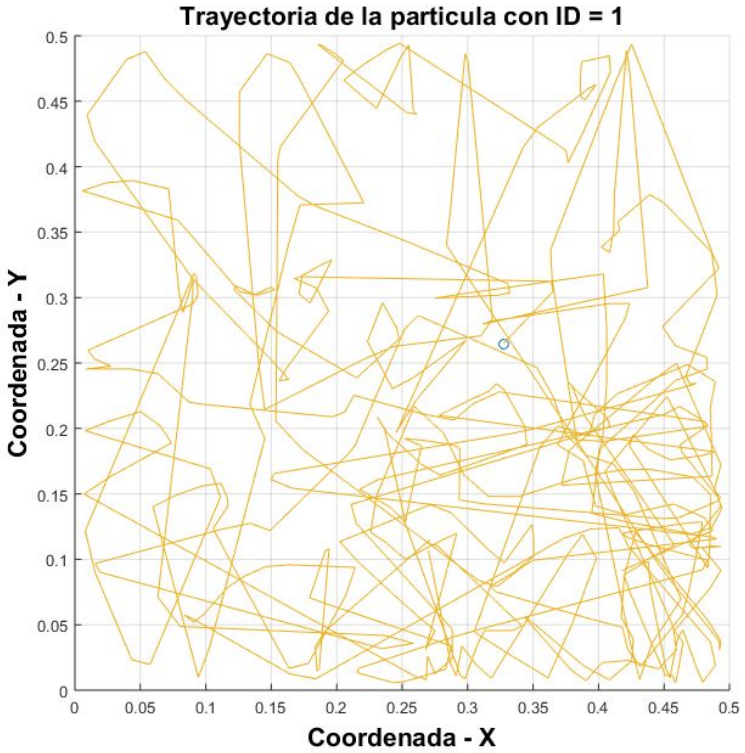
Trayectoria (N = 100 | V = 0.1 m/s | 2 min): Partícula Pequeña



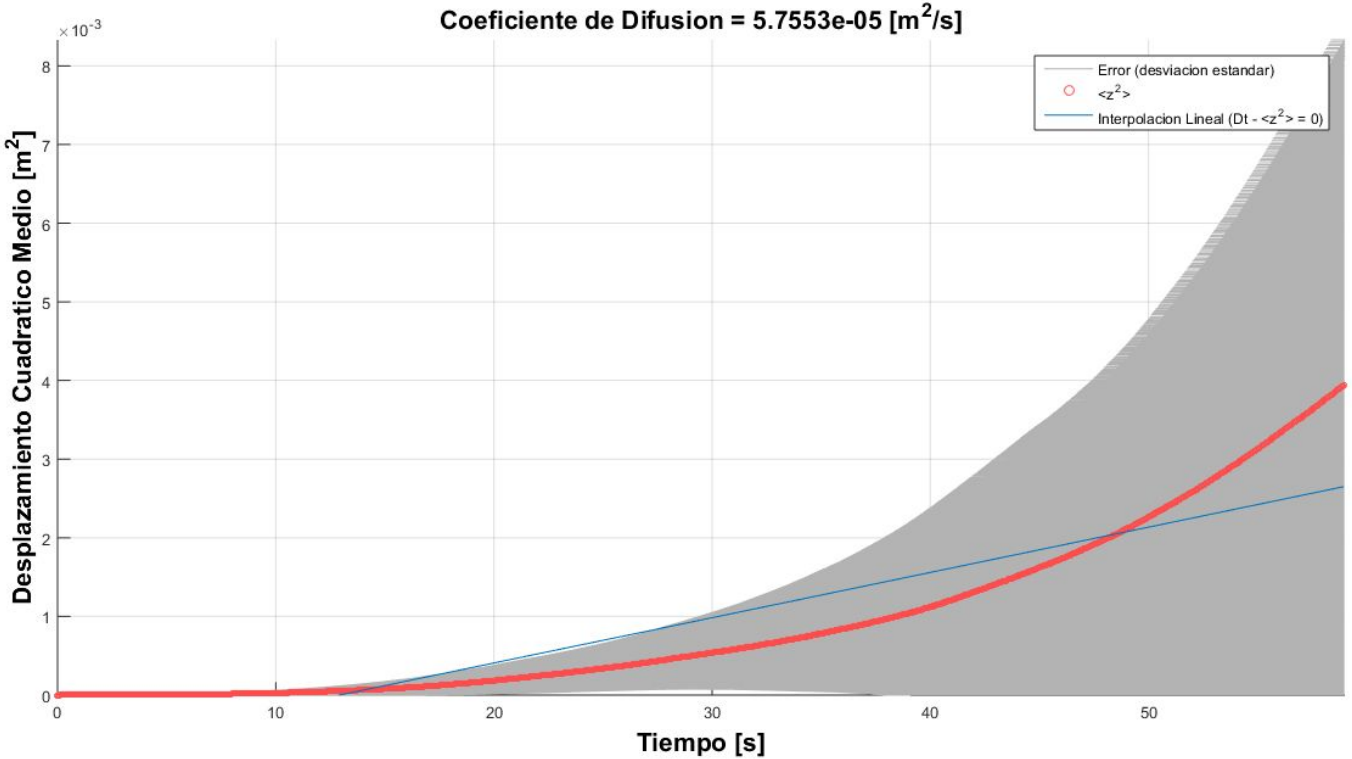
Trayectoria (N = 100 | V = 0.5 m/s | 1 min): Partícula Pequeña



