Final Projects, Part 1 & 2
Scattering by a Central Potential

Finn Joel Bjervig[*]

Computational Physics,
Department of Physics and Astronomy,
Uppsala university, Sweden

August 31, 2022

[*]Electronic address: `joelfbjervig@gmail.com`

Introduction

The first two parts in the final project concerns the scattering angle of a particle in the vicinity of a central potential. Imagine a particle traveling in a straight line. It continues to do so, with constant velocity, until it is close and passes a central potential.The particle is scattered into a new line of travel, deviating by some angle from its previous straight path , and will continue in this direction until something else occurs. The question is what this scattering looks like solved by a numerical method of choice, and solved analytically. How do they differ? What physical properties in the system changes the scattering angle?

Theory

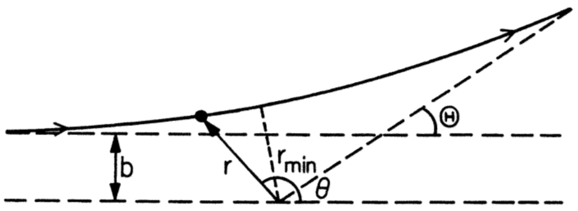


Figure 1: Scattering angle of a particle Θ in the vicinity of a central potential

The function that describes the deflection angle can be derived to be

$$\Theta = 2b\left[\int_b^{r_{max}} \frac{dr}{r^2}\left(1-\frac{b^2}{r^2}\right)^{-\frac{1}{2}} - \int_{r_{min}}^{r_{max}} \frac{dr}{r^2}\left(1-\frac{b^2}{r^2}-\frac{V}{E}\right)^{-\frac{1}{2}}\right]. \tag{1}$$

Analytical solution

To solve this we need to know what the potential looks like. Suppose the potential is a square potential such that

$$\begin{cases} V(r) = U_0 \quad \text{if} \quad r < r_{max} \\ V(r) = 0 \quad \text{if} \quad r > r_{max} \end{cases}. \tag{2}$$

It comes handy to make a substitution here such that the calculations become easier. Consider making the substitution $s = \frac{b}{r}$, $dr = -\frac{r^2}{b}ds$. Focusing on the first integral in equation 1, the substitution yields

$$-\frac{1}{b}\int_{r_{max}}^{b} \frac{1}{\sqrt{1-s^2}}ds \tag{3}$$

A recognizable integral indeed, it is equal to an trigonometric expression as such:

$$-\frac{1}{b}\left(\arcsin(s) - \arcsin(1)\right). \tag{4}$$

Substituting back from s, and applying the trigonometric shift of $\frac{\pi}{2}$, we have

$$I_1 = -\frac{1}{b}\left(\arcsin(\frac{b}{r_{max}}) - \frac{\pi}{2}\right) = \frac{1}{b}\arccos(\frac{b}{r_{max}}) \tag{5}$$

In a similar fashion the second integral can be computed to be

$$I_2 = \frac{1}{b}\arccos\left(\frac{b}{r_{max}}\frac{1}{\sqrt{1-\frac{V}{E}}}\right). \tag{6}$$

Thereby the entire integral of equation 1 is equal to

$$\Theta = 2\left(\arccos\left(\frac{b}{r_{max}}\right) - \arccos\left(\frac{b}{r_{max}}\frac{1}{\sqrt{1-\frac{V}{E}}}\right)\right). \tag{7}$$

Numerical method

There exists a variety of numerical quadrature methods to evaluate integrals: Trapezoidal, Simpsons rules, and Bodes rule. For Bodes rule, it is required that the number of intervals is divisible by $4n$, or that the number of equally spaced points $\{x_i\}$ is divisible by $4n+1$, where n is an positive integer. The evaluation for the integral by a set of five equally spaced points are

$$\int_{x_1}^{x_5} f(x)dx = \frac{2h}{25}\left(7f(x_1) + 32f(x_2) + 12f(x_3) + 32f(x_4) + 7f(x_5)\right). \tag{8}$$

To yield an evaluation of the entire integral, each sub-integral is summed up with a shift of four, such that the overlapping coefficients of 7 will sum up at the overlap. See figure 2. The error term for this method is quite good, at an order of $\mathcal{O}(h^7)$

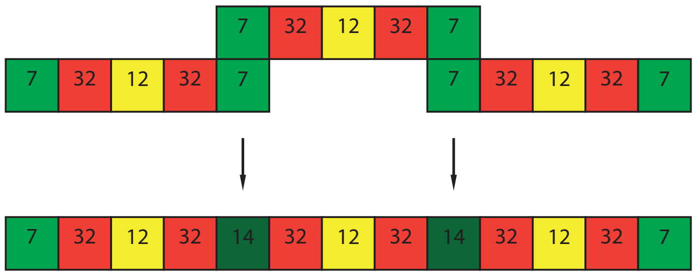


Figure 2: Bodes rule. Example of how three "subintegrals" would merge, such that the end coefficients of value 7 will become 14 at the overlap

