

UNIVERSIDAD NACIONAL DE COLOMBIA

Thesis submitted for the degree

Physics bachelor thesis

Interactive application of nuclear elastic scattering based on the R-matrix method on Python

by

Andres Felipe Rincón

Supervisor: Carolina Pinilla

July 2019



Abstract

We present an interactive application made in the Python programming language, where the elastic cross sections between two nuclei are computed. There are hardly ever found interactive cross elastic section applications on internet, so this work is very useful to strengthen the knowledge of physics and engineers students about scattering process between nuclei. Some elastic cross are also computed and shown between alpha particles, ^{58}Ni , ^{12}C , ^{90}Ni , ^{208}Pb , which are predicted with accuracy according to experimental results and other works.

Nomenclature

In the python code

A_p Mass number of the projectile.

z_p Atomic number of the projectile.

A_t Mass number of the target.

z_t Atomic number of the target.

$h2mu$ $20.736 \times (A_p + A_t) / (A_p \times A_t)$.

e^2 Electron constant (1.44MeVfm).

To my parents.

Contents

Abstract	ii
Nomenclature	iii
Contents	v
List of Figures	vii
List of Tables	viii
Introduction	1
1 The elastic scattering cross section	3
1.1 Short-range potential scattering	5
1.2 Coulomb scattering	6
1.3 Coulomb plus nuclear scattering	8
2 R matrix theory	10
3 Optical potential	15
4 Scattering matrix	18
5 Python Code	20
5.1 Python Interface	22

6	Elastic cross sections	24
6.1	^{58}Ni Elastic cross section with an alpha particle	24
6.2	^{208}Pb Elastic cross section with an alpha particle	26
6.3	^{90}Zr Elastic cross section with an ^{12}C nucleus	27
	Conclusion	29
	A The Code	30
	Bibliography	44

List of Figures

5.1	Window Interface	23
6.1	α - ^{58}Ni Real Optical Potential	25
6.2	α - ^{58}Ni Scattering. Similar theoretical prediction behaviour can be found in [3]	25
6.3	α and ^{208}Pb Real Optical Potential at 110 MeV	26
6.4	α - ^{208}Pb Scattering [3].	27
6.5	^{12}C and ^{90}Zr Real Optical Potential at 175 MeV.	28
6.6	^{12}C - ^{90}Zr Scattering [5].	28

List of Tables

6.1	Parameters of the ^{58}Ni and α scattering [3]	24
6.2	Parameters ^{208}Pb and α scattering. [3]	26
6.3	Parameters ^{90}Zr and ^{12}C nucleus scattering. [5]	27

Introduction

The differential scattering cross section is a powerful tool, which gives information about the number of particles scattered into an element of a solid angle [2]. The situation of scattering has been studied in quantum mechanics theory for decades. This topic has many applications in nuclear physics, in the study of fundamental particles, and the characterisation of molecules and solids.

There is a difference between the classical and the quantum scattering. Through this paper we only explore the quantum scattering because we work with nuclei. Here the nuclear, Coulomb and effective potentials are considered.

The description of the potential between nuclei is difficult, because there is not expression as it is the case for the Coulomb potential. There exists a lot of models, which describes the behaviour of the interaction between nuclei. One of them is to treat the interaction between nucleons as an effective short-range potential in a Woods-Saxon form.

The elastic cross section is constructed taking an important physical quantity, phase shift δ_l . Which is used to compute the nuclear scattering cross section. Here, the phase shift angle is calculated using the R-matrix method.

In this work, we computed the elastic cross section for different pairs of nuclei, which spin-orbit interaction is ignored, and because of that, the energy of the collision has to be

larger than the energy of the Coulomb barrier of the interacting nuclei must be in closed shells.

Interactive applications about the scattering processes are hardly ever found on internet. The aim of the work is to create an interactive application, which can be used by the physics students to get a better knowledge about the scattering processes.

This document is organized as follows: chapter 1 describes the scattering cross section with an analysis of the Coulomb and the nuclear scattering [7].

In chapter 2, the theory of the R-matrix method is exposed [4]. This method is based on the variational method, and calculates the Scattering matrix, which is used to obtain the Scattering Cross Section (It is explain with more details in chapter 4). The R matrix method also uses special functions that work under the Gaussian quadrature.

In chapter 3, we exposed the optical potential, its parts, the quantum mechanics continuity equation and how it is different from ones which use real the potentials.

In chapter 5, the program carried out in python is presented, the libraries and the parameters are also explained. In the Appendix A, the code is shown in details.

In Chapter 6 shows the results of the scattering cross section of the different nuclei and its potentials are exposed in the next part of the text, and finally the conclusion are exposed at the end of the text.

Chapter 1

The elastic scattering cross section

If the scattering of two collision partners is described by the potential $V(r)$, where $r = \left| \vec{r}_1 - \vec{r}_2 \right|$ is the distance between the two particles, then we say that the scattering is governed by a central potential. [6]

The two-body problem can be reduced to a problem of a point particle in a fixed potential, with this aim we introduce the center-of-mass coordinate \vec{R} and the relative coordinate \vec{r} are coordinates

$$\vec{R} = \frac{m_1 \vec{r}_1 + m_2 \vec{r}_2}{m_1 + m_2}, \quad (1.1)$$

$$\vec{r} = \vec{r}_1 - \vec{r}_2, \quad (1.2)$$

where \vec{r}_1, \vec{r}_2 and m_1, m_2 are the position and the mass of the projectile and, target respectively.

And the corresponding the total momentum \vec{P} and relative momentum \vec{p} are written as

$$\vec{P} = \vec{p}_1 + \vec{p}_2 \quad (1.3)$$

$$\vec{p} = \frac{m_1 \vec{p}_1 + m_2 \vec{p}_2}{m_1 + m_2}. \quad (1.4)$$

The equations 1.1, 1.4 allow to get a Hamiltonian that relates the relative motion and the center of mass kinetic energy.

$$H_{TOT} = \frac{p_1^2}{2m_1} + \frac{p_2^2}{2m_2} + V(r) = \frac{P^2}{2M} + H \quad (1.5)$$

with

$$H = \frac{p^2}{2\mu} + V(r) \quad (1.6)$$

Here, $\mu = \frac{m_1 m_2}{m_1 + m_2}$ and $M = m_1 + m_2$, and the H and \vec{P} can be rewritten in terms of the operator form as

$$\vec{P} = -i\hbar \nabla_R, \quad H = -\frac{\hbar^2}{2\mu} \nabla^2 + V(r), \quad (1.7)$$

where ∇^2 is the Laplacian operator in the relative coordinate r .

The two-body wave function can be written in terms of the center of a mass motion and a function $\psi(\vec{r})$ describing the relative motion

$$\Psi(\vec{r}_1, \vec{r}_2) = e^{i\vec{K} \cdot \vec{R}} \psi(\vec{r}) \quad (1.8)$$

with $\vec{K} = \vec{P}/\hbar$, the wave function $\psi(\vec{r})$ satisfies the stationary one body *Schrödinger* equation

$$H\psi(\vec{r}) = E\psi(\vec{r}). \quad (1.9)$$

The differential cross section $d\sigma(\Omega)$ for elastic scattering is defined as the ratio of the asymptotic probability current flowing radially into an element of solid angle $d\Omega$, over the probability density of the incident wave.

$$d\sigma(\Omega) = \frac{\text{probability current into } d\Omega \text{ in the direction } \Omega}{\text{probability current density of the incident wave}}. \quad (1.10)$$

The differential cross section and the scattering amplitude are related as

$$\frac{d\sigma(\Omega)}{d\Omega} = |f(\Omega)|^2. \quad (1.11)$$

1.1 Short-range potential scattering

If the scattering of two collision partners is described by the potential $V(r)$, where $r = |\vec{r}_1 - \vec{r}_2|$ is the distance between the two particles, then we say that the scattering is governed by a central potential.

