Grupo 4

Trabajo Práctico N° 4

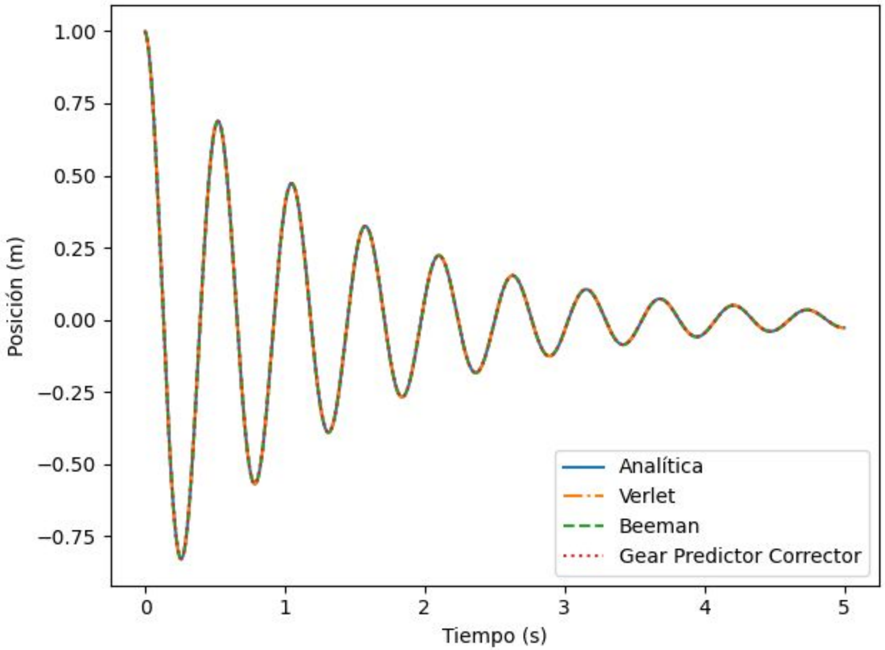
Dinámica Molecular regida por el paso temporal

- Banfi, Malena - 61008
- Caeiro, Alejo Francisco - 60692
- Fleischer, Lucas - 61153

Sistema 1

Oscilador Puntual Amortiguado

Solución analítica y numérica



Parámetros:
$\Delta t = 10^{-4}$
$m = 70$ kg
$k = 10^4$ N/m
$\gamma = 100$ kg/s
$T_f = 5$ s

Condiciones iniciales

$$r(t=0) = 1m$$

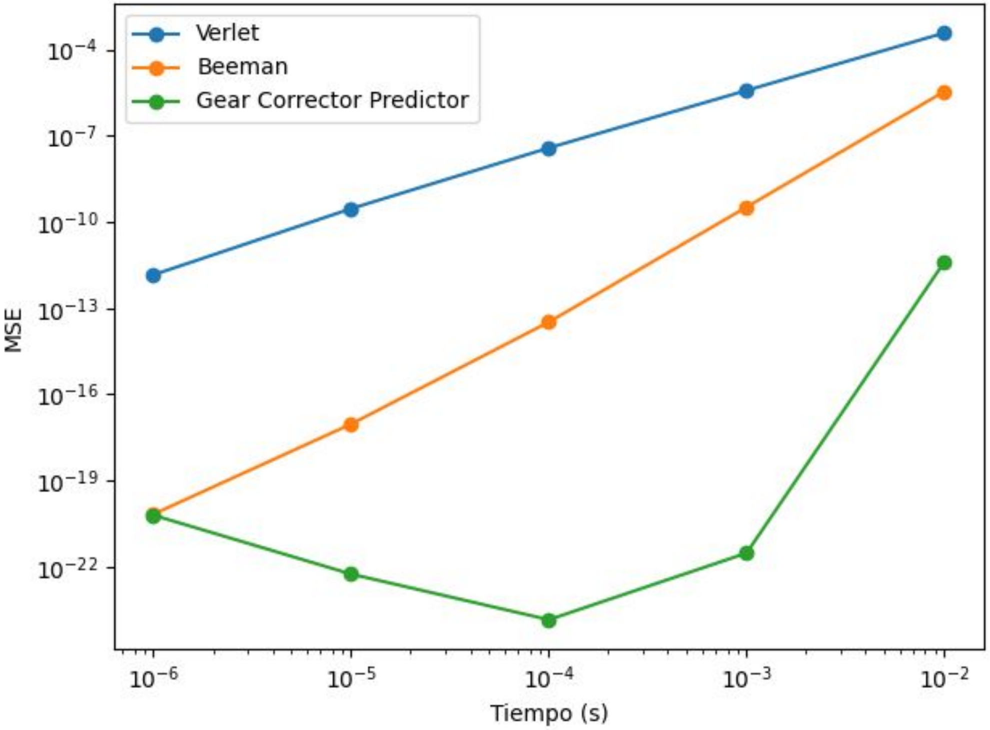
$$v(t=0) = -A\frac{\gamma}{2m}\frac{m}{s}$$

