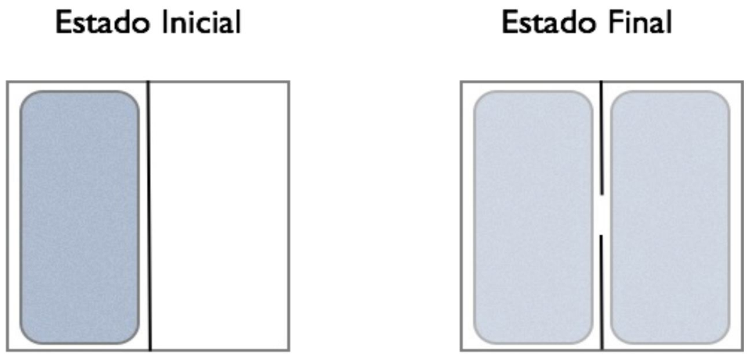
Difusión de un gas 2D

Agustin Jerusalinsky, Legajo 60406
Agustin Tormakh, Legajo 60041
Agustin Spitzner, Legajo 60142

Introduccion

Sistema

- N particulas rigidas
- Inicialmente en el recinto izquierdo
- Colisiones elásticas entre otras partículas y paredes
- No hay gravedad



Fundamentos

Fundamentos

Fórmula vuelo libre de partículas:

$$x_i\,(t_c) = x_i(0) + v_{x_i} t_c$$
$$y_i\,(t_c) = y_i(0) + v_{y_i} t_c$$

Posiciones iniciales partículas:

$$(x_i - x_j)^2 + (y_i - y_j)^2 > (R_i + R_j)^2$$

Tiempo de choque:

$$(x_{p2} - R) = x(0) + vx\; t \;\Rightarrow\; t_c = (x_{p2} - R - x(0)) \,/\, vx$$
$$(x_{p1} + R) = x(0) + vx\; t \;\Rightarrow\; t_c = (x_{p1} + R - x(0)) \,/\, vx$$

Colisión de partículas (con conservación del Impulso)

$$J_x = \frac{J\,\Delta\,x}{\sigma}, \quad J_y = \frac{J\,\Delta\,y}{\sigma}, \quad \text{donde} \quad J = \frac{2\,m_i\,m_j\,(\Delta v \cdot \Delta r)}{\sigma\,(m_i + m_j)}$$

$$vx_i^d = vx_i^a + J_x/m_i \qquad vx_j^d = vx_j^a - J_x/m_j$$
$$vy_i^d = vy_i^a + J_y/m_i \qquad vy_j^d = vy_j^a - J_y/m_j$$

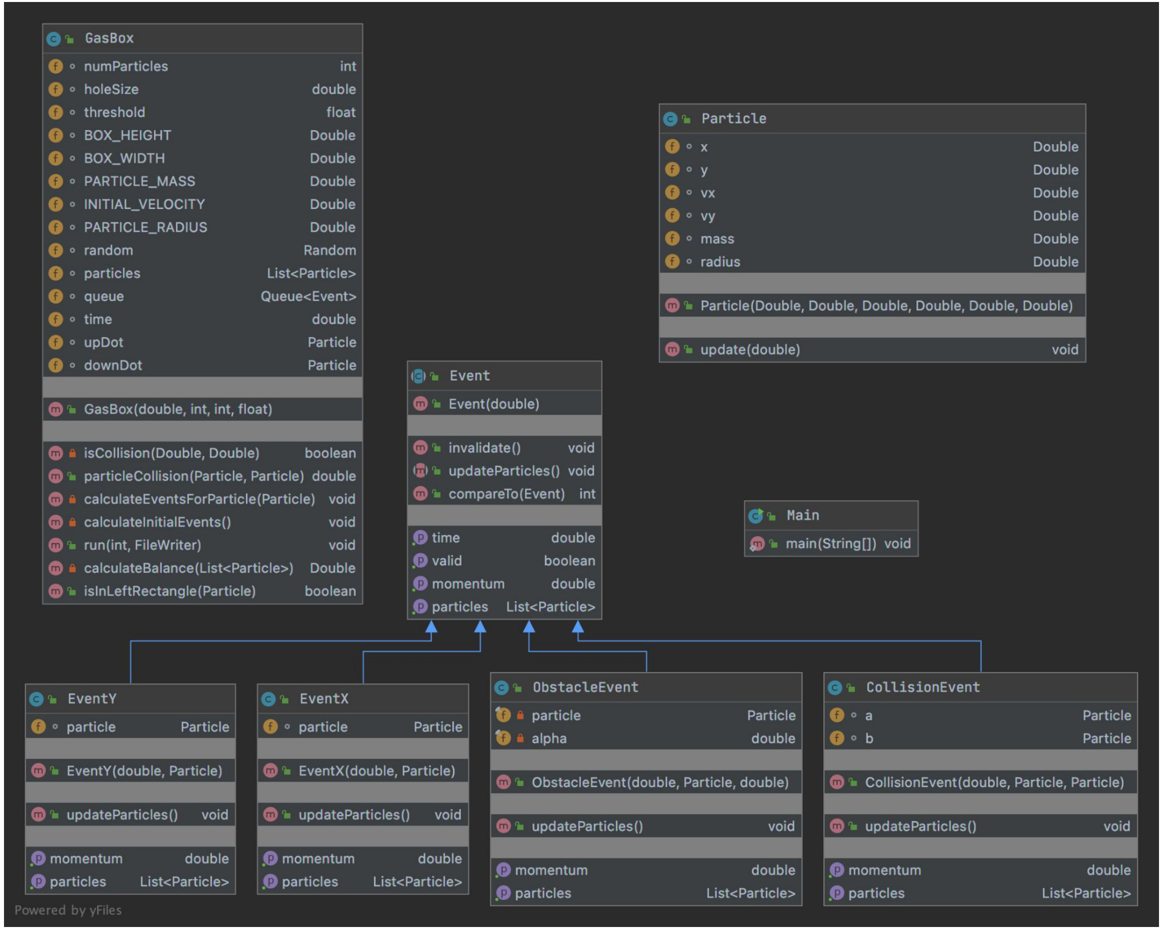
Colisión entre una partícula y un obstáculo fijo