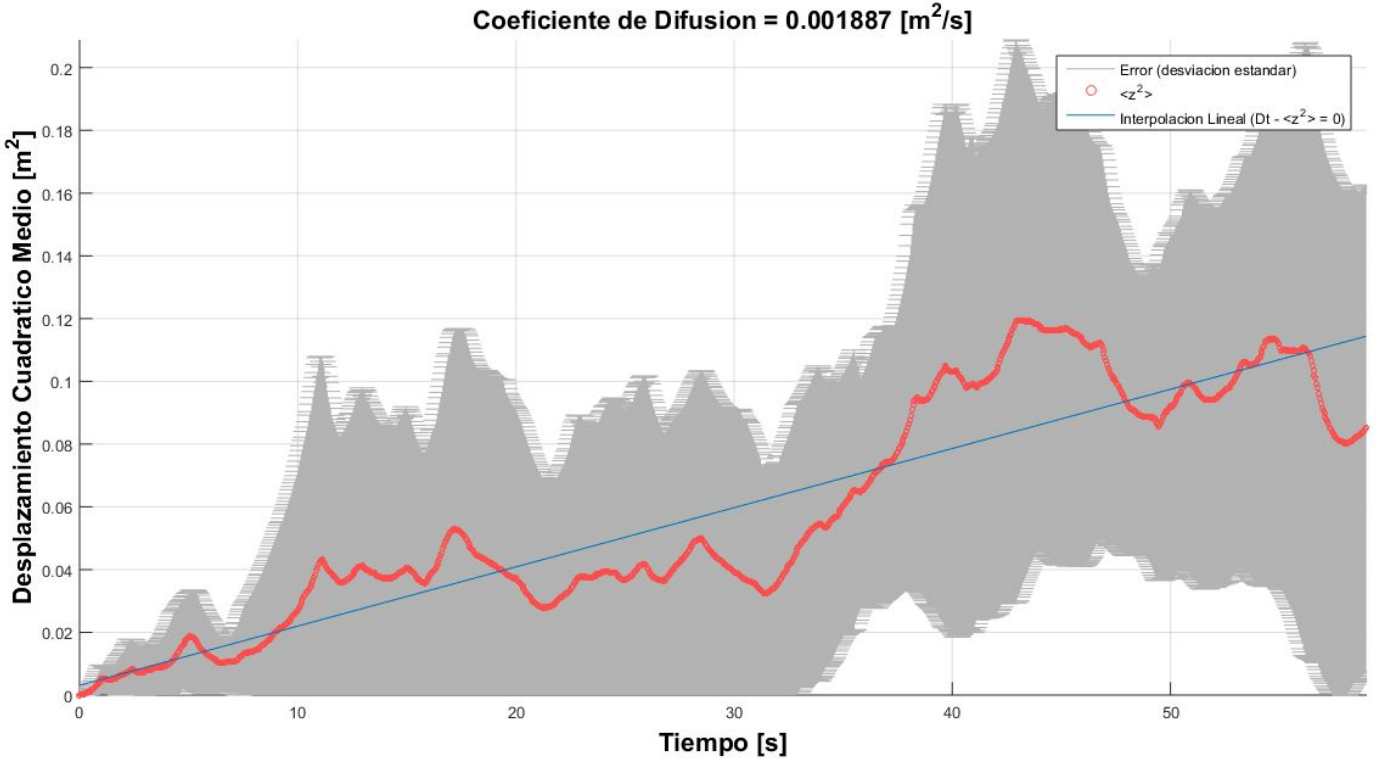
Trayectoria (N = 100 | V = 1.0 m/s | 30 seg): Partícula Pequeña