Error cuadrático medio
Gear: $5{,}57\cdot10^{-23}$ m²
Beeman: $3{,}27\cdot10^{-14}$ m²
Verlet: $3{,}70\cdot10^{-8}$ m²

Gráfico de MSE vs tiempo

<u>Integrador a utilizar:</u>

- Gear Corrector Predictor



Sistema 2
Sistema de partículas unidimensionales

Introducción

Modelo matemático

Propulsión de las partículas

$$F_i^d = (u_i - v_i)/\tau$$

Ecuación de movimiento

$$ma_i = F_i^d + \sum_j F_{ij}$$

Colisión elástica entre partículas

$$F_{ij} = \mathcal{K}(|x_j - x_i| - 2r)\mathrm{sign}[x_j - x_i]$$

Donde:

$\mathcal{K}$ constante de fuerza,
ai es la aceleración,
vi es la velocidad,
ui es la velocidad límite,
τ es el tiempo de reacción característico.

Implementación

Diagrama UML

Pseudocódigo

Particle.java

```java
class Particle{

  int id
  double x, y , x2, x3, x4, x5
  double radius, mass, forceX, forceY
  double velX, velY, u

  boolean collidesWith(Particle p, Double dt){
    if(movingAway)
      return false

    return goingToCollideInTime(dt)
  }

}
```

Pseudocódigo

GearAlgorithm.java

```java
class GearAlgorithm {

  List<Particle> particles
  double dt

  List<Particle> apply(){

    List<Particle> newParticles = [];

    for(Particle p : particles){
      double[] r = predict()
      double deltaR2 = evaluate(r[2])
      Particle newParticle = correct(r, deltaR2)

      newParticles.add(newParticle)
    }

    return newParticles
  }

}
```

GearAlgorithm.java

```java
double[] predict() {
  rp = r + r1 * dt + ...+ r5 * pow(dt, 5) / factorialNumber(5)
  r1p = r1 + r2 * dt + ...+ r4 * pow(dt, 5) / factorialNumber(5)
  ...
  r5p = r5

  return [rp, r1p, r2p, r3p, r4p, r5p]
}

double evaluate(double accelerationPred){
  deltaA = particleMovementEquation(newParticle, particles) - accelerationPred
  deltaR = deltaA * pow(dt, 2) / factorialNumber(2);

  return deltaR
}

Particle correct(double r[], deltaR){
  rc = []
  q = 0
  for(double ri : r ){
    rc[i] = ri + (alpha*deltaR* (factorialNumber(q)/pow(dt, q)
    q++
  }

  return newParticle(rc)
}
```

Pseudocódigo

OneDimensionalSystem.java

```java
class OneDimentionalSystem {

  GearAlgorithm gear
  List<Particle> particles
  double dt
  double L
  double r
  double k

  double getCollisionForce(Particle p1, Particle p2)
  double getPropulsionForce(Particle p)
  double particleMovementEquation(Particle p, List<Particle> particles)

}
```

Pseudocódigo

main_density_graphics.py

```
data = [density, velocity]
orderedData = data.orderAsc()
windowSize = 500
dataAvgWindow = slidingWindow(orderedData, windowSize)
plot (dataAvgWindow)
```

Simulación

Sistema a simular

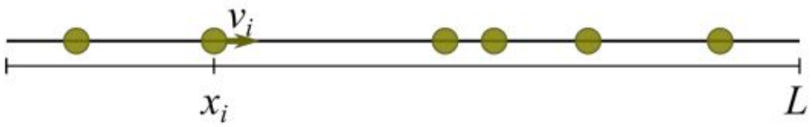
- *N* partículas autopropulsadas con una velocidad inicial de u_i, radio *r* y masa *m* constantes.
- Las partículas se mueven sobre una línea imaginaria de longitud *L* fija y una velocidad v_i.
- Paso temporal variable Δt y Δt_2 fijo para imprimir el estado del sistema.
- Tiempo de simulación T_f.
- Método de integración: Gear Corrector Predictor.