We define a short-range potential as the one that is less singular than $1/r^2$ at the origin. This condition is sufficient to guarantee that the wave function must go over into a free wave function at large distances, so the potential vanishes beyond some distance a .

$$V(r) = 0 \quad \text{for} \quad r \geq a. \quad (1.12)$$

The boundary condition on the scattering wave function at large distances, $\psi(\vec{r})$ consists of a scattered part which has the form of an outgoing spherical wave, emerging from the scattering centre, and an incident plane wave with momentum \vec{p} , where $\frac{p^2}{2\mu} = E$.

$$\psi(\vec{r}) \rightarrow e^{i\vec{k} \cdot \vec{r}} + f(\Omega) \frac{e^{i\vec{k} \cdot \vec{r}}}{r}; \quad r \rightarrow \infty \quad (1.13)$$

Here $\vec{k} = \vec{p}/\hbar$ and Ω is the solid angle associated with the coordinate \vec{r} , $f(\Omega)$ is called the scattering amplitude and it determines the strength of the scattered wave.

1.2 Coulomb scattering

The radial *Schrödinger* equation for two quantum bodies interacting through the Coulomb potential at $E > 0$ is given by

$$\left[\nabla^2 + k^2 - \frac{2k\eta}{r} \right] \psi_C(\vec{r}) = 0. \quad (1.14)$$

with

$$k = \frac{\sqrt{2\mu E}}{h}, \quad (1.15)$$

$$\eta = \frac{Z_1 Z_2 e^2 \mu}{h^2 k}, \quad (1.16)$$

and $Z_1 e$, $Z_2 e$ of the bodies.

In order to solve the equation above, it is useful to take the ansatz. The explanation can be explored with more details in [1]

$$\psi_C(r) = C e^{ikz} g(r - z). \quad (1.17)$$

Inserting the ansatz in the Schrödinger equation and if we name $\nu = r - z$, we have

$$\nu g''(\nu) + (1 - ik\nu)g'(\nu) - \eta k g(\nu) = 0. \quad (1.18)$$

If we rename $s = ik\nu$, $w(s) = g(\nu)$, $d = -i\eta$ and $c = 1$. The differential equation becomes

$$s w''(s) + (c - s)w'(s) - d w(s) = 0, \quad (1.19)$$

The solution of this equation has the confluent hypergeometric function ($F_1(d, c, s)$), then

$$\psi_C(r) = C e^{ikz} F_1(-i\eta, 1, ik(r - z)). \quad (1.20)$$

The function $F_1(-i\eta, 1, ik(r-z))$ can be written in function of W_1 and W_2 (integrals in the converse plane) as

$$F_1(-i\eta, 1, ik(r-z)) = W_1(-i\eta, 1, ik(r-z)) + W_2(-i\eta, 1, ik(r-z)), \quad (1.21)$$

The wave function $\psi_C(r)$ is then given by the sum

$$\psi_C(r) = \psi_i(r) + \psi_s(r), \quad (1.22)$$

Where

$$\psi_i(r) = Ce^{ikz}W_1(-i\eta, 1, ik(r-z)), \quad (1.23)$$

$$\psi_s(r) = Ce^{ikz}W_2(-i\eta, 1, ik(r-z)), \quad (1.24)$$

Here

$$C = A\Gamma(1+i\eta)e^{-\pi\eta/2} \quad (1.25)$$

Aiming at obtaining $f_c(0)$, we take the limit $|r-z| \rightarrow \infty$

$$\psi_i(r)_{|r-z| \rightarrow \infty} = Ae^{ikz} \cdot [e^{-i\eta \ln k(r-z)}] \quad (1.26)$$

$$\psi_s(r)_{|r-z| \rightarrow \infty} = A \frac{e^{ikz}}{r} \cdot [f_c(\theta) e^{-i\eta \ln(2kr)}] \quad (1.27)$$

It is possible to separate the solutions as $\psi_C(r) = \psi_i(r) + \psi_s(r)$

$$\psi_C(r)_{|r-z| \rightarrow \infty} = A \left(e^{ikz} \cdot [e^{-i\eta \ln k(r-z)}] + f_c(\theta) \frac{e^{ikz}}{r} [e^{-i\eta \ln(2kr)}] \right) \quad (1.28)$$

$f_c(\theta)$ is the Coulomb scattering amplitude, which is given by

$$f_c(\theta) = -\frac{\eta}{2k \sin\left(\frac{\theta}{2}\right)^2} e^{2i(\sigma_0 - \eta \ln \sin(\frac{\theta}{2}))} \quad (1.29)$$

Where σ_0 is the s-wave Coulomb phase shift, which is given by

$$\sigma_0 = \arg(\Gamma(1 + i\eta)). \quad (1.30)$$

1.3 Coulomb plus nuclear scattering

The Coulomb potential generated by the nuclear charge distributed over the volumes of the nuclei must be supplemented with the short-range nuclear optical potential in order to describe the interaction of the nuclei. The uniform charge distributions are assumed to be spherical, so the Coulomb interaction for spherical nuclei has the form [6]

$$V_c(r) = \begin{cases} Z_1 Z_2 e^2 / (2R_c) [3 - (r/R_c)^2] & \text{for } r < R_c \\ Z_1 Z_2 e^2 / r & \text{for } r > R_c \end{cases} \quad (1.31)$$

Here R_c is the Coulomb radius, which is usually the sum of the charge radii of the projectile and the target.

The radial *Schrödinger* equation for Coulomb plus nuclear scattering is

$$\left(\frac{d^2}{dr^2} - \frac{2\mu}{\hbar^2} [V_c + V_N] - \frac{l(l+1)}{r^2} + k^2 \right) \psi_l(r) = 0, \quad (1.32)$$

with the condition at the origin $\psi(0) = 0$. A real solution at positive energy E behaves asymptotically as

$$\psi_l(r) \xrightarrow{r \rightarrow \infty} \cos \delta_l F_l(\eta, kr) + \sin \delta_l G_l(\eta, kr), \quad (1.33)$$

The important physical quantity in equation (1.33) is the phase shift δ_l .

The wave function $\psi(r)$ is also constructed in terms of the functions $\psi_c(r)$ and $\psi_N(r)$ where

$$\psi_N(r) \rightarrow f_N(\theta) \frac{e^{i[kr - \eta \ln(2kr)]}}{r} \quad ; \quad r \rightarrow \infty, \quad (1.34)$$

The $\psi_N(r)$ in equation (1.34) can be also written as summation for $r \rightarrow \infty$ due to the asymptotic form of the nuclear part.

$$\psi_N(r) \rightarrow \frac{1}{2ikr} \sum_{l=0}^{\infty} (2l+1) i^l e^{i\sigma_l} (e^{2i\delta_l} - 1) e^{i[kr - \eta \ln(2kr) - \frac{\pi}{2}l + \sigma_l]} P_l(\cos \theta) \quad for \quad r \rightarrow \infty \quad (1.35)$$

Here $\sigma_l = \arg(\Gamma(1 + l + i\eta))$. Comparing this expression with the one above, we have

$$f_N(\theta) = \frac{1}{2ik} \sum_{l=0}^{\infty} (2l+1) i^l e^{2i\sigma_l} (e^{2i\delta_l} - 1) P_l(\cos \theta). \quad (1.36)$$

As a result, the complete scattering amplitude is given by

$$f(\theta) = f_c(\theta) + f_N(\theta). \quad (1.37)$$

Chapter 2

R matrix theory

The phase shift δ_l can be obtained using boundary methods as the Numerov or the R-matrix methods. The latter is used in this thesis.

In order to implement the R-matrix method, the configuration space must be divided at the channel radius a into an internal and external regions. In the internal region with $r \leq a$, the nuclear and Coulomb potential interactions dominate and in the external region, with $r > a$, the nuclear potential vanishes.

The R-matrix method uses the Lagrange-mesh technique, which is an approximate variational calculation. This method employs a set of N functions $f_i(x)$, defined in an interval (a,b) , called Lagrange-functions. The mesh is made of a set of N x_i points related with the zeros of a polynomial in the Gauss quadrature associated with the mesh [4]

$$\int_a^b g(x) dx \approx \sum_{k=1}^N \lambda_k g(x_k), \quad (2.1)$$

where λ_k are the weight functions.