$$\mathbf{v}^f = \begin{pmatrix} -\,c_n \cos^2(\alpha) + c_t\, sen^2(\alpha) & -\,(c_n + c_t)\, sen(\alpha)\cos(\alpha) \\ -\,(c_n + c_t)\, sen(\alpha)\cos(\alpha) & -\,c_n\, sen^2(\alpha) + c_t \cos^2(\alpha) \end{pmatrix} \mathbf{v}^a$$

Implementación

- UML
- Algoritmo y Pseudocodigo

UML


GasBox
numParticles int
holeSize double
threshold float
BOX_HEIGHT Double
BOX_WIDTH Double
PARTICLE_MASS Double
INITIAL_VELOCITY Double
PARTICLE_RADIUS Double
random Random
particles List<Particle>
queue Queue<Event>
time double
upDot Particle
downDot Particle
GasBox(double, int, int, float)
isCollision(Double, Double) boolean
particleCollision(Particle, Particle) double
calculateEventsForParticle(Particle) void
calculateInitialEvents() void
run(int, FileWriter) void
calculateBalance(List<Particle>) Double
isInLeftRectangle(Particle) boolean
Particle
x Double
y Double
vx Double
vy Double
mass Double
radius Double
Particle(Double, Double, Double, Double, Double, Double)
update(double) void
Event
Event(double)
invalidate() void
updateParticles() void
compareTo(Event) int
time double
valid boolean
momentum double
particles List<Particle>
Main
main(String[]) void
EventY
particle Particle
EventY(double, Particle)
updateParticles() void
momentum double
particles List<Particle>
EventX
particle Particle
EventX(double, Particle)
updateParticles() void
momentum double
particles List<Particle>
ObstacleEvent
particle Particle
alpha double
ObstacleEvent(double, Particle, double)
updateParticles() void
momentum double
particles List<Particle>
CollisionEvent
a Particle
b Particle
CollisionEvent(double, Particle, Particle)
updateParticles() void
momentum double
particles List<Particle>
Powered by yFiles

Algoritmo y pseudocodigo

```
public void correrSimulacion(){
    Particulas particulas = crearDistribucionInicialParticulas()

    List<Eventos> eventos = calcularEventosIniciales(particulas)

    while( (balance= calcularBalanceParticulas(particles)) >= 50+UMBRAL){

        Evento evento = eventos.pop()

        actualizarPosicionParticulas(particulas, evento.tiempoColision)

        resolverColision(particulas, evento)

        recalcularEventos(particulas, evento, eventos)

    }

}
```

Simulación

- Rango de variables
- Colisiones
- Observables (fp y P)
- Promedios Iteraciones

Variables

Constantes

```
ancho_caja=0.24m

alto_caja=0.09m

masa partículas=1kg

radio partículas=0.0015m

umbral=5%
```

Dinamicas

N = cantidad de partículas

```
N=[50,75,100,125,150]
```

D = ancho apertura

```
D=[0.02,0.03,0.04,0.05]
```

Vi = velocidad Inicial

```
Vi=[0.01,0.03,0.04]
```

Colisiones con otras partículas

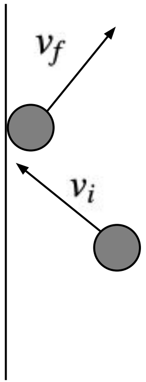
$$J_x = \frac{J\,\Delta x}{\sigma}, \quad J_y = \frac{J\,\Delta y}{\sigma}, \quad \text{where} \quad J = \frac{2\,m_i\,m_j\,(\Delta v \cdot \Delta r)}{\sigma\,(m_i + m_j)}$$