Parámetros fijos:

- $r = 2.25cm$
- $L = 135cm$
- $m = 25g$
- $u_i \in$ Uniforme $[9,12]$
- $T_f = 180s$
- $\Delta t_2 = 10^{-1}s$
- $k = 2500 \frac{g}{s^2}$

Parámetros variables:

- $N \in [5, 10, 15, 20, 25, 30]$
- $\Delta t \in [10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}]$



Observables

Phi:

$$\Phi^k(t) = \sum_{i=1}^{N_b} ||r_i^{k+1}(t) - r_i^k(t)||$$

Velocidad Promedio:

$$v_p = \frac{1}{N}\sum_{i=1}^{N} v_i(t)$$

Densidad individual:

$$p_i = \frac{1}{d_{ij} + d_{ik}}$$

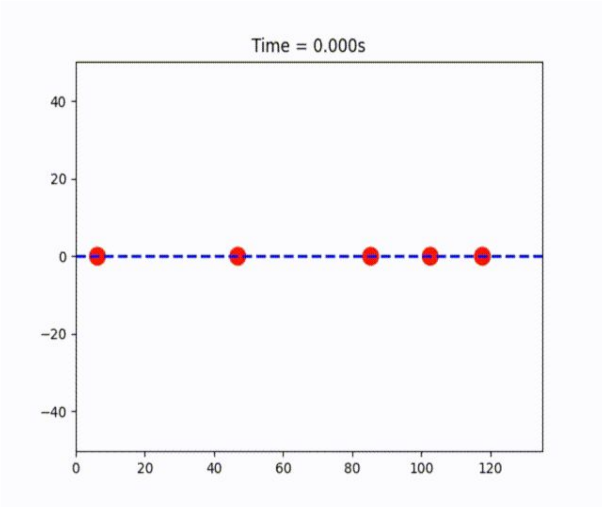
PDF (Función de densidad de probabilidad):