The channel radius is chosen large enough so that $V(r)$ can be approximated by $V_c(r)$ in the external region at the required accuracy.

The wave function $\psi_l(r)$ in equation (1.32) is approximated in the external region given by the exact asymptotic expression

$$\psi_l(r)^{ext} = C_l [I_l(kr) - U_l O_l(kr)]. \quad (2.2)$$

Here $I_l(kr)$ and $O_l(kr)$ are the linear combinations of the Coulomb functions F_l and G_l . ($I_l = G_l - iF_l$ and $O_l = G_l + iF_l$).

The wave function in the internal region is expanded over some finite basis involving N linearly independent functions $\varphi_j(r)$ as

$$\psi_l(r)^{int} = \sum_{j=1}^N c_j \varphi_j(r). \quad (2.3)$$

The functions φ_j are not necessary orthogonal although they must vanish at the origin.

If we separate the angular part from the *Schrödinger* equation, the radial *Schrödinger* equation becomes

$$(H_l - E) \psi_l(r) = 0, \quad (2.4)$$

Here H_l is defined as $H_l = T_l + V(r)$, which is not Hermitian over the internal region $(0, a)$. This problem can be solved with the help of the surface operator L introduced by Bloch [4]

$$L = \frac{\hbar^2}{2\mu} \delta(r - a) \frac{d}{dr}. \quad (2.5)$$

The inhomogeneous Bloch–*Schrödinger* equation is approximated by the *Schrödinger* equation in the internal region

$$(H_l + L - E) \psi(r)^{int} = L \psi(r)^{ext}, \quad (2.6)$$

where the external solution is used in the right-hand side.

The mathematical problem is complemented with the continuity condition already included in Eq (2.5) and (2.6)

$$\psi(a)^{int} = \psi(a)^{ext} \quad (2.7)$$

and

$$\left. \frac{d\psi(r)^{int}}{dr} \right|_{r=a} = \left. \frac{d\psi(r)^{ext}}{dr} \right|_{r=a} \quad (2.8)$$

The inhomogeneous Bloch–Schrödinger equation can be written as a Green differential equation [4]. The R matrix at energy E can be obtained from the wave function $\psi(r)$ as

$$\psi(a) = R_l(E) [a\psi'(a)] \quad (2.9)$$

Taking the equation above and because of the green function, $R_l(E)$ matrix can be written as [4]

$$R_l(E) = \frac{h^2}{2\mu a} G_l(a, a) \quad (2.10)$$

$\psi(r)$ is projected on $\varphi_j(r)$ and taking the inhomogeneous Bloch–Schrödinger equation and R matrix above, we have

$$\sum_{j=1}^N C_{ij}(E, B) c_j = \frac{h^2}{2\mu a} \varphi_i(a) (a\psi'(a)^{ext}). \quad (2.11)$$

The elements of C are defined as

$$C_{i,j}(E, B) = \langle \varphi_i | T_l + (B) + V - E | \varphi_j \rangle, \quad (2.12)$$

and therefore, the R matrix can be again re-written as

$$R_l(E, B) = \frac{h^2}{2\mu a} \sum_{i,j=1}^N \varphi_i(a) (C^{-1})_{ij} \varphi_j(a). \quad (2.13)$$

The Lagrange functions are used as base functions $\varphi_i(r)$ in the $(0, a)$ interval, where $i = 1, 2, \dots, N$. These functions are really useful because they do not loss accuracy due to the Gauss quadrature. They can be written as

$$\varphi_i(r) = (-1)^{N+i} \left(\frac{r}{ax_i} \right)^n \sqrt{ax_i(1-x_i)} \frac{P_N(2r/a-1)}{r-ax_i}. \quad (2.14)$$

The regularization coefficient n is taken as $n = 1$. This ensures that the wave function vanishes at the origin and allows the accurate treatment of the Coulomb potential at the Gauss approximation.

P_N is a Legendre polynomial of order N and the mesh point satisfy

$$P_N(2x_i - 1) = 0. \quad (2.15)$$

The elements of the Matrix C follow are given by [4].

For $i = j$

$$\langle \varphi_i | T_{i,j} + L_{i,j} | \varphi_j \rangle = \frac{h^2}{2\mu} \frac{(4N^2 + 4N + 3) x_i (1 - x_i) - 6x_i + 1}{3a^2 x_i^2 (1 - x_i)^2}. \quad (2.16)$$

For $i \neq j$

$$\langle \varphi_i | T_{i,j} + L_{i,j} | \varphi_j \rangle = \frac{h^2}{2\mu a^2 [x_i x_j (1 - x_i) (1 - x_j)]^{1/2}} \times \left[N^2 + N + 1 + \frac{x_i + x_j - 2x_i x_j}{(x_i - x_j)^2} - \frac{1}{1 - x_i} - \frac{1}{1 - x_j} \right] \quad (2.17)$$

At the corresponding Gauss approximation the matrix elements take a very simple form for the potential

$$V_{i,j} = V(ax_i) \delta_{i,j}. \quad (2.18)$$

Chapter 3

Optical potential

The complex nuclear potential (usually called optical potential) enables us to describe some features of the elastic scattering of composite particles. The imaginary part of this interaction mocks up the loss of flux into non-elastic processes.

If the Hamiltonian is Hermitian, i.e if there is not an imaginary term in the nuclear potential, the total current is

$$\vec{j} = \frac{\hbar}{2\mu i} \left[\psi^*(\vec{r}) \nabla \psi(\vec{r}) - \psi(\vec{r}) \nabla \psi^*(\vec{r}) \right]. \quad (3.1)$$

This equation satisfies the continuity equation

$$\vec{\nabla} \cdot \vec{j} = 0. \quad (3.2)$$

For complex potentials the continuity equation is different. In this case the Hamiltonian has the form

$$H = -\frac{\hbar^2}{2\mu} \nabla^2 + V(r), \quad V(r) = U(r) + iW(r) \quad (3.3)$$

where $U(r)$ is real and $W(r)$ is a short range negative function of r .

In case of complex nuclear potentials, the continuity equation must be modified. In order to archive it, the Schrodinger equation and its complex conjugate are rewritten as:

$$\left[-\frac{\hbar^2}{2\mu} \nabla^2 + U(r) + iW(r) \right] \psi(\vec{r}) = E\psi(\vec{r}) \quad (3.4)$$

$$\left[-\frac{\hbar^2}{2\mu} \nabla^2 + U(r) - iW(r) \right] \psi^*(\vec{r}) = E\psi^*(\vec{r}) \quad (3.5)$$

Then, multiplying by $\psi^*(r)$ and $\psi(r)$

$$-\psi^*(\vec{r}) \frac{\hbar^2}{2\mu} \nabla^2 \psi(\vec{r}) + \psi^*(\vec{r}) U(r) \psi(\vec{r}) + i\psi^*(\vec{r}) W(r) \psi(\vec{r}) = E\psi(\vec{r}) \psi^*(\vec{r}) \quad (3.6)$$

$$-\psi(\vec{r}) \frac{\hbar^2}{2\mu} \nabla^2 \psi^*(\vec{r}) + \psi(\vec{r}) U(r) \psi^*(\vec{r}) - i\psi(\vec{r}) W(r) \psi^*(\vec{r}) = E\psi^*(\vec{r}) \psi(\vec{r}) \quad (3.7)$$

If the equations (3.6) and (3.7) are subtracted we get

$$\frac{\hbar}{2\mu i} [\psi^*(\vec{r}) \nabla^2 \psi(\vec{r}) - \psi(\vec{r}) \nabla^2 \psi^*(\vec{r})] = \frac{2}{\hbar} W(r) |\psi(\vec{r})|^2 \quad (3.8)$$

Thus, we end up with

$$\vec{\nabla} \cdot \vec{j} = \frac{2}{\hbar} W(r) |\psi(\vec{r})|^2. \quad (3.9)$$

In general, the real or imaginary parts of the optical potential can be written as

$$V_N(r) = V_v f_{ws}(r, a_v, R_v) + V_D \frac{d}{dr} f_{ws}(r, a_D, R_D) + V_{SO} \frac{1}{r} \frac{d}{dr} f_{ws}(r, a_{SO}, R_{SO}) \quad (3.10)$$

Here $V_v f_{ws}(r, a_v, R_v)$ represents the Volumetric part, $V_D \frac{d}{dr} f_{ws}(r, a_D, R_D)$ the Superficial part and $V_{SO} \frac{1}{r} \frac{d}{dr} f_{ws}(r, a_{SO}, R_{SO})$ the Spin-orbit part, where

$$f_{ws}(r) = \frac{1}{1 + e^{(r-R_i)/a_i}} \quad (3.11)$$

is a Woods-Saxon form factor and a is the length, which represents the "surface thickness", $R_i = r_0 A^{1/3}$, A is the mass number and r_0 is a fixed constant.