$$vx_i' = vx_i + Jx \,/\, m_i, \quad vx_j' = vx_j - Jx \,/\, m_j$$

$$vy_i' = vy_i + Jy \,/\, m_i, \quad vy_j' = vy_j - Jy \,/\, m_j$$

Colisiones con paredes

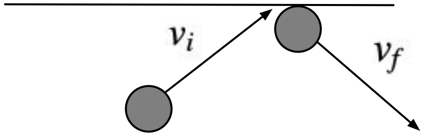
Horizontal



Después de colisión

$$vx_f = -vx_i$$
$$vy_f = vy_i$$

Vertical



Después de colisión

$$vx_f = vx_i$$
$$vy_f = -vy_i$$

Colisiones con vértice del tabique

$$\mathbf{v}^f = \begin{pmatrix} -c_n \cos^2(\alpha) + c_t \operatorname{sen}^2(\alpha) & -(c_n + c_t) \operatorname{sen}(\alpha) \cos(\alpha) \\ -(c_n + c_t) \operatorname{sen}(\alpha) \cos(\alpha) & -c_n \operatorname{sen}^2(\alpha) + c_t \cos^2(\alpha) \end{pmatrix} \mathbf{v}^a$$

Donde Cn = Ct = 1 por ser choque elástico

$$\alpha = atan2(Ry - Borde_y, Rx - Borde_x)$$

```
particle.vx = (sin * sin - cos * cos) * vx - 2 * sin * cos * vy;
particle.vy = -2 * sin * cos * vx + (cos * cos - sin * sin) * vy;
```

Observables

$g(p) = 1$ si p esta del lado izquierdo, 0 si p esta del lado derecho

$$fp = \frac{\sum_p g(p)}{N}$$

Siendo N la cantidad de partículas, en otras palabras es la suma de las partículas de lado izquierdo dividido la cantidad de partículas totales

$$P = \frac{\sum_i 2*mi*|v_n|}{\Delta t*L}$$

P presión, “i” identifica a una partícula que colisionó con una pared dentro del tiempo Δt , ”L” es el perímetro del sistema y “Vn” es la velocidad normal a la superficie en el momento del impacto. Por lo tanto, la Presión es igual a la suma de los momentos transferidos a la pared dividido el lapso de tiempo y dividido el perímetro del sistema

$t_{eq}(umbral) = t$ tal que $fp - 0.5 < umbral$ en el instante t

Tiempo de equilibrio es el tiempo que tarda el sistema en tener una diferencia de la proporción de partículas en cada recinto menor a un umbral

Promedios

```
Valores = [v1,  v2,  v3,  … , vn]

Promedio = mean(valores)

Error = stdev(valores)/sqrt(len(valores))
```

Resultados

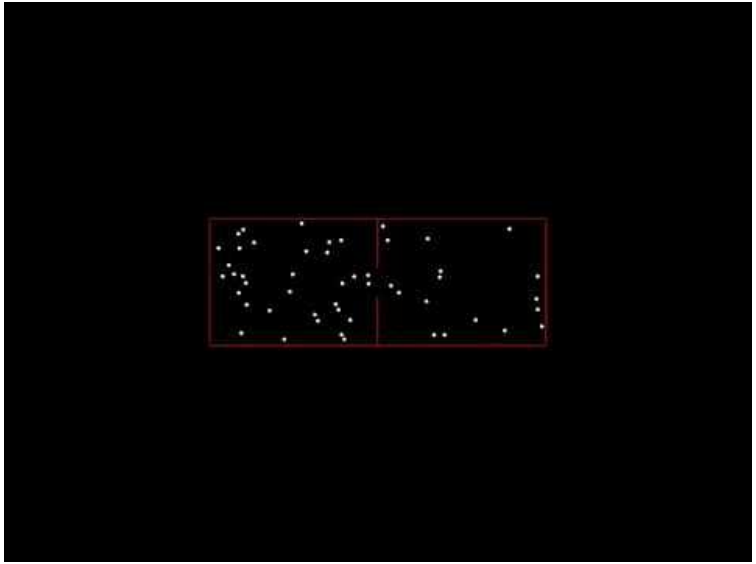
- N = cantidad partículas
- D = Ancho tabique
- Ley gases ideales

N = Número de partículas

Condiciones:

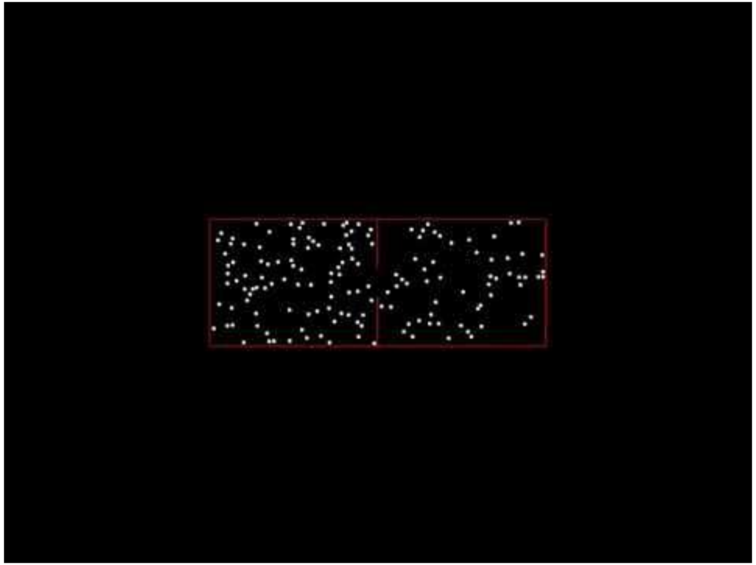
- *vi= 0.01m/s*
- *radio=0.015m*
- *masa=1kg*
- *D=0.03m*
- *umbral= 5%*