$$y_i = \frac{N_i}{(dx_i N)}$$

Resultados

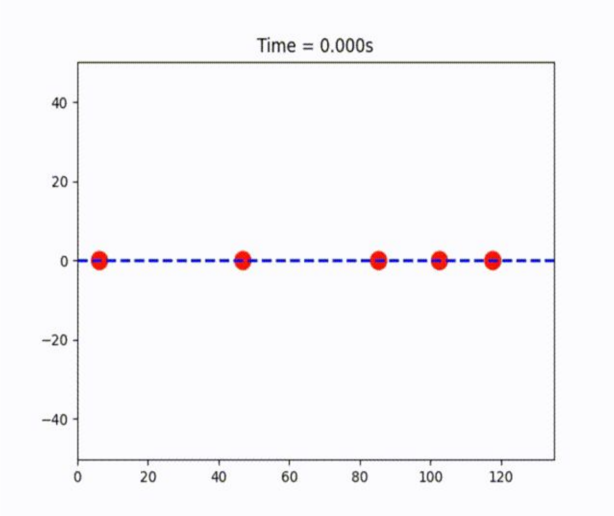
Animaciones con N=5

$\Delta t = 10^{-1}$



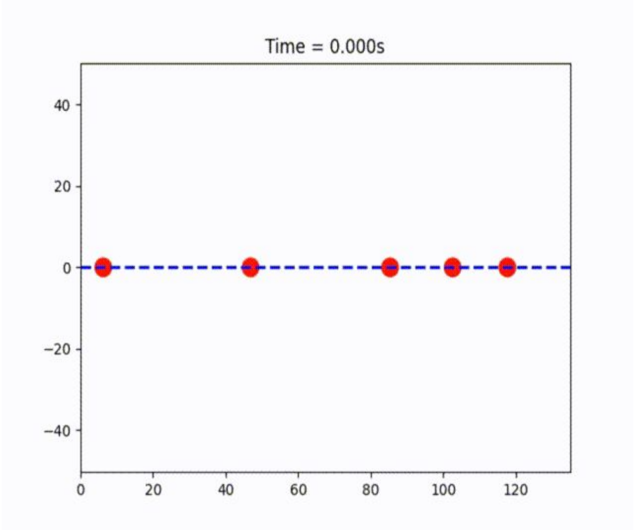
https://youtu.be/oX_NpBbFoOA

$\Delta t = 10^{-3}$



https://youtu.be/xuoPLUURBpI

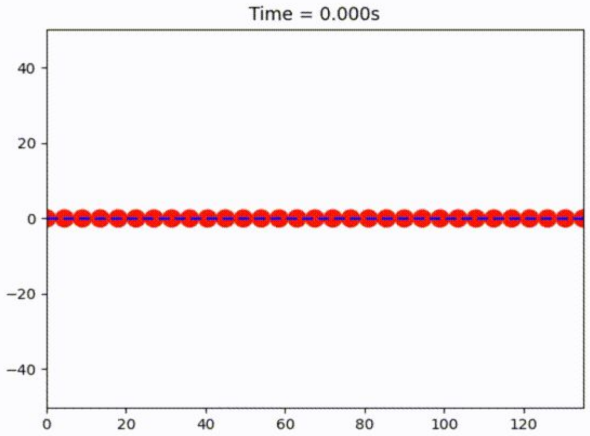
$\Delta t = 10^{-5}$



https://youtu.be/eM2JOsg7i48

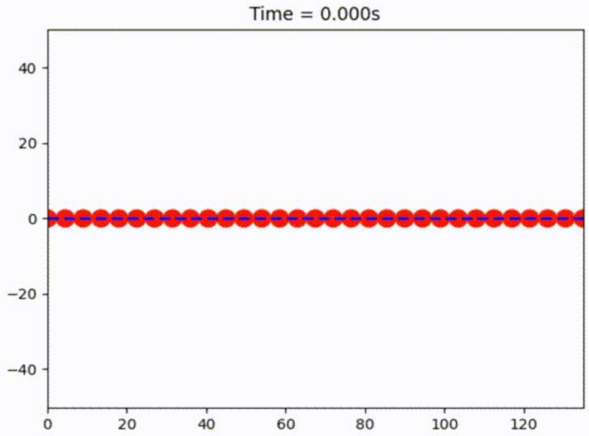
Animaciones con N=30

$\Delta t = 10^{-1}$



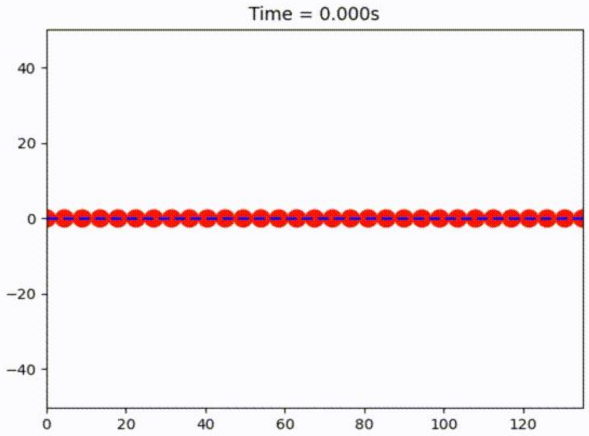
https://youtu.be/0VcTKbEslwE

$\Delta t = 10^{-3}$



https://youtu.be/1FWVdIRG_XA

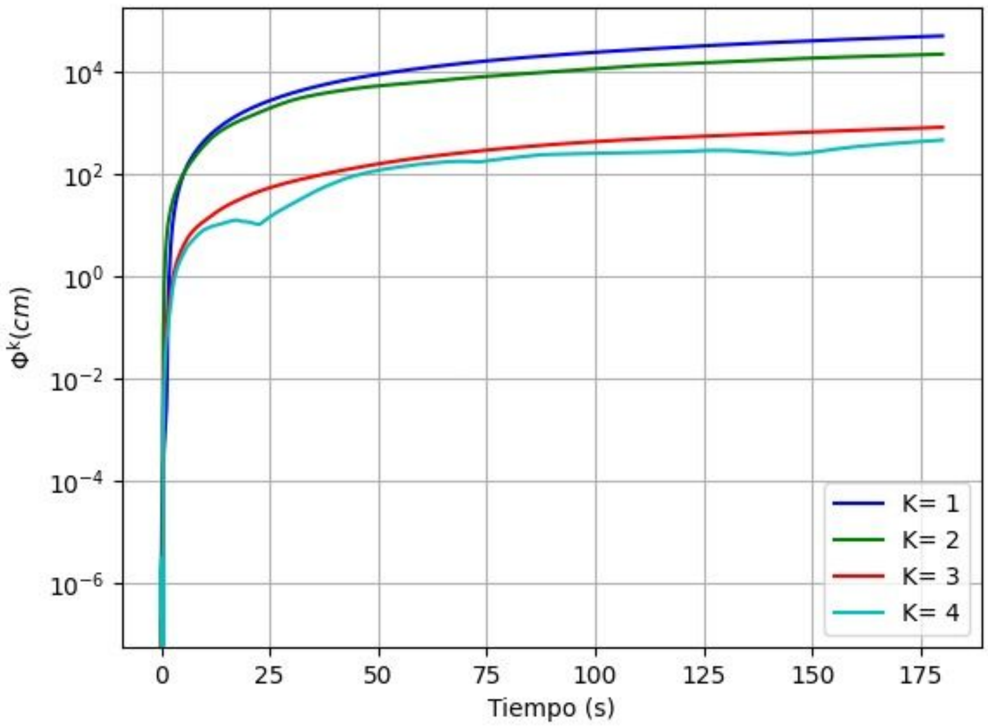
$\Delta t = 10^{-5}$



https://youtu.be/ijza5uwUT7Y

Gráfico de ϕ vs tiempo

- N=25
- Δt = 10^{-3} es el más adecuado.



¿Sería posible determinar Δt a partir de la conservación de energía del sistema?