Chapter 4

Scattering matrix

The scattering matrix gives information about the initial and the final state of a physical system undergoing a scattering process.

The Scattering-matrix elements must be expressed in terms of the R-matrix. In the external region of the potential ($r > a$). The radial wave function becomes

$$\psi(r)^{ext} = \frac{i}{2} \left[H_l^{(-)}(\eta, kr) - U_l H_l^{(+)}(\eta, kr) \right] \quad (4.1)$$

where the incoming and outgoing Ricatti-Haenkel functions $H_l^{(-)}(\eta, kr)$ and $H_l^{(+)}(\eta, kr)$ behave asymptotically as

$$H_l^{(\pm)}(\eta, kr) \rightarrow e^{\pm i(kr - l\pi/2)}. \quad (4.2)$$

The required relation in order to obtain the scattering function in terms of the Hankel function is readily obtained from the external wave function at the surface, following the rule

$$U_l = \frac{H_l^{(-)}(\eta, ka)}{H_l^{(+)}(\eta, ka)} \left[\frac{1 - \left(L_l^{(+)} \right)^* R_l(E)}{1 - L_l^{(+)} R_l(E)} \right], \quad (4.3)$$

Where $L_l^{(+)}$ is the logarithmic derivative of the outgoing Ricatti-Hankel function

$$L_l^{(+)} = \frac{ka}{H_l^{(+)}(\eta, ka)} \left[\frac{dH_l^{(+)}(\eta, \rho)}{d\rho} \right]_{\rho=ka}. \quad (4.4)$$

Once the R matrix is computed, it is possible to find the scattering matrix that shows up the asymptotic scattering wave function and it is given as

$$U_l = \frac{H_l^{(-)}(\eta, ka) - akR_l(E) H_l^{(-)'}(\eta, ka)}{H_l^{(+)}(\eta, ka) - akR_l(E) H_l^{(+)' }(\eta, ka)} \quad (4.5)$$

Chapter 5

Python Code

The program constructed in Python, has been constructed with eight libraries: sys, tkinter, matplotlib, mpmath, scipy and numpy. They are described in the following:

- **Numpy:** Provides various derived objects (such as masked arrays and matrices), a multidimensional array object, and an assortment of routines for fast operations on arrays, including mathematical, basic linear algebra and sorting.
- **Scipy:** Provides linear algebra and statistical tools, user-friendly and efficient numerical routines, and many useful functions such as the Legendre and its respective roots, which are used in our python code.
- **Mpmath:** Provides a lot of special functions, which are used in physics and in quantum mechanics such as the Bessel and the Coulomb functions, which are an important part of this code.
- **Tkinter:** is a standard Python interface, which gives a independent platform windowing toolkit. A lot of information can be introduce in the window, after that, these information is processed to the Python code to get some results about physical simulations.

- **Matplotlib:** is used to generate histograms, power spectra, bar charts, errorcharts, scatterplots, plots, etc.
- **Pandas:** is a library, which is used to manipulate data and to analyse it. The data is analysed by a Data frame and a Series.

In the first part of the code, the Regularized Legendre functions must be constructed evaluated in a . From the scipy library, the Legendre's roots can be obtained. Then we can redifined them in order to satisfy the equation

$$P_N(2x_i - 1) = 0. \quad (5.1)$$

As a second part, the Regularized Legendre functions are evaluated in a , because it saves computational time due to the fact that $P_N(1) = 1$. As a result, the Regularized Legendre functions evaluated in a becomes

$$\varphi_i(a) = (-1)^{N+i} \left(\frac{1}{x_i}\right)^n \sqrt{ax_i(1-x_i)} \frac{1}{a-ax_i} \quad (5.2)$$

Once the Regularized Legendre functions are constructed, the R-matrix is obtained as a function of the potential and the C-matrix as it is exposed in the R matrix chapter.

The C matrix is constructed in N dimensions. Theorretically N must be infinite, but it cannot be done computationally. Therefore we can take N sufficiently to rise convergence of the physical quantities of interest. After that, the C-matrix is filled by zeros using the numpy extension, which is named "matrix".

Then the zeros are replaced with the C-matrix elements exposed in equations 2.16 and 2.17). After that, using the command `linalg.inv` of the numpy library, the inverse of C-matrix is calculated.

The R-matrix is obtained by the projection of the vector that has the Regularized legendre functions of delay on the matrix C inverse matrix.

Once the R matrix is obtained, the Scattering matrix is constructed following the formula (2.13). The derivative of the Hankel function is calculated by the definition of finite difference.

Finally, The Scattering matrix is obtained from the R matrix and a while cycle is used to creating the summation of the Nuclear Elastic Cross Section (equation 1.36). After that, it is summed with the Coulomb Scattering Cross Section and as a result the Scattering Cross Section is obtained.


5.1 Python Interface

The windows interface is constructed using the tkinter library with the command Tk, the parameters, which are going to be included, are inserted by the Label command. The window that appears in the screen of the computer is shown below 5.1 :

The following parameters in the window interface (5.1) describes some data of nuclei to be studied

- Ap: Mass number of the projectile.
- Az: Atomic number of the target.
- Zp: Mass number of the projectile.
- Zt: Atomic number of the target.
- Rr: Real radius of the Woods-Saxon potential.
- Ai: Imaginary surface parameter of the Woods-Saxon potential.

Elastic Cross Section



$$U(r) = -V_0 f(x) - iW_v f(x') + V_c(r)$$

where:

$$f(x) = (1 + e^x)^{-1}$$

$$x = (r - R_r \cdot (A_t)^{1/3}) / A_r$$

$$x' = (r - R_i \cdot (A_t)^{1/3}) / A_i$$

Ap

At

zp

zt

Rr

Ar

Ri

Ai

Energy

Wv

V0

rc

zero Potential

POTENTIAL

SUBMIT

Figure 5.1: Window Interface

- Wv: Imaginary depth of the Wood-Saxon potential.
- V0: Real depth of the Wood-Saxon potential.
- Rc: is known as the Coulomb radius.
- $V_c(r)$: Coulomb potential.
- Energy: The centre of mass collision energy between the two nuclei.
- Channel radius: The point where the nuclear potential is neglected.

The interface shows the Scattering Cross Section of two nuclei colliding, and the Real part of the optical potential. The SUBMIT button sketches the scattering Cross Section and the bottom POTENTIAL sketches the real part of the potential.

Chapter 6

Elastic cross sections

6.1 ^{58}Ni Elastic cross section with an alpha particle

Here is exposed the results of the scattering cross section between an alpha particle and a nucleus of ^{58}Ni made in Python with center of mass collision energy of 139 MeV:

Six parameters in the Woods-Saxon optical potential

$$V(x) = -Vf(x) - iWf(x') + V_c(x) \quad (6.1)$$

Here $f(x) = (1 + e^x)^{-1}$, $x = (r - r_0A^{1/3})/a$ and $x' = (r - r'_0A^{1/3})/a'$. The values of the constants are shown in the Table (6.1).

Table 6.1: Parameters of the ^{58}Ni and α scattering [3]

$V(\text{MeV})$	$r(\text{fm})$	$a(\text{fm})$	$W(\text{MeV})$	$r'(\text{fm})$	$a'(\text{fm})$	$r_c(\text{fm})$
110.0	1.315	0.705	21.27	1.509	0.673	1.40

The Scattering Cross Section and the real part of the potential are shown in Figures (6.1) and (6.2).

