MALMÖ UNIVERSITY

Computational Physics: Introductory Course
Spring 2023

Assignment 9:
Monte-Carlo methods

Name: Nikolaos Giannakis
Personummer: 950417921
Hand-in date: 2022-04-17

Exercise 1

Compute the definite integral

$$\int_0^{10} xe^{-x} dx$$

using a Monte-Carlo method with $10^2, 10^3, \iddots, 10^7$ *random numbers. Compare with the analytical result. Present your results in a table*

```python
import numpy as np

# Define the function to integrate
def f(x):
    return x * np.exp(-x)

# Define the limits of integration
a, b = 0, 10

# Define the analytical solution
analytical_solution = 1 - np.exp(-b) * (b + 1)

# Define the number of random samples to use
num_samples = [10**2, 10**3, 10**4, 10**5, 10**6, 10**7]

# Define a list to store the results
results = []

# Loop over the number of samples
for n in num_samples:
    # Generate n random samples in the interval [a, b]
    x = np.random.uniform(a, b, n)

    # Evaluate the function at the random samples
    y = f(x)

    # Compute the integral approximation using the Monte Carlo method
    integral_approximation = (b - a) * np.mean(y)

    # Compute the error with respect to the analytical solution
    error = np.abs(integral_approximation - analytical_solution)

    # Append the results to the list
    results.append([n, integral_approximation, error])

# Print the results in a table
print("Number of samples\tIntegral approximation\tAnalytical Solution\tError")
for n, integral, error in results:
    print(f"{n}\t\t\t\t\t\t\t\t{integral:.6f}\t\t\t\t{analytical_solution:.6f}\t\t\t{error:.6f}")
```

```

Number of samples Integral   Analytical Solution   Error
100               1.046059        0.999501      0.046559
1000              0.945703        0.999501      0.053798
10000             0.998178        0.999501      0.001323
100000            1.002517        0.999501      0.003016
1000000           0.999850        0.999501      0.000350
10000000          0.998962        0.999501      0.000538
```

As it is clear the number of samples makes a difference in the error. We can see that higher number of samples gives a more precise approximation when using the Monte Carlo simulation.

Exercise 2

The density in g/cm3 in the interior of the sun can be approximated by the following expression:

$$(x) = (2139x + 155)e^{13.8x},$$

where x is a radial coordinate, stretching from the center ($x = 0$) to the surface of the sun ($x = 1$). Using a Monte-Carlo method (see section 22.6 in the lecture notes), compute the mass and moment of inertia of the sun and compare your result with other sources. For the radius of the sun, use $6.96 * 10^8$ m.

```python
import numpy as np

# Constants
R_sun = 6.96e8  # Radius of the sun in meters
N = 100000  # number of points in D0

# Define density function
def rho(x):
    return (2139*x + 155)*np.exp(-13.8*x)

# Initialize variables
fsum = 0

# Generate random points and compute sums
for i in range(N):
    # Generate random radial coordinate
    x = -R_sun + 2*R_sun*np.random.rand()
    y = -R_sun + 2*R_sun*np.random.rand()
    z = -R_sun + 2*R_sun*np.random.rand()
    r2 = x**2 + y**2 + z**2
    r = np.sqrt(r2/R_sun**2)
    # Compute density at current radial coordinate
    if r2 < R_sun**2:
        fsum = fsum + rho(r)*(x**2+ y**2)
```

```
        mass = fsum + rho(r)


# Compute moment of inertia contribution and update moment of inertia sum
I = 8 * R_sun ** 3/N*fsum



# Print results
print("Moment of inertia: %6.4e g*cm^3" %I)
print("Mass: ", mass)

# Moment of inertia: 6.0182e+43 g*cm^3
# Mass:  2.2312626819951719e+21
```

Exercise 3

In assignment 6 and exercise 2, we studied Rutherford scattering in two dimensions by solving a set of coupled first order differential equations. In particular, for a given impact parameter b you could find the corresponding numerical value of the scattering angle θ. In this exercise, you will extend the code and simulate thousands of scattered alpha particles and investigate how many alpha particles that are scattered at different angles. Finally, you will compare your results with predictions from the differential cross section for Rutherford scattering. To solve the exercise, you need to consult the Physics background section below. Assume the same values for the parameters as in assignment 6 and exercise 2. That is, $K = 3.47710^{13} fm^3 fs^{-2} e^{-2}, Q = 79e, q = 2e$ and $v = 1.5310^7 fm/fs(m/s)$. The scattering angles that will be investigated are in the interval $50 - 180°$. Find the impact parameter b_{max} corresponding to $50°$. Now follow the steeps below:

- For each alpha particle, construct coordinates b_y and b_z of the impact parameter b by assigning them uniformly distributed random numbers between 0 and $bmax$. The impact parameter is then given as $b = \sqrt{b_y^2 + b_z^2}$
- If $b < bmax$, use the code obtained in assignment 6, exercise 2 to determine the numerical scattering angle. Store the scattering angle in an array.
- Repeat steps (a) and (b) for at least 1 000 alpha particles and plot a histogram in the range $50° - 180°$ based on the array with the scattering angles. Choose a suitable bin number.
- Compare the resulting histogram with the predicted number of scattered alpha particles in each bin. See figure 1 for expected results.

```python
import numpy as np
from scipy.integrate import solve_ivp
import matplotlib.pyplot as plt

# Define the constants
K = 3.477e13  # fm^3 fs^-2 e^-2
q = 2  # elementary charge
Q = 79  # gold nucleus charge
v = 1.53e7  # fm/fs
theta = np.deg2rad(50)
m = 6.64e-27
b_max = (K * Q * q / v**2) * (1 / np.tan(theta/2))
# Generate random impact parameters for 1000 alpha particles
np.random.seed(42)
by = np.random.uniform(0, b_max, size=1000)
bz = np.random.uniform(0, b_max, size=1000)
b_values = np.sqrt(by**2 + bz**2)


```

```
print("B is equal to: ", b_values)
print("B_max is equal to: ",b_max)



def f(t, y):
    x, y, vx, vy = y
    r = np.sqrt(x**2 + y**2)
    fx = (K * q * Q * x) / r**3
    fy = (K * q * Q * y) / r**3
    return [vx, vy, fx, fy]

# Define the time range
t = np.linspace(0, 0.1, 1000)

# Define the initial conditions
x0 = -1e6
y0 = 25
vx0 = v
vy0 = 0

# Solve the differential equations
sol = solve_ivp(f, [0, 0.1] , [x0,y0,vx0,vy0] , t_eval= t)

# extract the solution
x, y, vx, vy = sol.y
theta_numerical = []


# Solve the differential equations and calculate the scattering angles
for i,b in enumerate(b_values):
    if(b<b_max):
        y0 = [x0, b, vx0, vy0]
        sol = solve_ivp(f, [0, 0.1], y0, t_eval=t, rtol=1e-10, atol=1e-10)
        x, y, vx, vy = sol.y
        dy = y[1000-1]-y[1000-11]
        dx = x[1000-1]-x[1000-11]
        thetanum = np.rad2deg(np.arctan(dy/dx))
        if dy < 0 and dx < 0:
            thetanum = 180 - thetanum
        elif dy > 0 and dx < 0:
            thetanum = 180 + thetanum
        elif dy < 0 and dx > 0:
            thetanum = - thetanum

        if thetanum >= 50 and thetanum <= 180:
            theta_numerical.append(thetanum)


# Specify the range and number of bins for the histogram
binedge = np.arange(50,181,10)
```

```python

# Plot the histogram and the analytical solution
plt.hist(theta_numerical, bins=binedge)

# Add labels and title
plt.xlabel('Scattering angle (degrees)')
plt.ylabel('Frequency')
plt.title('Histogram of Scattering Angles')

# Add legend
plt.legend()

# Show the plot
plt.show()

# I tried to add
# theta_analytical = np.linspace(50, 180, 100)
# def total_cross_section(theta):
    #return (4 * np.pi * K**2 * q**2 * Q**2) / (m * v**2 * np.sin(theta)/2)**4
# cross_sections_analytical = total_cross_section(theta_analytical)
# Plot the histogram and the analytical solution
# plt.hist(theta_numerical, bins=binedge, density=True, label='Numerical')
# plt.plot(theta_analytical, cross_sections_analytical, label='Analytical')
```

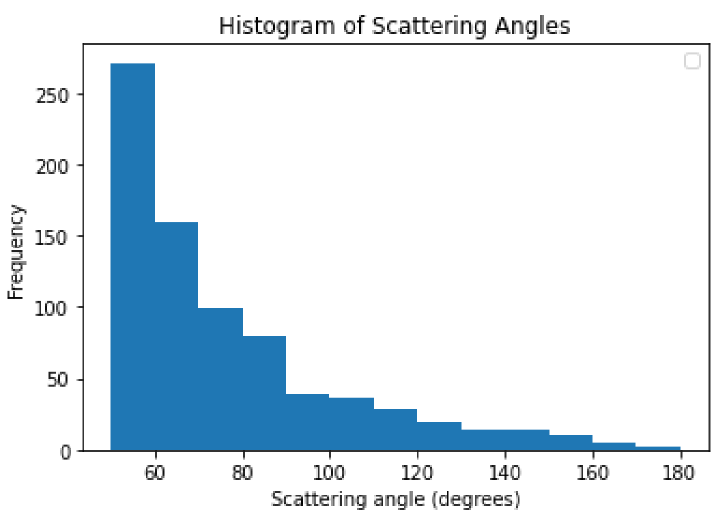

Figure 1: Histogram when simulating Rutherford with 1000 alpha particles

Now it is working properly I think for the histogram. However, I tried to add the analytical solution but I did not manage to make it work. I think I have a huge problem understanding this exercise to be honest. See the comments please on the code and let me know if there is anything to be fixed. Thank you in advance.