No es posible determinar el **Δt** considerando la energía del sistema ya que no es conservativo. Existe una fuerza que impulsa a las partículas en toda la simulación.

Gráfico de la velocidad promedio vs tiempo

- Estado estacionario: 120s

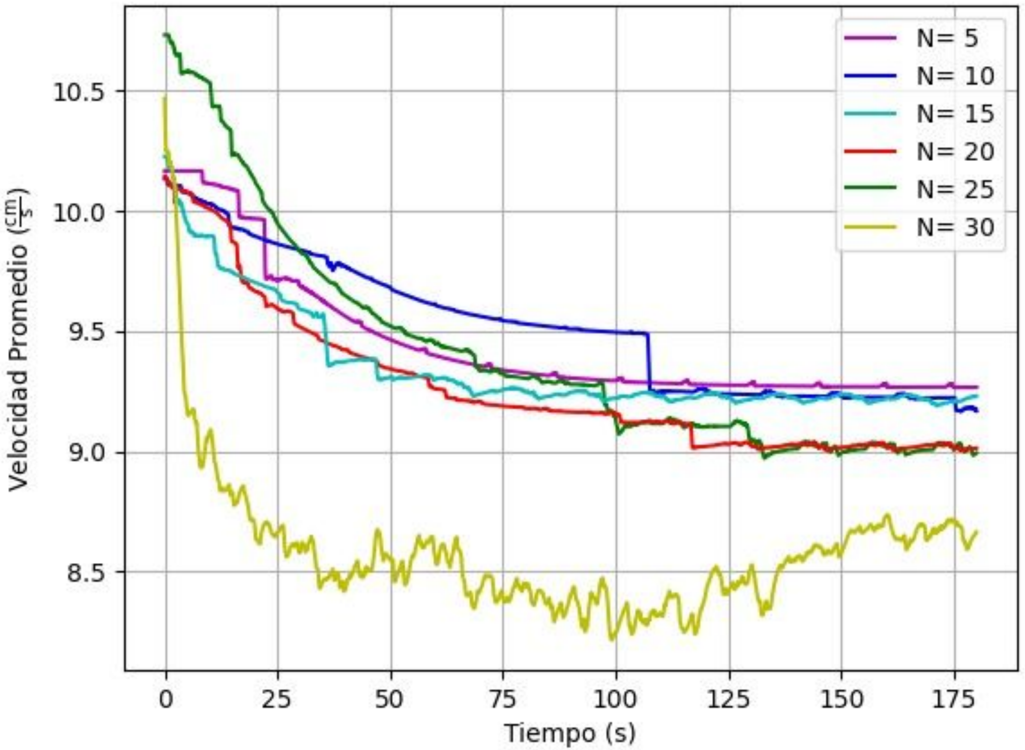


Gráfico de velocidad promedio temporal vs N

- 5 realizaciones
- Promedio entre 120s (estacionario) a T_f.

