"P9" Interacciones entre partículas

NESTOR

Abril 2022

1. Objetivo

El objetivo de la práctica consiste en un modelo simplificado para los fenómenos de atracción y repulsión de física .Supongamos que contemos con n partículas que habitan un cuadro unitario bidimensional y que cada partícula tiene una carga eléctrica, distribuida independientemente e normalmente al azar entre $[-1, 1]$. Cargas de un mismo signo producirán una repulsión mientras cargas opuestas resultan en una atracción la magnitud de la fuerza estará proporcional a la diferencia de magnitud de las cargas (mayores diferencias resultando en fuerzas mayores), y además la fuerza será inversamente proporcional a la distancia euclideana entre las partículas [1].

2. Desarrollo

Basandome en el desarrollo en la codificación implementado por E. Schaeffer y todas las instrucciones se encuentran en el repositorio de N. Rodríguez en GitHub.

Para comenzar se hace primero generar la función para generar partículas de atracción y repulsión con esto para poder visualizar la masa

Código 1: Generamos las partículas

```
import numpy as np
import pandas as pd
from random import uniform
import matplotlib.pyplot as plt
paso = 256 // 10
niveles = [i/256 for i in range(0, 256, paso)]
eps = 0.001
```

Generamos las posiciones de las partículas en "x y y", se les asigna una carga y una masa con números random uniformemente con la finalidad de asignar masas variables.

Código 2: Partículas en "x y y"

```
if __name__ == "__main__":
    inicial=[]
    popen('rm -f p9p_t*.png') # borramos anteriores en el caso que lo hayamos corrido
    n = 25
    x = np.random.normal(size = n)
    y = np.random.normal(size = n)
    masa = [uniform(0,50)for i in range(n)]
    c = np.random.normal(size = n)
    #masa=[s * (-1) for s in masa]
```

Se realizó ya las masas variables, ahora se codifica que la masa cause un efecto ya que está asignada a una partícula, ya que con esto se hace varios pasos para que vayan avanzando las partículas, a continuación se muestra la codificación:

Código 3: masa

```
def fuerza(i, shared):
    p = shared.data
    n = shared.count
    pi = p.iloc[i]
    xi = pi.x
    yi = pi.y
    ci = pi.c
```

```
    mi = pi.masa
    fx, fy = 0, 0
    for k in range(n):
        pk = p.iloc[k]
        ck = pk.c
        mk = pk.masa
        dire_c = (-1)**(1 + (ci * ck < 0))
        dire_m = (-1)**(1 + (mi * mk > 0))
        factor_c = dire_c * fabs(ci - ck) / (sqrt(dx**2 + dy**2) + eps)
        factor_m = dire_m * fabs(mi - mk) / (sqrt(dx**2 + dy**2) + eps)
        fx -= dx * factor_m * factor_c
        fy -= dy * factor_m * factor_c
    return (fx, fy)
```

Generamos la correlación de rango de Spearman, esto nos ayudará a correlacionar las tres variables en una medida de asociación lineal que se utilizará para los rangos.

Código 4: Spearman

```
### Correlacion de rango de Spearman
    from scipy.stats import spearmanr
    stat, p = spearmanr(x, z)
    print("correlacion carga con velocidad")
    print('stat=%.3f, p=%.3f' % (stat, p))
    if p > 0.05:
        print('Probablemente dependiente')
    else:
        print('Probablemente independiente')
    print("correlacion masa con velocidad")
    stat2, p2 = spearmanr(y, z)
    print('stat=%.3f, p=%.3f' % (stat2, p2))
    if p2 > 0.05:
        print('Probablemente dependiente')
    else:
        print('Probablemente independiente')
```

3. Resultados

Cuadro 1: Correlacion de rango de Spearman para las partículas

Mediciónes	Estadística
Relacion entre la velocidad con carga y velocidad con masa	0,37567824 porciento
Correlacion carga con velocidad	stat=0,380, p=0,061 (Probablemente dependiente)
Correlacion masa con velocidad	stat=−0,285, p=0,167 (Probablemente dependiente)