If one considers a more smooth and realistic potential, such as the Leonard-Jones potential, the solution becomes quite different. Analytically, it is cumbersome to solve this problem. Therefore bodes method will suffice. see figure 3.

$$V(r) = 4v_0\left[\left(\frac{a}{r}\right)^{12} - \left(\frac{a}{r}\right)^{6}\right] \tag{9}$$

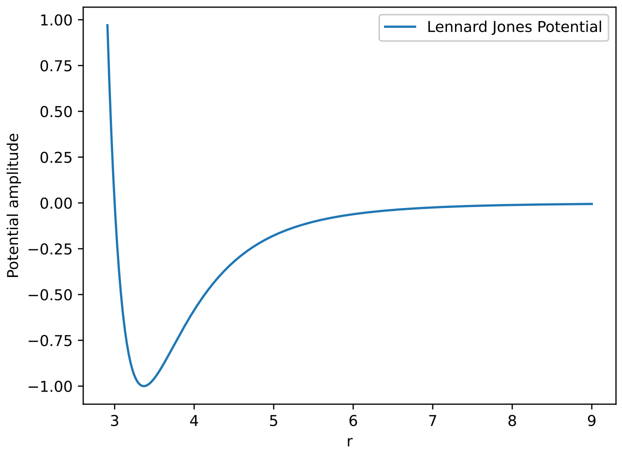
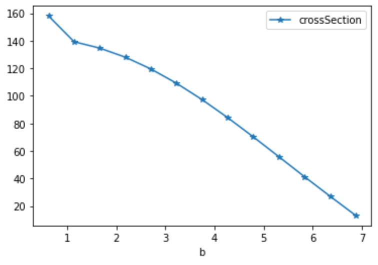


Figure 3: Lennard Jones potential, $r \in [r_{min}, r_{max}]$

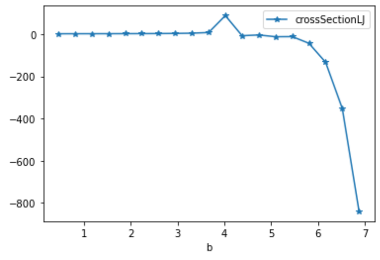
A experimental observable is the differential cross section which is connected to the deflection function as

$$\frac{d\sigma}{d\Omega} = \frac{b}{\sin(\Theta)} \left| \frac{db}{d\Theta} \right| \tag{10}$$

$db/d\Theta$ can be found by computing $\left(d\Theta/db\right)^{-1}$. The result for the differential cross section can be seen in figure 4.



(a) Square potential

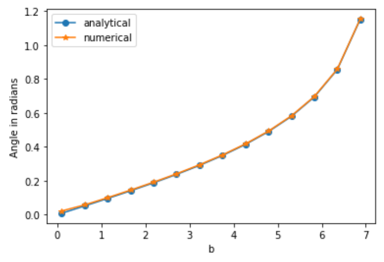


(b) Lennard Jones potential

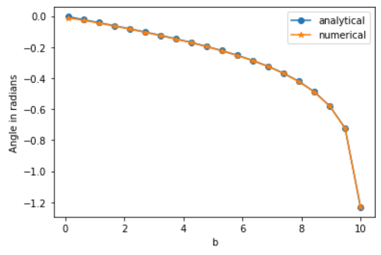
Figure 4: The differential cross section for two different potentials. See the code in appendix for more details

Discussion

As seen in 5, solving the integrals to yield the scattering angle by using Bodes rule was successful, and the error is tolerable although one would expect a smaller error by the fact that 100001 points were used to evaluate each of the integral terms, see figure 6. The set parameters the describe the system were chosen such that the solution wouldn't become imaginary. The restrictions were $E < V$ and $r > b$. Since the impact parameter b is the lateral distance from the initial particle trajectory, to the center potential (see figure 1), it is obvious that the radial coordinate r cant be smaller. $E < V$ is needed because otherwise the argument within the square root of equation 7 becomes negative, and thus the expression imaginary. The numerical solution using bodes method to calculate the deflection angle due to the presence of a square potential, seems to coincide with the analytical solution.

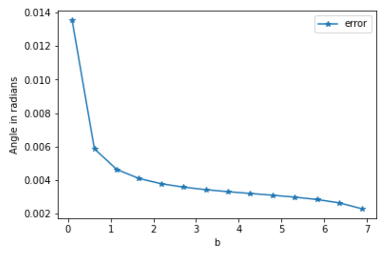


(a) Positive square potential $U > 0$

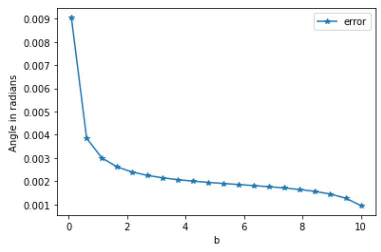


(b) Negative potential potential

Figure 5: The deflection angle due to a central potential. The potentials follows the definition in equation 2, but with opposite signs.