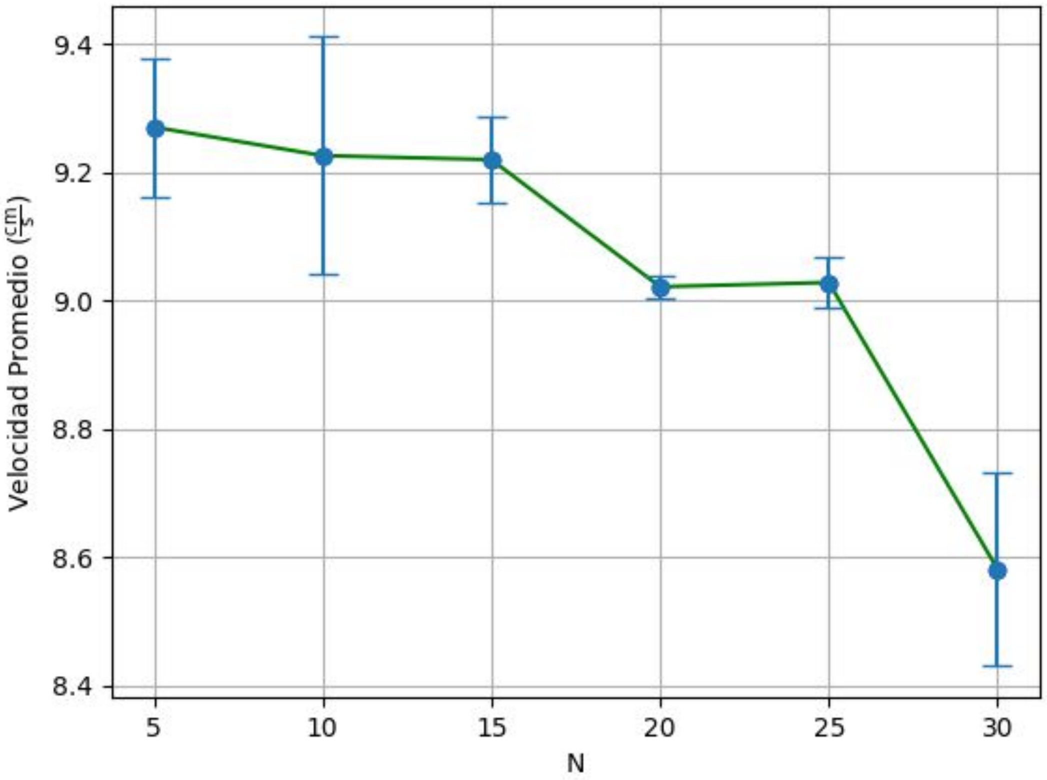


Gráfico de distribución de probabilidades de velocidades en el estado estacionario