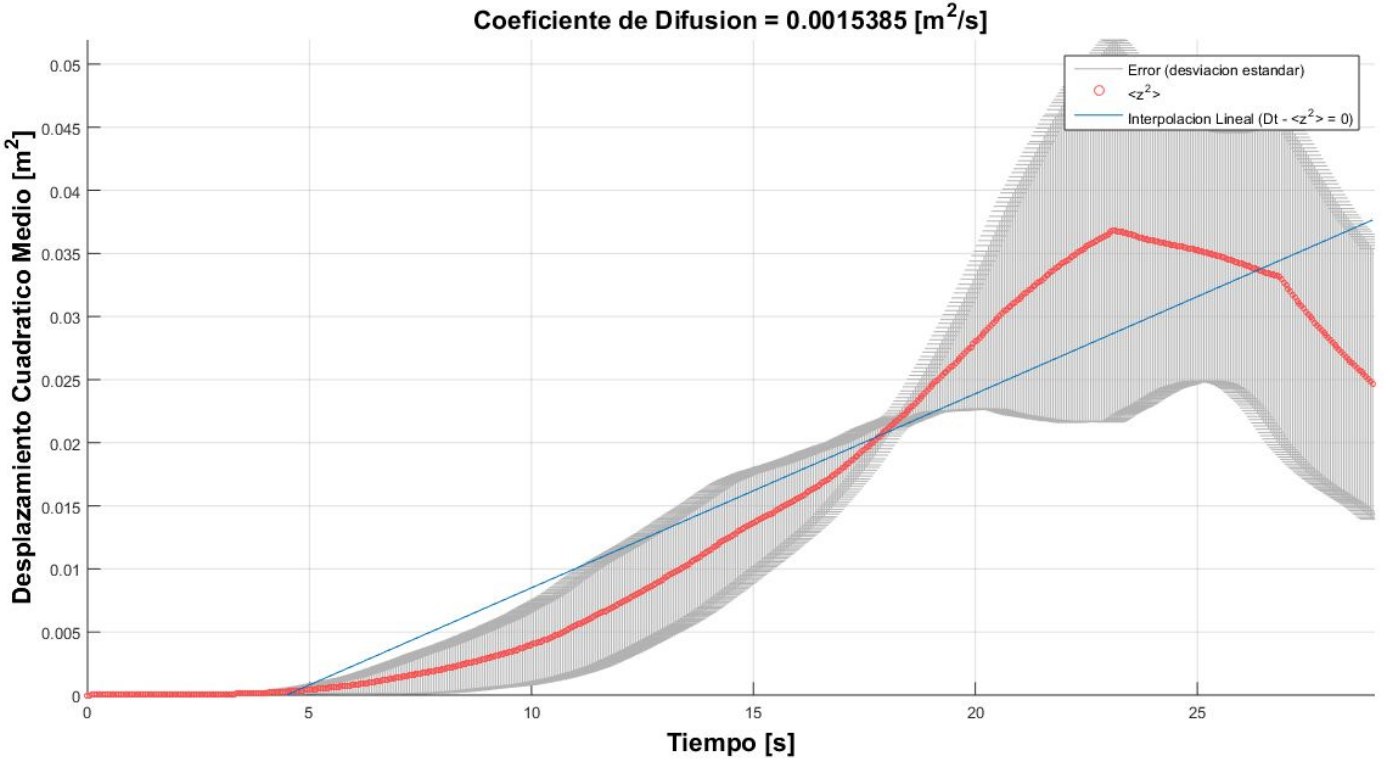
10 Simulaciones (N = 100 | V = 0.1 m/s): Partícula Distinguida