From figure 6.1 is clear that the Nuclear potential can be neglected after 17 fm, therefore as a good approximation, we can take the channel radius a 17 fm in the Interface

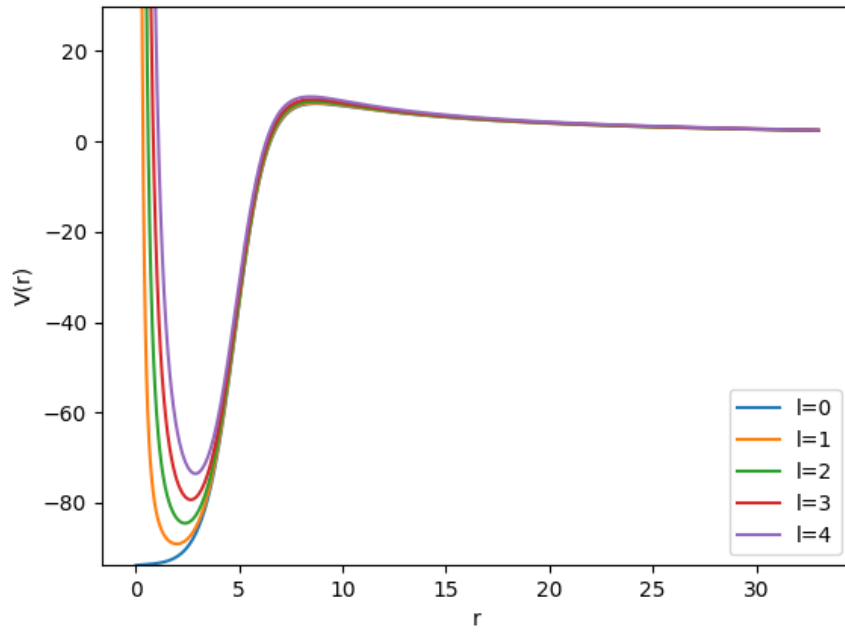


Figure 6.1: α - ^{58}Ni Real Optical Potential

Screen. The analytic result is compared with the experimental one in the paper "Scattering of 139-MeV Alpha Particles by ^{58}Ni and ^{208}Pb [2].

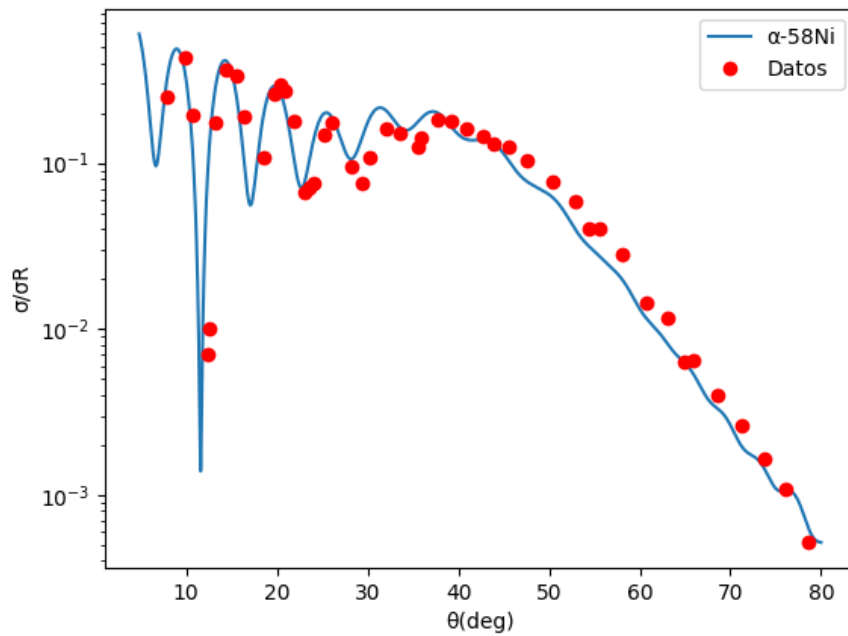


Figure 6.2: α - ^{58}Ni Scattering. Similar theoretical prediction behaviour can be found in [3]

6.2 ^{208}Pb Elastic cross section with an alpha particle

Here is exposed the results of the scattering cross section between an alpha particle and a nucleus of ^{208}Pb calculated computed in python with the collision energy in 139MeV:

Six parameters (6.2) in Woods-Saxon optical potential are taken in a potential of the form (6.1). Four values of V are taken in order to construct the Scattering Cross Section

Table 6.2: Parameters ^{208}Pb and α scattering. [3]

$V(\text{MeV})$	$r(\text{fm})$	$a(\text{fm})$	$W(\text{MeV})$	$r'(\text{fm})$	$a'(\text{fm})$	$r_c(\text{fm})$
110.0	1.315	0.705	21.27	1.509	0.673	1.40
155.0	1.282	0.677	23.26	1.478	0.733	1.40
200.0	1.261	0.657	24.50	1.462	0.767	1.40
245.0	1.245	0.647	25.89	1.448	0.788	1.40

The Scattering Cross section is shown in the Figure 6.4 for different Energies and for the real part of the optical potential shown in Figure 6.3.

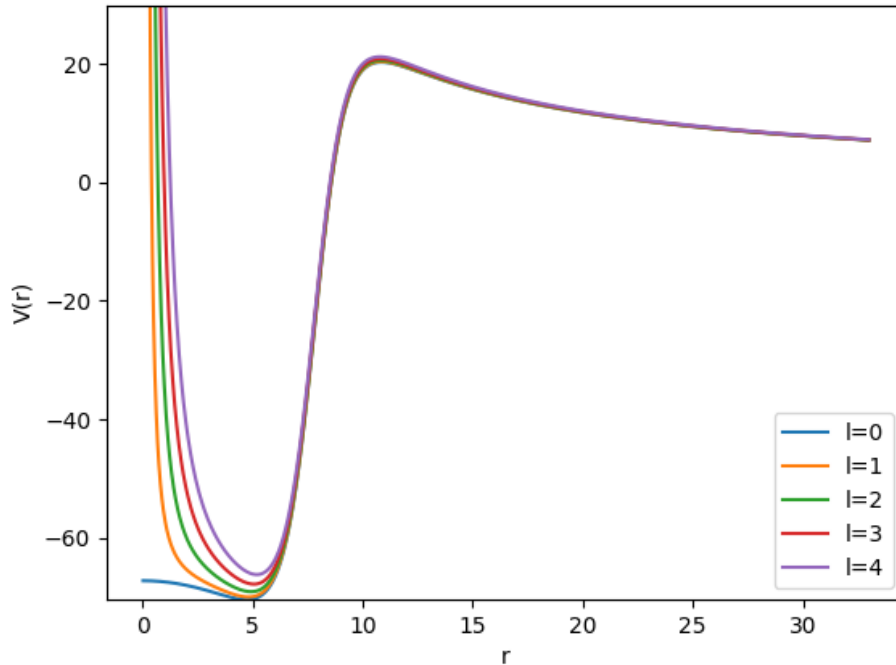


Figure 6.3: α and ^{208}Pb Real Optical Potential at 110 MeV

From figure 6.3 is clear that the Nuclear potential can be neglected from 17 fm, therefore 17 fm is put in the Interface Screen.