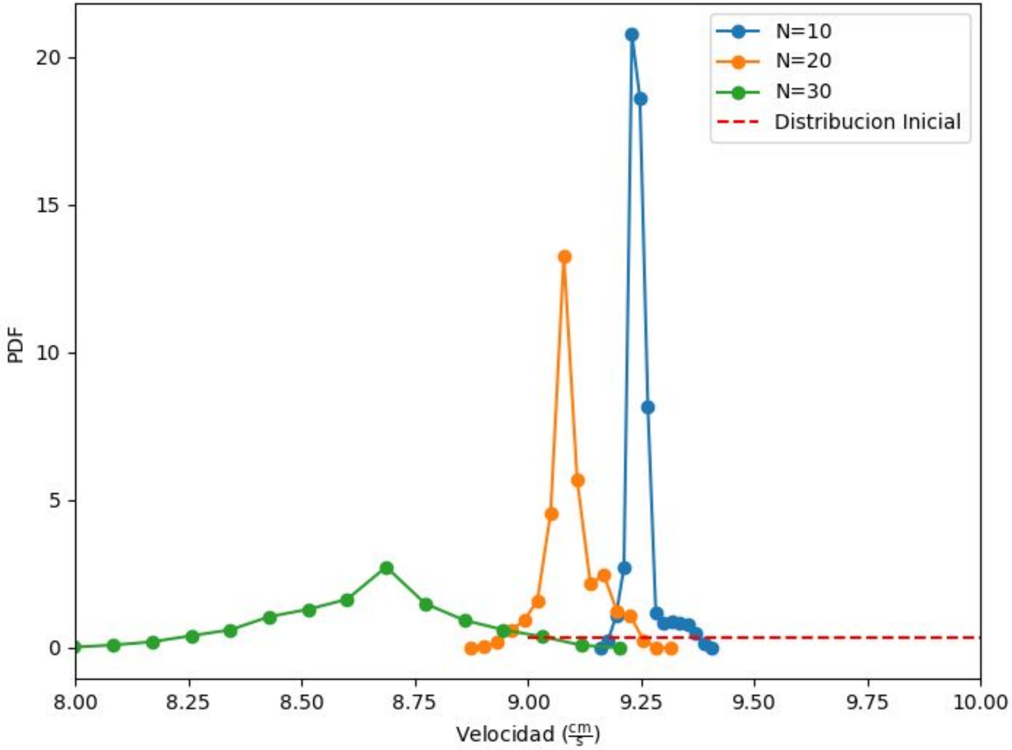


Gráfico de velocidad vs densidad

- N = 5, 10, 15, 20, 25, 30

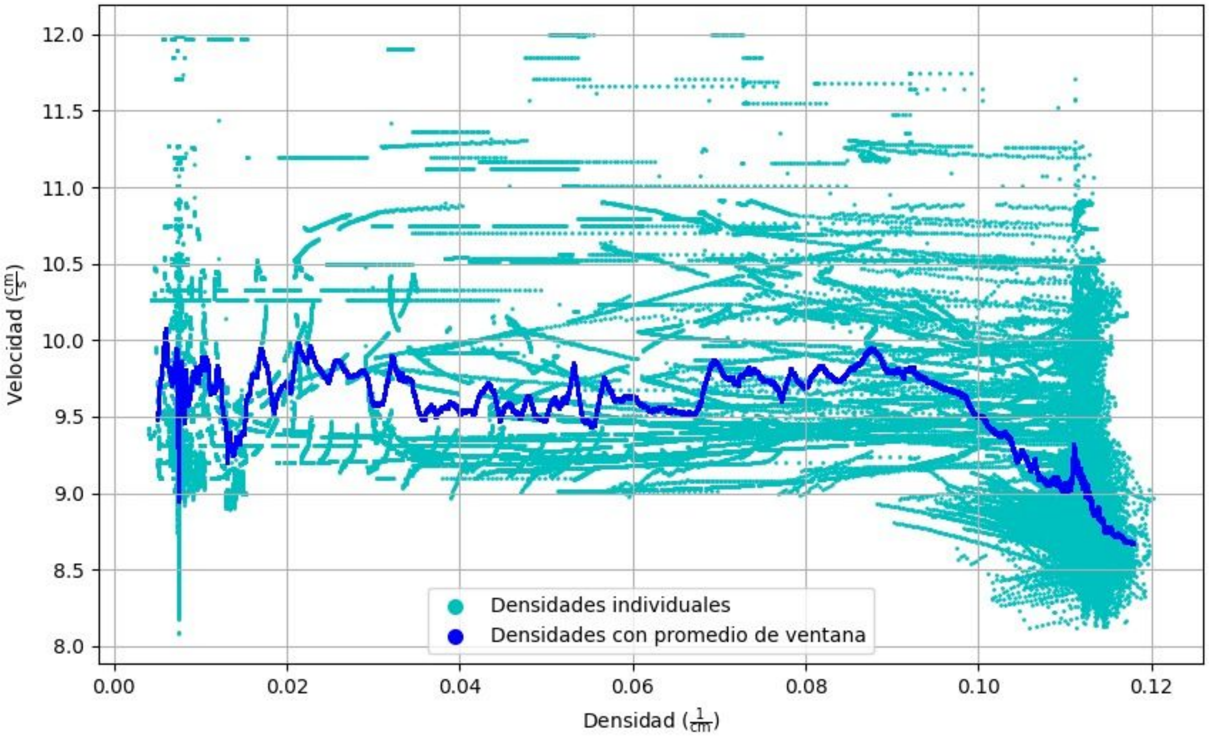


Gráfico de la velocidad promedio vs tiempo

<u>Partículas ordenadas inicialmente con u_i creciente</u>

- Estado estacionario: 120s

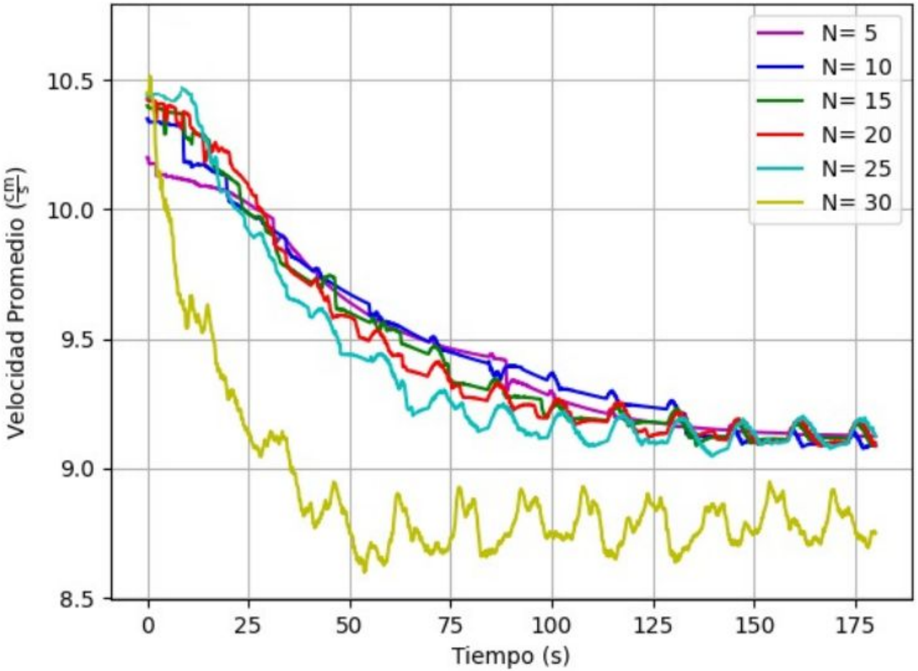


Gráfico de velocidad promedio temporal vs N

Partículas ordenadas inicialmente con u_i creciente

- 5 realizaciones

- Promedio entre 120s (estacionario) a T_f.