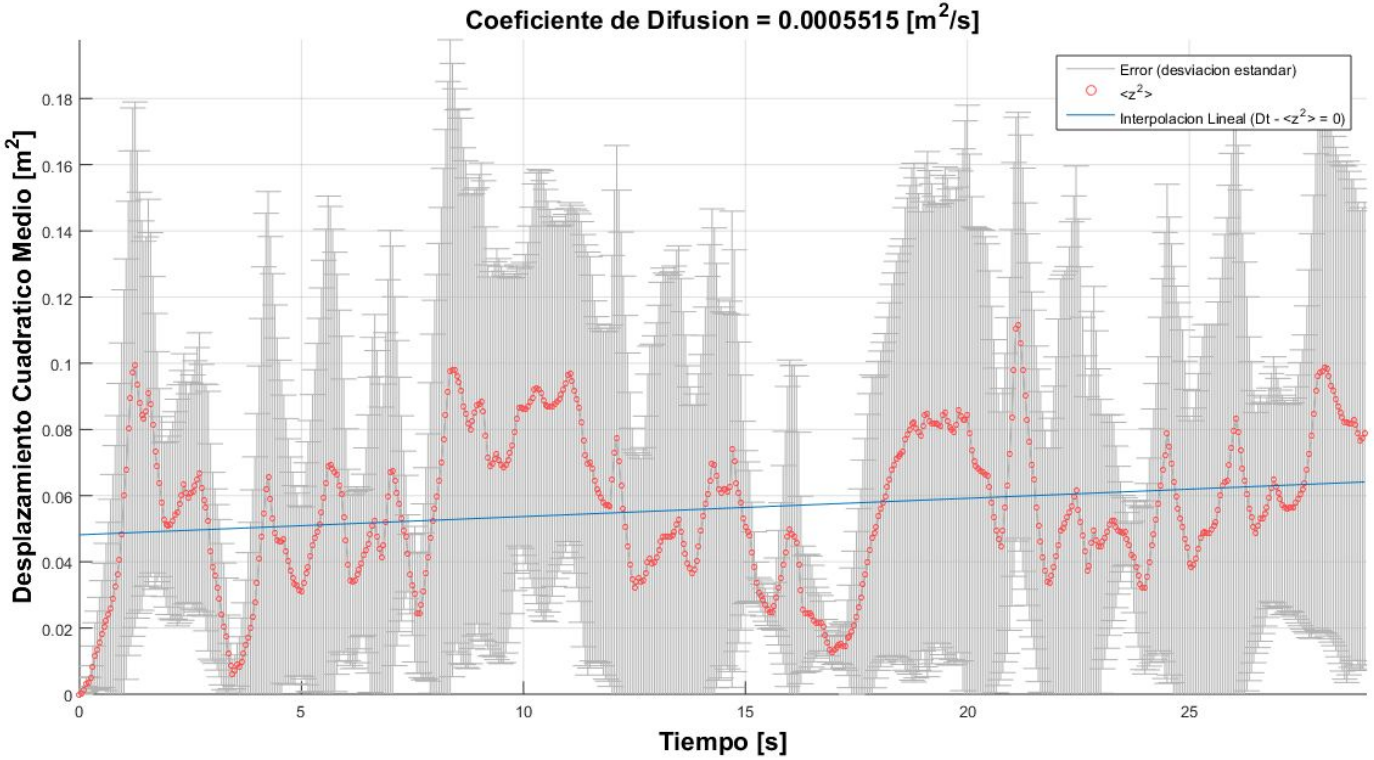
10 Simulaciones (N = 100 | V = 0.1 m/s): Partícula Pequeña



5 Simulaciones (N = 100 | V = 0.5 m/s): Partícula Distinguida



5 Simulaciones (N = 100 | V = 0.5 m/s): Partícula Pequeña



Conclusiones

Conclusiones

- A más velocidad, más frecuencia de colisiones.
- La distribución de velocidades se corresponde con la distribución de ***Maxwell-Boltzmann***. Para altas temperaturas, la curva se aplana.
- La difusión aumenta a más velocidad, pero la partícula distinguida se ve afectada por su tamaño y masa.
- A mayor velocidad, la probabilidad de encontrar una partícula en cualquier lugar del espacio aumenta.

Gracias!

Grupo 5: Golmar & Lobo