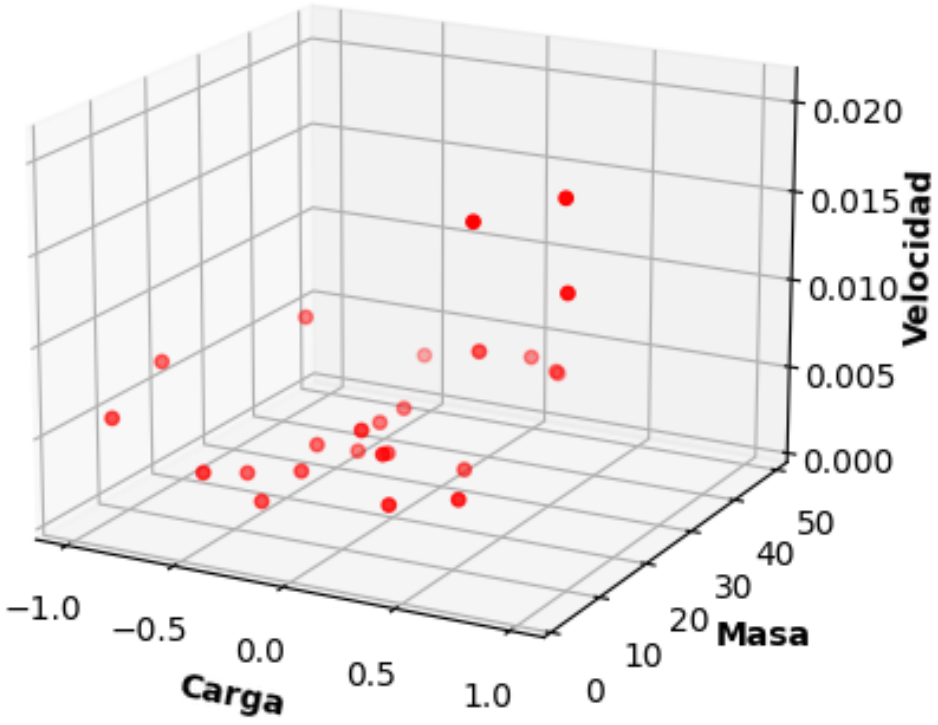

Figura 1: Partículas en la carga.

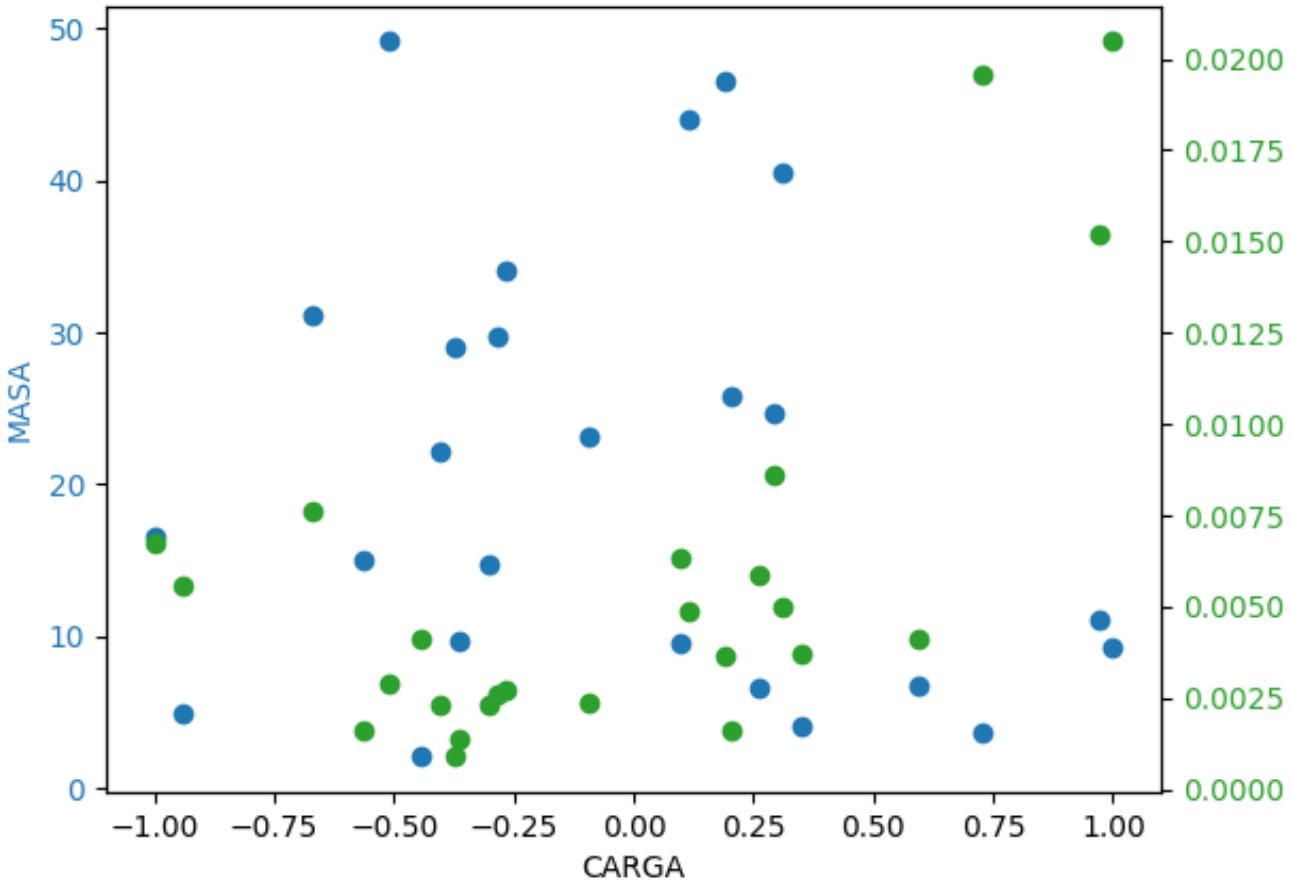

Figura 2: Comportamiento de carga.

4. Conclusiones

Se concluye que las partículas se correlacionan con las tres variables con respecto a la carga y la menor velocidad ya que se encuentra la diferencia en velocidades.

Referencias

[1] E. Schaeffer. Búsqueda local. *Repositorio, GitHub*, 2022. URL https://github.com/satuelisa/Simulation/blob/master/Particles/creation.py.