(a) Positive square potential $U > 0$



(b) Negative square potential $U < 0$

Figure 6: How much the numerical solution deviates from the analytical solution decreases initially with great speed and continues to do so for increasing b. The deviation refers to the absolute value of the error.

If one were to change the sign of the square potential from a positive value to a negative, the scattering angle will no longer increase with b. Which makes sense since the particle will attract instead of deflect, see the picture to the right in figure 5. But how does the particle scatter depending on its own kinetic energy E? You might suspect it to deflect more if its slowly passing the central potential, and less if it moves faster. Indeed, the graph generated in figure 7 we see how the deflection angle decreases as the particles energy increases. This is for a Lennard Jones potential, which stretches out into infinity, in contrast to the square potential.

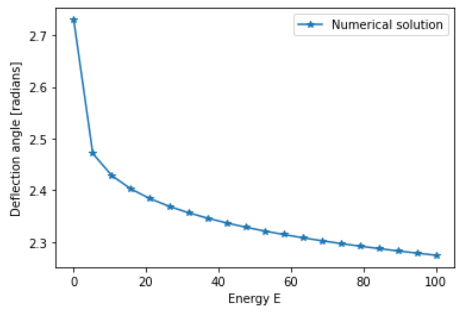


Figure 7: When the particle passes the potential slowly, its subjected to the force during a longer period and will therefore change its trajectory more.

1 Appendix

1.1 Task 1

```
# -*- coding: utf-8 -*-
"""
Created on Fri Mar  5 13:33:24 2021

@author: anton
"""
import numpy as np
import math
import matplotlib.pyplot as plt


def bode(f,a,b,N):
    if (N-1)%4 != 0:
        print("N is not a multiple of the form 4p+1")
        return None
    s=0
    h=(b-a)/N
    for i in range(0,N-3,4):
        integ=(7*f(a+i*h)+32*f(a+(i+1)*h)+12*f(a+(i+2)*h)+32*f(a+(i+3)*h)+7*f(a+
        integ*=(2*h)/45
        s+=integ
    return s



def theta(b,rmax,rmin):
    return np.subtract(bode(theta1,b+1e-6,rmax,N), bode(theta2,rmin+1e-6,rmax,N)



N=100001 #
E=2 #
V=-1
rmax=10

# first integrand term for b<r<r_max
theta1 = lambda r: 2*b*(r**(-2)*1/(np.sqrt(1-b**2/r**2)))
# second integrand term for r_min<r<r_max
theta2 = lambda r: 2*b*(r**(-2)*1/(np.sqrt(1-b**2/r**2-V/E)))
```

```
analytical = lambda b: 2*(np.arccos(b/rmax)-np.arccos(b/rmax*(E/(E-V))**0.5))



bvec=np.linspace(0.1,rmax,20)
rmin=bvec*np.sqrt(E/(E-V))
its=len(rmin[rmin<rmax])
bvec=bvec[0:its]
rmin=rmin[0:its]
sol=np.zeros(len(bvec))
error=np.zeros(len(bvec))
thetaPrime=np.zeros(len(bvec)-1)
crossSec=np.zeros(len(bvec)-1)
h=bvec[1]-bvec[0]

for i in range(0,its):
    b=bvec[i]
    sol[i]=(theta(b,rmax,rmin[i]))
    error[i]=(abs(sol[i]-analytical(b)))


for j in range(1,its):
    thetaPrime[j-1]=(sol[j]-sol[j-1])/((h))
    crossSec[j-1]=(bvec[j])/(math.sin(sol[j])*abs(thetaPrime[j-1]))


plt.figure(1)
plt.plot(bvec,analytical(bvec),label="analytical",marker='o')
plt.plot(bvec,sol,label="numerical",marker='*')
plt.xlabel("b")
plt.ylabel('Angle in radians')
plt.legend(loc='best')
plt.show()


plt.figure(2)
plt.plot(bvec,error,label="error",marker='*')
plt.xlabel("b")
plt.ylabel('Angle in radians')
plt.legend(loc='best')
plt.show()

plt.figure(3)
# plt.plot(bvec[1:its],thetaPrime,label="thetaPrime",marker='*')
plt.plot(bvec[1:its],crossSec,label="crossSection",marker='*')
plt.xlabel("b")
plt.ylabel('')
plt.legend(loc='best')
plt.show()

plt.figure(4)
plt.plot(bvec[1:its],thetaPrime,label="thetaPrime",marker='*')
# plt.plot(bvec[1:its],crossSec,label="crossSection",marker='*')
plt.xlabel("b")
plt.ylabel('')
plt.legend(loc='best')
plt.show()
```