N=50



https://youtu.be/EYAeoUGMenc

N=150

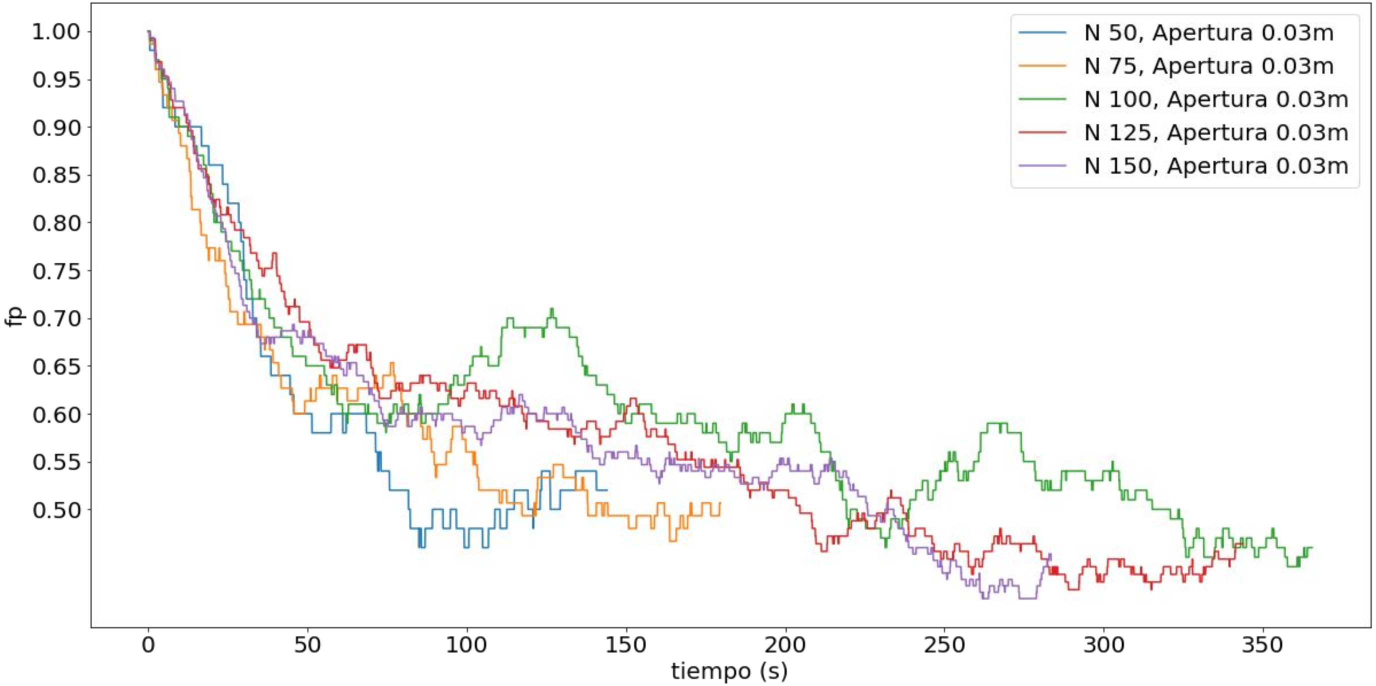


https://youtu.be/_Vrtkp3vPoo

N = Número de partículas

Condiciones:

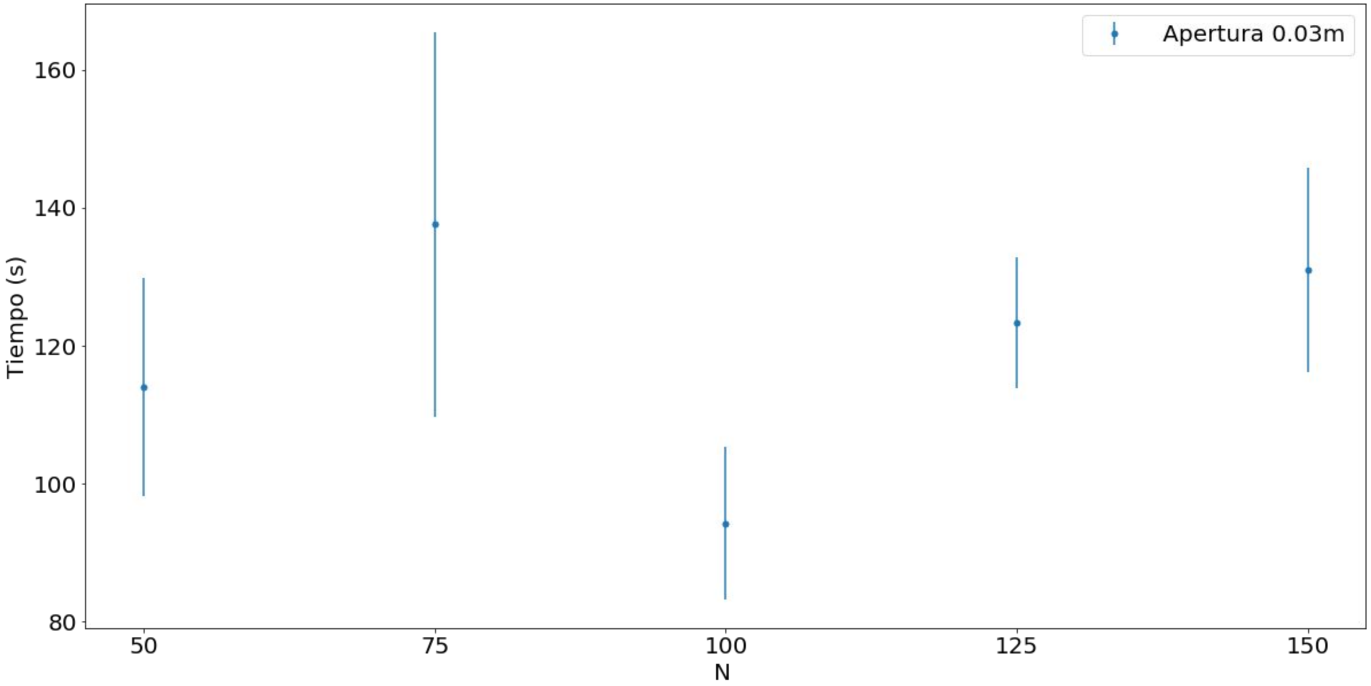
- *vi= 0.01m/s*
- *radio=0.015m*
- *masa=1kg*
- *D=0.03m*
- *umbral= 5%*



N = Número de partículas

Condiciones:

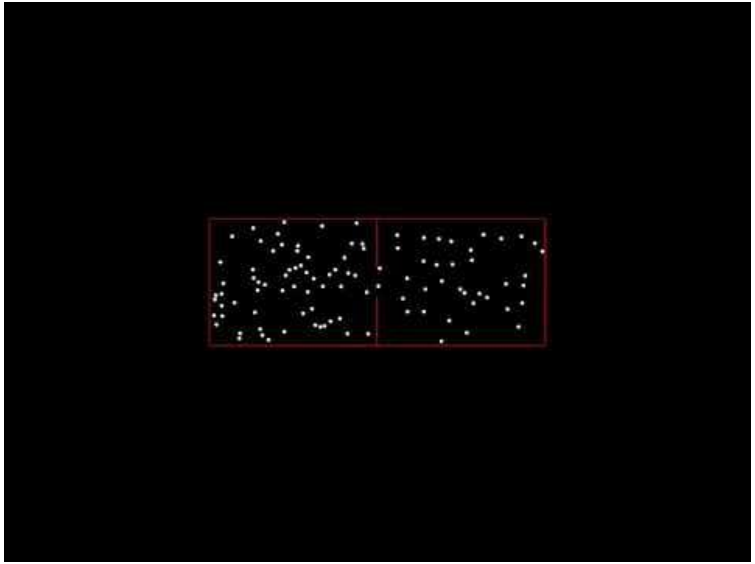
- *vi= 0.01m/s*
- *radio=0.015m*
- *masa=1kg*
- *D=0.03m*
- *umbral= 5%*



D = ancho de la apertura

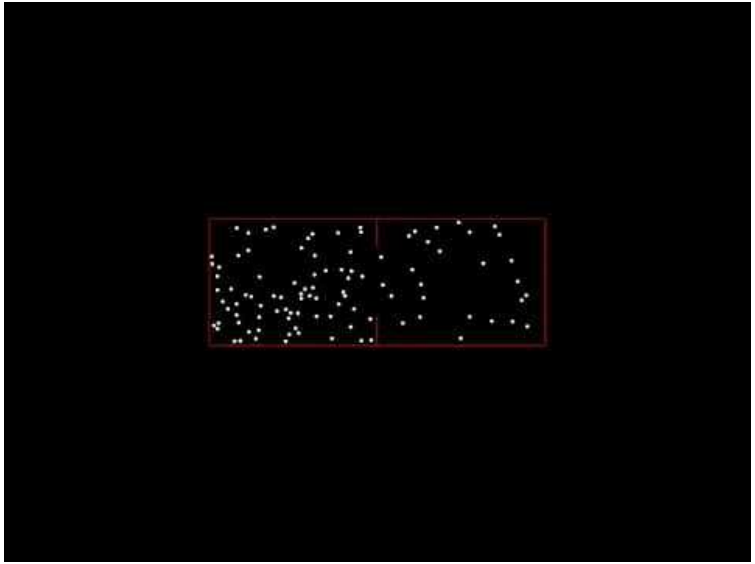
Condiciones:
- *vi= 0.01m/s*
- *radio=0.015m*
- *masa=1kg*
- *N=100*
- *umbral= 5%*