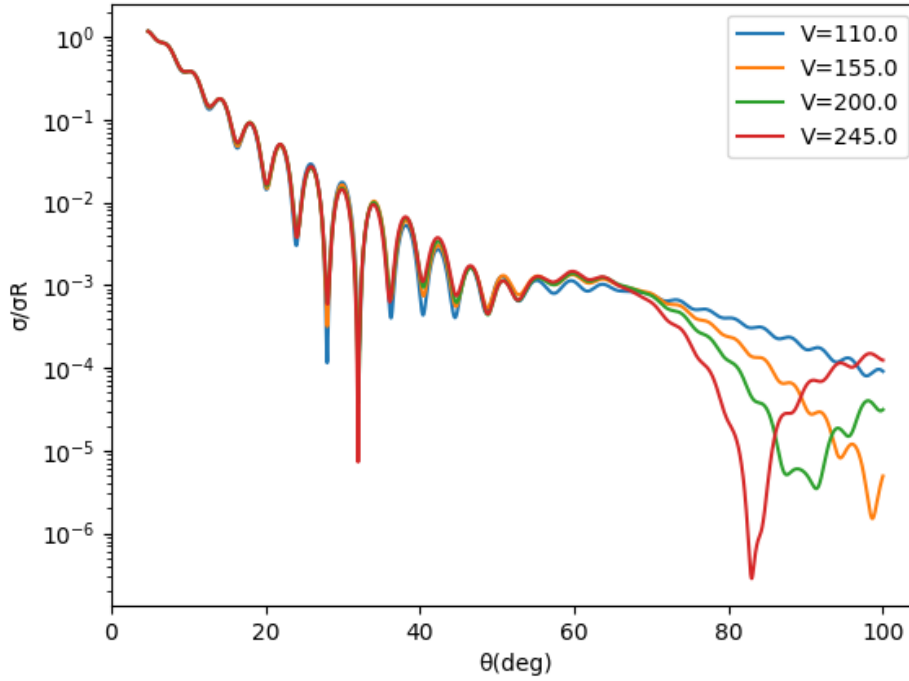


Figure 6.4: α - ^{208}Pb Scattering [3].

6.3 ^{90}Zr Elastic cross section with an ^{12}C nucleus

Here are exposed the results of the scattering cross section between an ^{12}C nucleus as a projectile and a ^{90}Zr nucleus as a target made in the Python language at 139 MeV mass centre collision energy .

Six parameters in Woods-Saxon optical potential of the form. They are given in Table 6.3

Table 6.3: Parameters ^{90}Zr and ^{12}C nucleus scattering. [5]

$V(\text{MeV})$	$r(\text{fm})$	$a(\text{fm})$	$W(\text{MeV})$	$r'(\text{fm})$	$a'(\text{fm})$	$r_c(\text{fm})$
175.0	3.051	0.900	156	5.100	0.593	1.40

The Scattering Cross section is shown in Figure 6.6 for different l values. The real part of the Optical Potential in figure 6.5.

From Figure 6.5 is clear that the Nuclear potential is depreciated at 35 fm, therefore 35 is put in the zero Potential space in the Interface Screen.

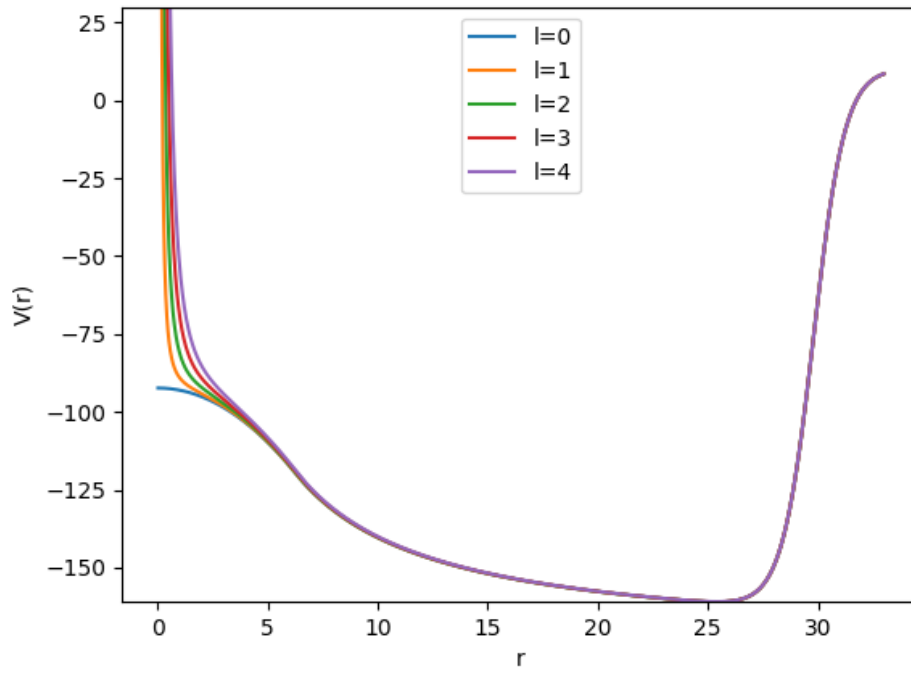


Figure 6.5: ^{12}C and ^{90}Zr Real Optical Potential at 175 MeV.

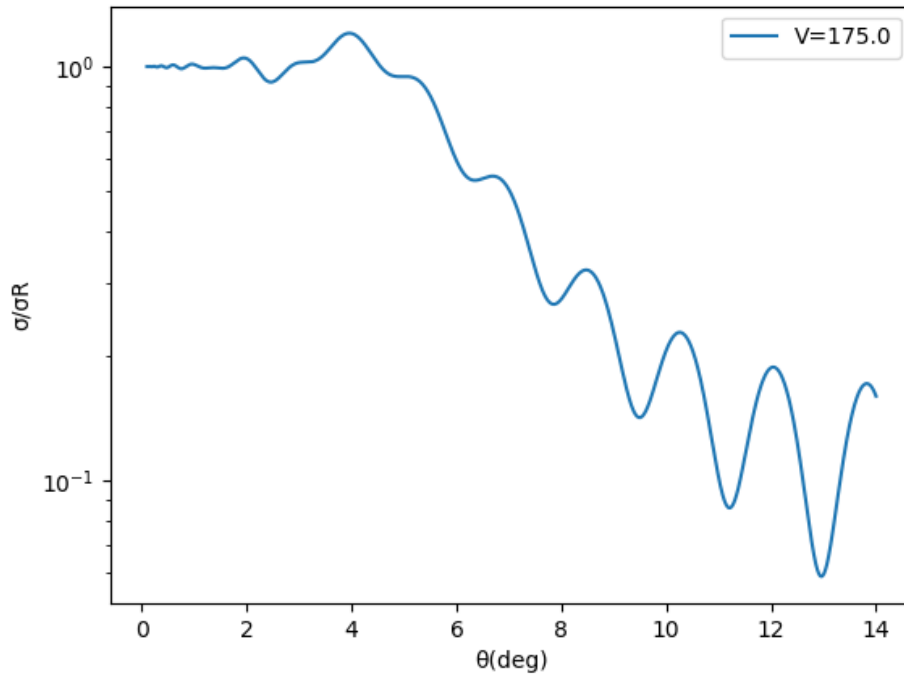


Figure 6.6: ^{12}C - ^{90}Zr Scattering [5].

Conclusion

An interactive platform based on the R matrix method, of the calculation of the Elastic cross section between two different nuclei are exposed in this thesis. Where the nuclear, central and Coulomb potential are taken in consideration, and the spin orbit interaction is neglected.

We show the elastic cross section for the scattering of ^{90}Zr with an ^{12}C nucleus, ^{208}Pb and an alpha particle, ^{58}Ni and an alpha particle. Their real part of the Optical potential is also displayed in this work.

The R-matrix method implemented in Python language is very useful and accurate to predict the elastic cross sections between two nuclei would behave.