1.2 Task 2

```
# -*- coding: utf-8 -*-
"""
Created on Wed Mar 17 15:34:45 2021

@author: anton
"""
# -*- coding: utf-8 -*-
"""
Created on Fri Mar  5 13:33:24 2021

@author: anton
"""
import numpy as np
import math
import matplotlib.pyplot as plt

def bissection(a,b,f,maxit):
    if f(a)*f(b) >= 0:
        print("Doesn't␣change␣sign␣on␣interval")
        return None
    c=(a+b)/2
    for i in range(maxit):
        c=(a+b)/2
        if f(c)*f(a)<0:
            b=c
        elif f(c)*f(b)<0:
            a=c
        elif f(c)==0.0:
            print("Exact␣solution␣found.")
            return c
    return (a+b)/2

def bode(f,a,b,N):
    if (N-1)%4 != 0:
        print("N␣is␣not␣a␣multiple␣of␣the␣form␣4p+1")
        return None
    s=0
    h=(b-a)/N
    for i in range(0,N-3,4):
        integ=(7*f(a+i*h)+32*f(a+(i+1)*h)+12*f(a+(i+2)*h)+32*f(a+(i+3)*h)+7*f(a+
        integ*=(2*h)/45
        s+=integ
    return s
```

```
# first integrand term for b<r<r_max
theta1 = lambda r: 2*b*(r**(-2)*1/(np.sqrt(1-b**2/r**2)))

# numberical solution
def theta(b,E,rmax,rmin):
    return np.subtract(bode(theta1,b+1e-6,rmax,N), bode(theta2,rmin+1e-6,rmax,N)

# part 2
# LJ dependet on E
V=1
a=3.333
b=1
rmax=3*a
N=100001
# Energy is varying
Evec = V*np.linspace(0.1,100,20)

# Leonard Jones potential
pot = lambda r: 4*V*((a/r)**12-(a/r)**6)

rminfunc = lambda r,E: 1-b**2/r**2-pot(r)/E
rmin = np.zeros(len(Evec))
sol = np.zeros(len(Evec))

 # Find the positive root of the function
for i in range(0,len(Evec)):
    rminfuncE = lambda r: 1-b**2/r**2-pot(r)/Evec[i]
    rmin[i] = bissection(0.1, rmax, rminfuncE, 100)
    E=Evec[i]
    theta2 = lambda r: 2*b*(r**(-2)*1/(np.sqrt(1-b**2/r**2-pot(r)/E)))
    sol[i]= theta(b,E, rmax, rmin[i])



# part 2 cross section
bvec=np.linspace(0.1,6.87368,20)
h=bvec[1]-bvec[0]
E=2

theta2 = lambda r: 2*b*(r**(-2)*1/(np.sqrt(1-b**2/r**2-pot(r)/E)))
def theta(b,rmax,rmin):
    return np.subtract(bode(theta1,b+1e-6,rmax,N), bode(theta2,rmin+1e-6,rmax,N)


rminLJ=np.zeros(len(bvec))
```

```
solLJ=np.zeros(len(bvec))
for i in range(0,len(bvec)):
    b=bvec[i]
    rminfuncLJ = lambda r: 1-b**2/r**2-pot(r)/E
    rminLJ[i] = bissection(0.1, rmax, rminfuncLJ, 100)
    solLJ[i]=theta(b,rmax,rminLJ[i])

thetaPrimeLJ=np.zeros(len(bvec)-1)
crossSecLJ=np.zeros(len(bvec)-1)
for j in range(1,len(bvec)):
    thetaPrimeLJ[j-1]=((solLJ[j]-solLJ[j-1])/((h)))
    crossSecLJ[j-1]=(bvec[j])/(math.sin(solLJ[j])*abs(thetaPrimeLJ[j-1]))



plt.figure(1)
plt.plot(Evec,sol,label="Numerical solution",marker='*')
plt.xlabel("Energy E")
plt.ylabel('Deflection angle [radians]')
plt.legend(loc='best')
plt.show()

plt.figure(2)
plt.plot(bvec,solLJ,label="Numerical solution",marker='*')
plt.xlabel("b")
plt.ylabel('Deflection angle [radians]')
plt.legend(loc='best')
plt.show()

plt.figure(3)
plt.plot(bvec[1:100],crossSecLJ,label="crossSectionLJ",marker='*')
plt.xlabel("b")
plt.ylabel('')
plt.legend(loc='best')
plt.show()

plt.figure(47)
plt.plot(bvec[1:100],thetaPrimeLJ,label="thetaPrimeLJ",marker='*')
plt.xlabel("b")
plt.ylabel('')
plt.legend(loc='best')
plt.show()
```