D=0.02m



https://youtu.be/fNAgGzaoUmw

D=0.05m

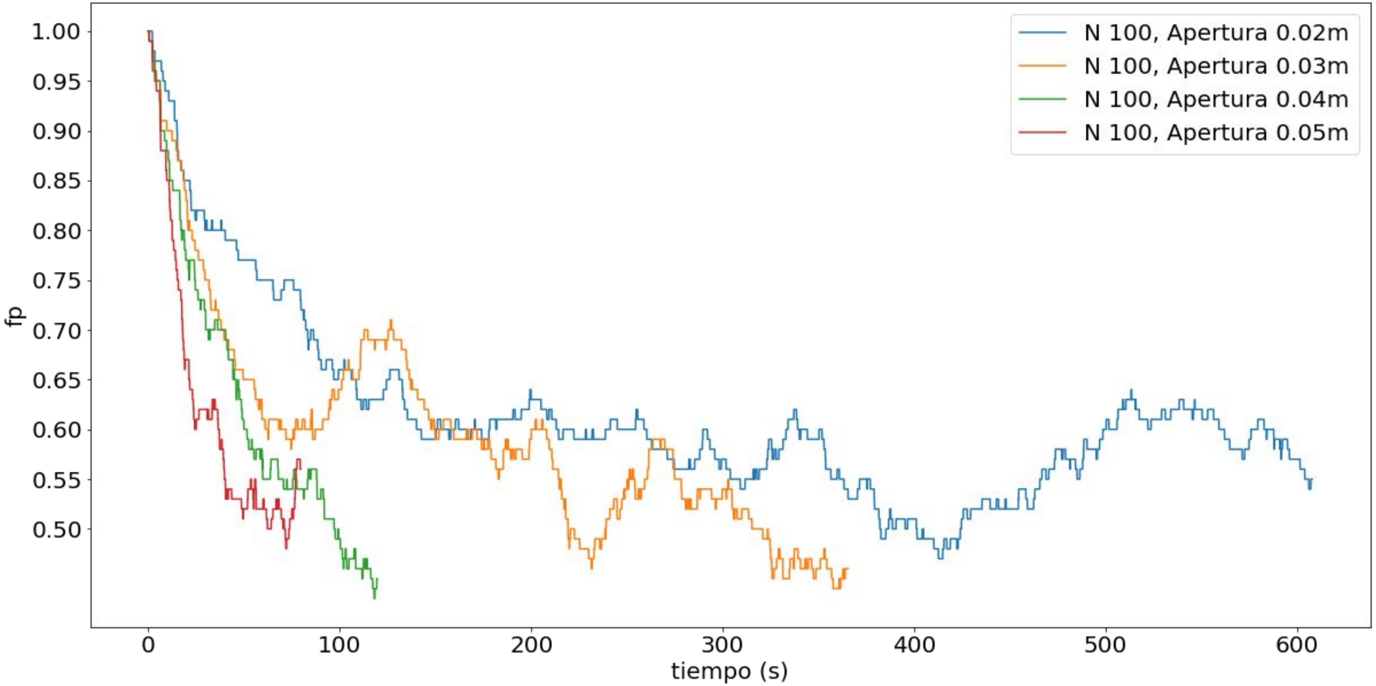


https://youtu.be/SsIWOW4YSCk

D = ancho de la apertura

Condiciones:

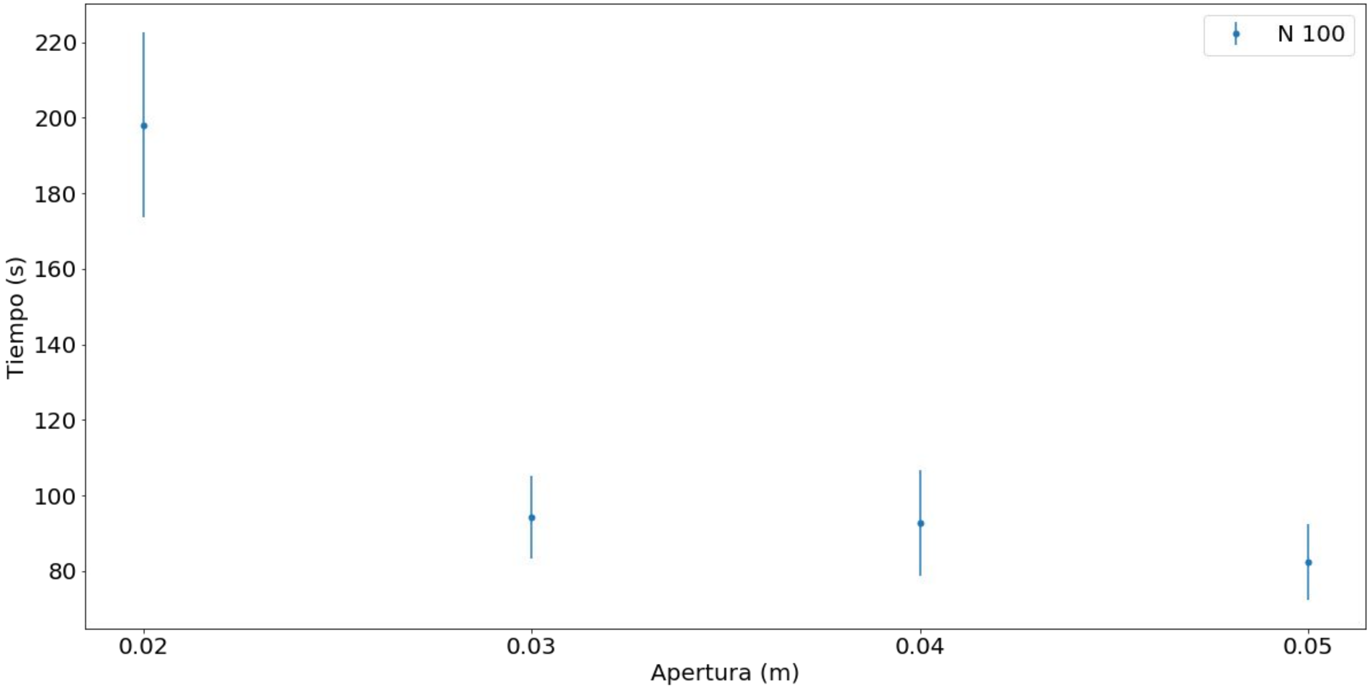
- *vi= 0.01m/s*
- *radio=0.015m*
- *masa=1kg*
- *N=100*
- *umbral= 5%*



D = ancho de la apertura

Condiciones:

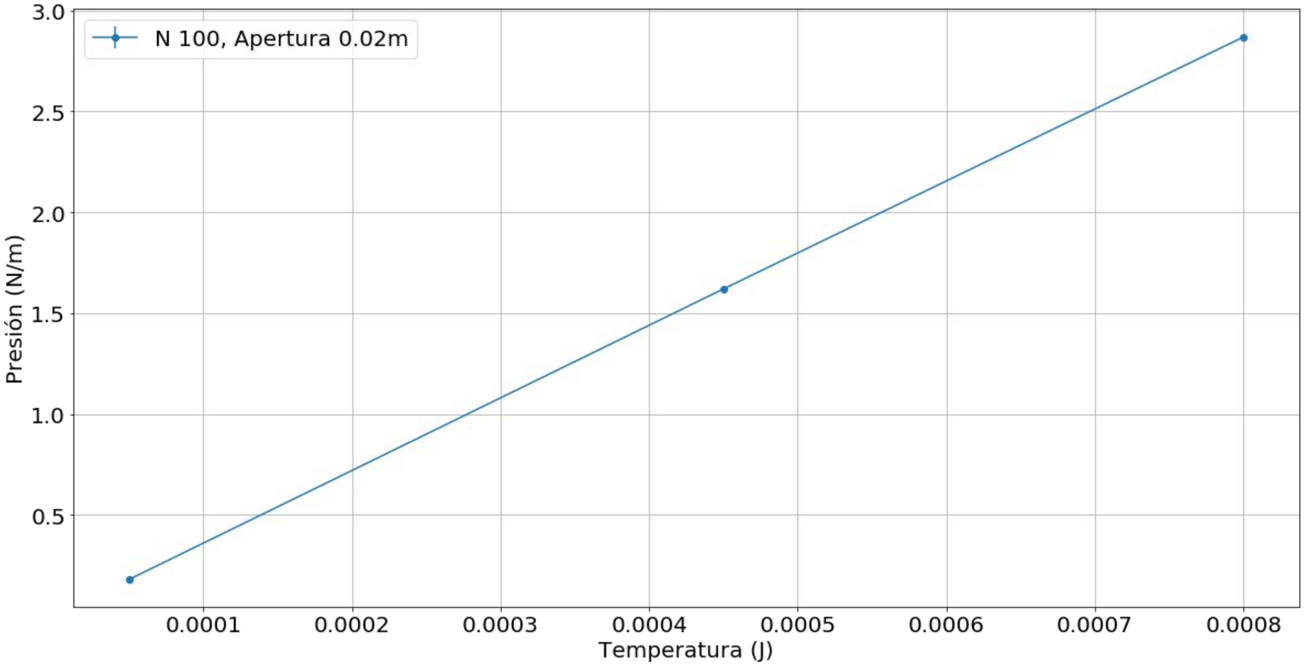
- *vi= 0.01m/s*
- *radio=0.015m*
- *masa=1kg*
- *N=100*
- *umbral= 5%*



Ley gases ideales

Condiciones:

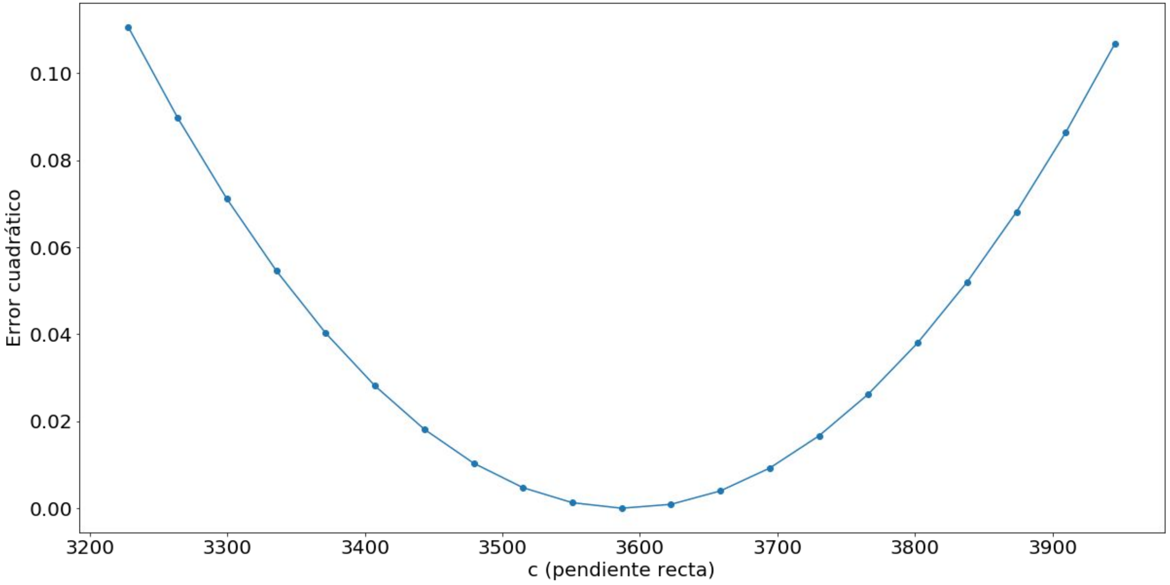
- *vi= 0.01m/s*
- *radio=0.015m*
- *masa=1kg*
- *N=100*
- *D=0.02m*
- *umbral= 5%*



Ley gases ideales

Condiciones:

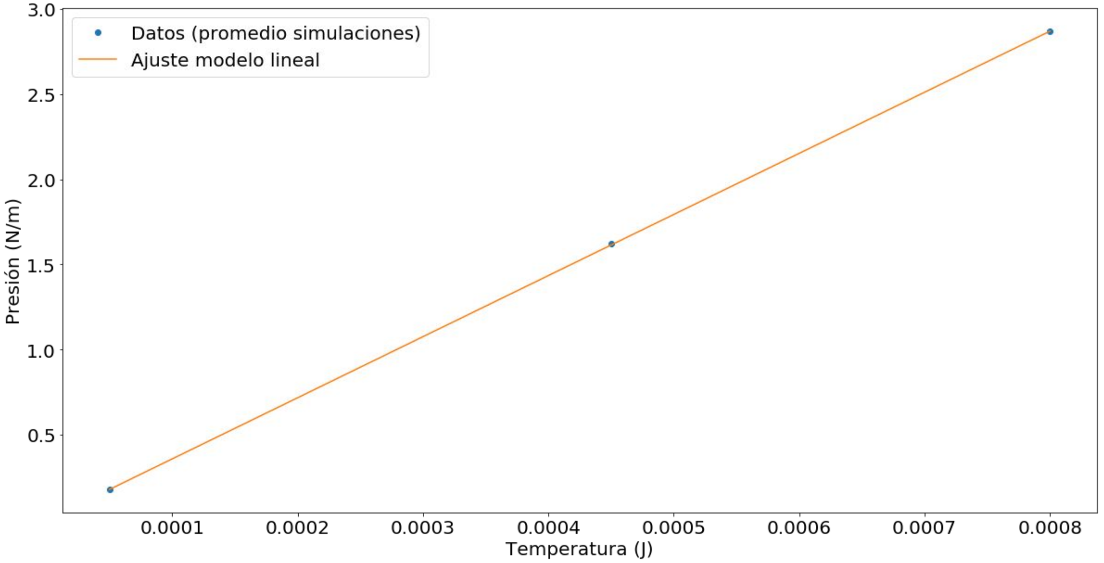
- *vi= 0.01m/s*
- *radio=0.015m*
- *masa=1kg*
- *N=100*
- *D=0.02m*
- *umbral= 5%*



Ley gases ideales
(P = c*T)

Condiciones:

- *vi= 0.01m/s*
- *radio=0.015m*
- *masa=1kg*
- *N=100*
- *D=0.02m*
- *umbral= 5%*



c =3586.7617   error=0.00003

Conclusiones

Conclusiones

- El tiempo de llegada al equilibrio disminuye al aumentar D
- Aumentar N no siempre produce que disminuya el tiempo de llegada al equilibrio
- En el equilibrio se cumple la ley de gases ideales

Gracias