Appendix A

The Code

```
import sys
from tkinter import *
import matplotlib.pyplot as plt
import mpmath as mp
from scipy.special import eval_legendre
from scipy.special import roots_legendre
from scipy.special import erf
from scipy.special import gamma, factorial
import numpy as np
import pandas as pd
```

#-----Parameters-----#

e2= 1.44

N=120

l=100

```

#-----Legendre roots and function-----#

def root(x):  #zeros legendre
    zero=roots_legendre(x)[0]
    xi=(zero+1)/2
    return xi

def L(N,x):  #eval legendre function
    b=eval_legendre (N,x)
    return b

def Legendre_functions(N):  #Regulazed Legendre functions evaluated in a
    M=[]
    zerosN=root(N)
    k=0
    for xi in zerosN:
        p=(-1)**(N+k+1)*(1/xi)*(a*xi*(1.-xi))**(1./2)*1/(a-a*xi)
        k=k+1
        M.append(p)
    MM=np.array(M)
    return MM

```

```

#-----Potential-----#

```

```

def V(x,l,Ap,At,zp,zt,h2mu,Rr,Ar,Ri,Ai,Wv,V0,rc,R0):
    aa=Vn(x,Ap,At,zp,zt,h2mu,Rr,Ar,Ri,Ai,Wv,V0,rc,R0)
    bb=Vc(x,Ap,At,zp,zt,h2mu,Rr,Ar,Ri,Ai,Wv,V0,rc,R0)
    cc=Vl(x,l,Ap,At,zp,zt,h2mu,Rr,Ar,Ri,Ai,Wv,V0,rc,R0)
    return aa+bb+cc

```

```

def Vl(x,l,Ap,At,zp,zt,h2mu,Rr,Ar,Ri,Ai,Wv,V0,rc,R0):
    cent=h2mu*l*(l+1)/(x*x)
    return cent

```

```

def Vn(x,Ap,At,zp,zt,h2mu,Rr,Ar,Ri,Ai,Wv,V0,rc,R0):
    xi=(x-Rr*At**(1/3))/Ar
    ff=(1+np.exp(xi))**(-1)
    VN=-V0*ff
    return VN

```

```

def Vc(x,Ap,At,zp,zt,h2mu,Rr,Ar,Ri,Ai,Wv,V0,rc,R0):
    if x>R0:
        a=zp*zt*e2/x
    else:
        a=zp*zt*(e2/(2*R0))*(3-np.power(x/R0,2))
    return a

```

```

def Up(x,l,Ap,At,zp,zt,h2mu,Rr,Ar,Ri,Ai,Wv,V0,rc,R0):
    R=V(x,l,Ap,At,zp,zt,h2mu,Rr,Ar,Ri,Ai,Wv,V0,rc,R0)+1j*Wl(x,Ri,Ai,Wv,At)

```

```

    return R

def f1(x, Ri, Ai, At):
    xi=(x-Ri*At**(1/3))/Ai
    ff=(1+np.exp(xi))**(-1)
    return ff

def derf1(x, Ri, A):
    dx=1e-9
    df=(f1(x+dx/2, Ri, A)-f1(x-dx/2, Ri, A))/dx
    return df

def Wl(x, Ri, Ai, Wv, At):
    WEr=Wv*f1(x, Ri, Ai, At)4*Wd*derf1(x, Ri, Ai)
    return -WEr

##-----Matrix-----#

def Cmatrix(zeros1, zeros2, a, N, l, E, Ap, At, zp, zt, h2mu, Rr, Ar, Ri, Ai, Wv, V0, rc,
R0):
##Matrix Elements
    A = np.matrix(np.zeros((N, N), dtype = np.complex))
    i=0
    for xi in zeros2:

```

```

j=0
for xj in zeros1:
    if xi!=xj:
        element=((-1)**(i+j)/(a**2*(xi*xj*(1-xi)*
        (1-xj))**(0.5)))*(N**2+N+1+(xi+xj-2*xi*xj)
        /(xi-xj)**2-1./(1-xi)-1./(1-xj))
        A[i,j]=element*h2mu
    elif xi==xj:
        A[i,j]=((4*N**2+4*N+3)*xi*(1-xi)-6*xi+1)/
        (3*a**2*xi**2*(1-xi)**2)*h2mu+
        Up(a*xi,l,Ap,At,zp,zt,h2mu,Rr,Ar,Ri,Ai,Wv,V0,rc,R0)-E
    j=j+1
i=i+1
return A

```

#-----R-Matrix-----#

```

def RE(l,E,Ap,At,zp,zt,h2mu,Rr,Ar,Ri,Ai,Wv,V0,rc,R0,a):
    zeros1=root(N)
    zeros2=root(N)
    Regulazed_Legendre_functions=Legendre_functions(N,a)
    C=Cmatrix(zeros1,zeros2,a,N,l,E,Ap,At,zp,zt,
    h2mu,Rr,Ar,Ri,Ai,Wv,V0,rc,R0) #C matrix
    Cinv = np.linalg.inv(C) #C inverse matrix
    RE1=np.dot(Cinv,Regulazed_Legendre_functions) #Dot matrix vector
    p=RE1.tolist()[0]
    RE2=np.array(p)
    REa=(h2mu/a)*Regulazed_Legendre_functions.dot(RE2) #R-matrix

```

```
    return REa
```

```
#-----Scattering matrix-----
```

```
def F(l, eta, x): #F Coulumb function
```

```
    Fc=mp.coulombf(l, eta, x)
```

```
    return Fc
```

```
def G(l, eta, x): #G Coulumb function
```

```
    Gc=mp.coulombg(l, eta, x)
```

```
    return Gc
```

```
def derivateH(m, l, eta, x): #Derivate of a function
```

```
    dx=1e-10
```

```
    h=(H(m, l, eta, x+dx/2)-H(m, l, eta, x-dx/2))/dx
```

```
    return h
```

```
def H(m, l, eta, x): #Coulumb Hankel (+,-) function
```

```
    if m==1:
```

```
        h=G(l, eta, x)+1j*F(l, eta, x)
```

```
    else:
```

```
        h=G(l, eta, x)-1j*F(l, eta, x)
```

```
    return h
```

```
def U(l, eta, x, E, Ap, At, zp, zt, h2mu
```

```
, Rr, Ar, Ri, Ai, Wv, V0, rc, R0, a): #Scattering matrix
```

```
    ul=(H(0, l, eta, x)-x*(RE(l, E, Ap, At, zp, zt, h2mu, Rr, Ar, Ri, Ai, Wv, V0,
```

```
    rc, R0))*derivateH(0, l, eta, x))/(H(1, l, eta, x)-x*(RE(l, E, Ap, At, zp,
```

```

        zt , h2mu , Rr , Ar , Ri , Ai , Wv , V0 , rc , R0)) * derivateH (1 , l , eta , x))

    return  ul


def ScatterMatrix (l , i , Ap , At , zp , zt , h2mu , Rr , Ar , Ri , Ai , Wv , V0 , rc , R0 , a) :
    k=(i /h2mu)**(1./2)
    eta=zp*zt*e2/(2*k*h2mu)
    scatterM=U(l , eta , k*a , i , Ap , At , zp , zt , h2mu , Rr , Ar , Ri , Ai , Wv , V0 , rc , R0)
    return  scatterM


#-----Sigma-----#
def sigma0 (eta) :
    h=gamma(1+1j*eta)
    tt=np . angle (h)
    return  tt


def signal (L , eta) :
    h=gamma(1+L+1j*eta)
    tt=np . angle (h)
    return  tt


#-----Elastic Cross Section-----#

def f (i , l , x , Ap , At , zp , zt , h2mu , Rr , Ar , Ri , Ai , Wv , V0 , rc , R0 , a) :
    k=(i /h2mu)**(1./2)
    eta=zp*zt*e2/(2*k*h2mu)
    M=[]

```

```

fthetha2=-(eta/(2*k*np.sin(x/2)*np.sin(x/2)))*np.exp(2*j*sigma0(eta))*
(np.exp(-1*j*eta*np.log((np.sin(x/2)*np.sin(x/2)))))
L=0
while L<=l:
    print(L)
    b=eval_legendre(L,np.cos(x))
    b=np.array(b)
    MatrixS=ScatterMatrix(L,i,Ap,At,zp,zt,h2mu,Rr,Ar,Ri,Ai,Wv,V0,
rc,R0)
    fthetha1=complex((1/(2*1*j*k))*(2*L+1)*np.exp(2*1*j*sigma1(L,eta))
*(MatrixS-1))
    fthetha1=fthetha1*b
    ffthetha1=np.array(fthetha1)
    M.append(ffthetha1)
    L=L+1
Ma=M[0]
c=0
for i in range(len(M)-1):      #Plus componentes of Wa
    c=c+1
    Ma=Ma+M[c]
return (Ma,fthetha2)

```

#-----Plot-----#

```

def salida (Ap,At,zp,zt,h2mu,Rr,Ar,Ri,Ai,Wv,V0,rc,R0,a,Energy):
    grados=np.arange(0.1,14,0.001)
    radianes=np.radians(grados)