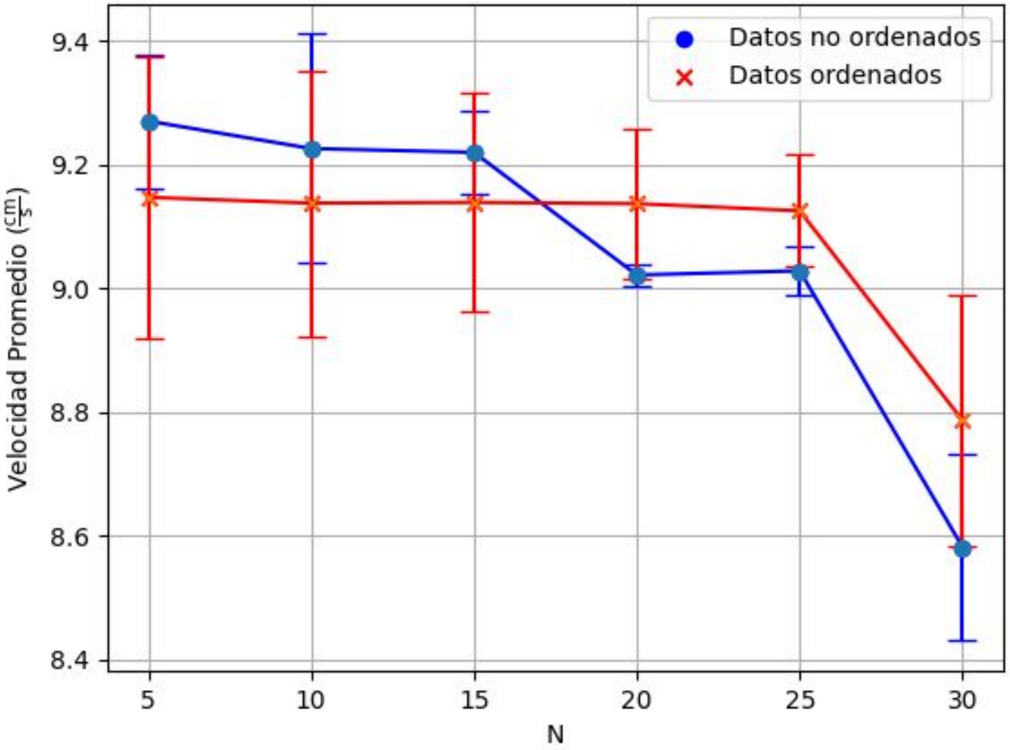
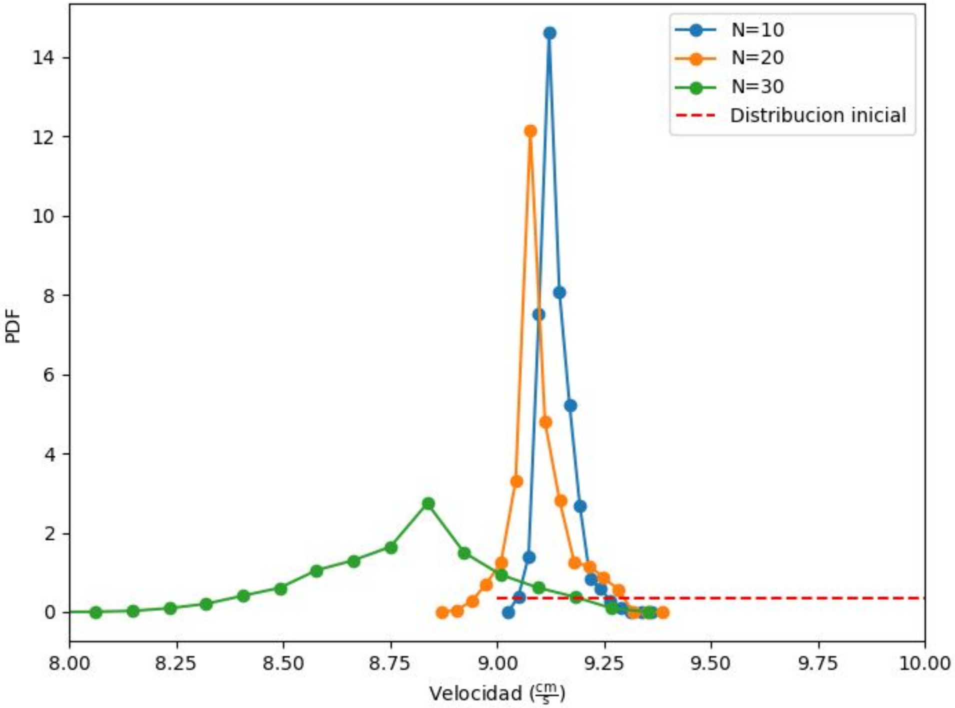


Gráfico de distribución de probabilidades de velocidades en el estado estacionario

Partículas ordenadas inicialmente con u_i creciente



Conclusiones

Conclusiones

- A menor Δt, menor Φ^k.
- A mayor N, menor velocidad.
- A mayor densidad menor velocidad ya que las partículas chocan más entre sí.
- Al ordenar las partículas por *u* creciente, tienden a la misma velocidad.
- Para N=30 no se presentan cambios en la velocidad cuanto a u ordenado o desordenado.

Fin

¡Muchas gracias!