```

```

FTH=f(Energy,l,radianes,Ap,At,zp,zt,h2mu,Rr,Ar,Ri,Ai,Wv,V0,rc,R0,a)
FTHa,FTHb=FTH[0],FTH[1]
ElasticCrosssection=(np.abs((FTHa+FTHb)/FTHb)**2)
plt.semilogy(grados,ElasticCrosssection,label="V={}".format(V0))
plt.title("Elastic_Cross_Section")
plt.ylabel("    / R ")
plt.xlabel("    (deg)")
plt.legend()
plt.savefig('foo.png')
plt.show()

def salida2(Ap,At,zp,zt,h2mu,Rr,Ar,Ri,Ai,Wv,V0,rc,R0):
    radio=np.arange(0.01,33,0.01)
    DEFI=[]
    for i in range(5):
        pop=[]
        for j in radio:
            Points=V(j,i,Ap,At,zp,zt,h2mu,Rr,Ar,Ri,Ai,Wv,V0,rc,R0)
            pop.append(Points)
        DEFI.append(pop)
    plt.title("Real_Potential")
    plt.ylabel("V(r)")
    plt.xlabel("r")
    top=30
    bottom=min(DEFI[0])
    for graph in range(5):

```

```

plt.ylim(bottom, top)

plt.title("Real_Potential")

plt.plot(radius,DEFI[graph],label="l={}".format(graph))

plt.legend()

plt.show()

```

###key down function

```

def click():
    Ap=textentry1.get() #Collect the text from the text entry box
    At=textentry2.get()
    zp=textentry3.get()
    zt=textentry4.get()
    Ap,At,zp,zt=float(Ap),float(At),float(zp),float(zt)
    h2mu=20.736*(Ap+At)/(Ap*At)
    Rr=textentry5.get()
    Ar=textentry6.get()
    Ri=textentry7.get()
    Ai=textentry8.get()
    Wv=textentry9.get()
    V0=textentry10.get()
    rc=textentry11.get()
    Rr,Ar,Ri,Ai,Wv,V0,rc=float(Rr),float(Ar),float(Ri),float(Ai)
    ,float(Wv),float(V0),float(rc)
    R0=rc*At**(1./3)
    salida(Ap,At,zp,zt,h2mu,Rr,Ar,Ri,Ai,Wv,V0,rc,R0,a,Energy)

def POTENTIAL():

```

```
Ap=textentry1.get()
At=textentry2.get()
zp=textentry3.get()
zt=textentry4.get()
Ap,At,zp,zt=float(Ap),float(At),float(zp),float(zt)
h2mu=20.736*(Ap+At)/(Ap*At)
Rr=textentry5.get()
Ar=textentry6.get()
Ri=textentry7.get()
Ai=textentry8.get()
Wv=textentry9.get()
V0=textentry10.get()
rc=textentry11.get()
Rr,Ar,Ri,Ai,Wv,V0,rc=float(Rr),float(Ar),float(Ri),
float(Ai),float(Wv),float(V0),float(rc)
R0=rc*At**(1./3)
salida2(Ap,At,zp,zt,h2mu,Rr,Ar,Ri,Ai,Wv,V0,rc,R0)
```

```
##### main:
```

```
window=Tk()
window.title("Elastic_Cross_Section_")
window.configure(background="black")
```

```
#### My Photo
```



```
photo1=PhotoImage( file="Quatum.png")
photo2=PhotoImage( file="potential.png")
Label(window,image=photo1,bg="black").grid(row=0, column=1, sticky=W)
Label(window,image=photo2,bg="black").grid(row=0, column=3, sticky=W)

#### create label
Label(window, text="Ap", bg="black", fg="white", font="none_12_bold")
.grid(row=1, column=0, sticky=W)
Label(window, text="At", bg="black", fg="white", font="none_12_bold")
.grid(row=2, column=0, sticky=W)
Label(window, text="zp", bg="black", fg="white", font="none_12_bold")
.grid(row=3, column=0, sticky=W)
Label(window, text="zt", bg="black", fg="white", font="none_12_bold")
.grid(row=4, column=0, sticky=W)

Label(window, text="Rr", bg="black", fg="white", font="none_12_bold")
.grid(row=6, column=0, sticky=W)
Label(window, text="Ar", bg="black", fg="white", font="none_12_bold")
.grid(row=7, column=0, sticky=W)
Label(window, text="Ri", bg="black", fg="white", font="none_12_bold")
.grid(row=8, column=0, sticky=W)
Label(window, text="Ai", bg="black", fg="white", font="none_12_bold")
.grid(row=9, column=0, sticky=W)
Label(window, text="Wv", bg="black", fg="white", font="none_12_bold")
.grid(row=6, column=2, sticky=W)
Label(window, text="V0", bg="black", fg="white", font="none_12_bold")
.grid(row=7, column=2, sticky=W)
```

```

Label(window, text="rc", bg="black", fg="white", font="none_12_bold")
.grid(row=8, column=2, sticky=W)
Label(window, text="zero_Potential", bg="black", fg="white", font="none
12_bold").grid(row=9, column=2, sticky=W)
Label(window, text="Energy", bg="black", fg="white", font="none_12_bold")
.grid(row=5, column=2, sticky=W)

```

```

##### create a text entry box

```

```

textentry1=Entry(window, width=7, bg="white")
textentry1.grid(row=1, column=1, sticky=W)           #Ap
textentry2=Entry(window, width=7, bg="white")
textentry2.grid(row=2, column=1, sticky=W)           #At
textentry3=Entry(window, width=7, bg="white")
textentry3.grid(row=3, column=1, sticky=W)           #zp
textentry4=Entry(window, width=7, bg="white")
textentry4.grid(row=4, column=1, sticky=W)           #zt
textentry5=Entry(window, width=7, bg="white")
textentry5.grid(row=6, column=1, sticky=W)           #Rr
textentry6=Entry(window, width=7, bg="white")
textentry6.grid(row=7, column=1, sticky=W)           #Ar
textentry7=Entry(window, width=7, bg="white")
textentry7.grid(row=8, column=1, sticky=W)           #Ri
textentry8=Entry(window, width=7, bg="white")
textentry8.grid(row=9, column=1, sticky=W)           #Ai
textentry9=Entry(window, width=7, bg="white")
textentry9.grid(row=6, column=3, sticky=W)           #Wv

```

```
textentry10=Entry(window, width=7, bg="white")
textentry10.grid(row=7, column=3, sticky=W)           #V0
textentry11=Entry(window, width=7, bg="white")
textentry11.grid(row=8, column=3, sticky=W)           #rc
textentry12=Entry(window, width=7, bg="white")
textentry12.grid(row=9, column=3, sticky=W)           #Zero potential
textentry13=Entry(window, width=7, bg="white")
textentry13.grid(row=5, column=3, sticky=W)           #Energy

##add a submit button
Button(window, text="SUBMIT", width=12, command=click)
.grid(row=10, column=3, sticky=W)
Button(window, text="POTENTIAL", width=12, command=POTENTIAL)
.grid(row=10, column=1, sticky=W)

window.mainloop()
```

Bibliography

- [1] L. F. Canto and M. S. Hussein. Scattering theory of molecules, atoms and nuclei. 2013.
- [2] F. L. Claude Cohen tannoudji, Bernard Diu. Quantum mechanics volumen 2. 2013.
- [3] H. G. P. P. R. D.A. Goldberg, S. M. Smith and N. Wall. Scattering of 139-MeV alpha particles by ^{58}Ni and ^{208}Pb . pages 1–4, 1972.
- [4] P. Descouvemont and D. Baye. The R-matrix theory. 2010.
- [5] A. R. A. L.Jarczyk, B.kamys and B. Styczen. Scattering of 344.5MeV ^{12}C ions on ^{11}B , ^{12}C , ^{27}Al , ^{58}Ni , ^{90}Zr , and ^{197}Au nuclei. 1990.
- [6] P.FROBRICH and R.LLIPERHEIDE. Theory of nuclear reactions. 1996.
- [7] N. Zettili. Quantum mechanics concepts and applications. 2009.