Chaotic Scattering for a Sliding Mass in a Complex Topography

Avi Vajpeyi

CS 200

May 2, 2016

Abstract

The Gaspard-Rice scattering model has been instrumental in framing the understanding of a wide range of chaotic scattering behavior problems in physical situations [10]. This paper draws inspiration from the model to simulate chaotic scattering of a point particle that is released with four hills in its path. The simulation resembles most classical scattering systems with one impact parameter that causes the particle to scatter. The trajectories of the impact parameters are generated by numerically integrating differential equations of motion using various algorithms such as the Euler-Cromer and Runge-Kutta methods. While traversing across the hills, certain impact parameters lead to the particle getting trapped for a period inside the system before exiting at an angle measured from the horizontal axis. The angle of exit from the system is compared to the impact parameter, and is found to display chaotic nature at certain regions. Additionally, it is found that the correlation between the parameters demonstrates fractal behavior.

List of Figures

1 Schematic of Scattering Problem 6
2 Schematic of Scattering with Potential Hills 7
3 Plots of the Potential 8
4 Midpoint Rule for Integration 10
5 Code for Euler-Cromer Integration 13
6 Screen Shot of Application 1 16
7 Screen Shot of Application 2 17
8 Screen Shot of the Application's Data Archive Window 18
9 Diagram for the Scattering Angle 18
10 Zoom in of an Energy Plot for the Particle 19
11 An Energy Plot for the Particle 20
12 Plots of the Scattering Angle VS Impact Parameter 21
13 Plots of the Scattering Angle VS Velocity 22
14 Plots of the Scattering Angle VS Impact Parameter for Different Velocities 23
15 Screen Shot of the Interface 27
16 Screen Shot of the About Box 27
17 Screen Shot of the Data Entry Window 28

Contents

1 Introduction **4**
1.1 What is Chaotic Scattering? 4
1.2 Motivations 4
1.3 Simple Example 5

2 Physics Model **7**
2.1 Description of the Model 7
2.2 Physical Theory 8

3 Numerical Integration and Computation Accuracy **9**
3.1 Integration in Simulations 9
3.2 Varying the Size of Time Steps 10
3.3 Numerical Integration Techniques 11
3.3.1 What is Numerical Integration? 11

3.3.2 Euler-Cromer 12
3.4 Four Stage Runge-Kutta 14
3.5 Integration Accuracy Checks 14

4 Software 15
4.1 Programming Language and APIs 15
4.2 Interface Design and Interacting with the Program 15
4.3 How the Scattering Angle is Calculated 18

5 Results and Analysis 19
5.1 Integration Accuracy Results 19
5.2 Chaotic, Non Chaotic and Fractal Regions in Plots 20
5.3 Changing Initial Velocity for a Fixed Impact Parameter 22
5.4 Effect of Initial Velocity on the Scattering Angle 23

6 Conclusions and Future Work 23

7 Appendix 25
7.1 Step Code and Implementation of RK4 25
7.2 The XML Interface Builder 27

1 Introduction

Nature is extraordinarily intricate and unpredictable. In 1961, Edward Lorenz, a meteorologist, created a computer simulation to model weather [7]. He rounded an input parameter to three decimal places instead of the conventional six and was surprised to find the impact that this tiny alteration had on his long term forecasting abilities [7]. This highlighted the fact that small uncertainties can yield drastically different outcomes. Hence predicting the patterns of complicated systems such as the migratory pattern of birds, pandemics, or even patterns of the human heart, is nearly impossible, as these predictions require very high precision measurements. This gives rise to chaos theory, a discipline of mathematics that studies the behavior and conditions of sensitive systems [8]. This paper examines chaotic scattering within chaos theory, a topic with applications in a wide range of fields including nuclear physics, celestial mechanics, and fluid mechanics. This paper examines the subject of scattering using a simulated study.

1.1 What is Chaotic Scattering?

Chaotic scattering, a branch of chaos theory, is a topic in nonlinear physics. It deals with the problem of finding and studying the chaotic relationship between input variables of trajectories and their resulting outputs [8]. These systems examine the impact of confined boundaries on trajectories before they escape to the regular dynamics of infinity. The bounded region is what causes the trajectories to change, and what is studied with the input parameters to understand the nature of the system.

1.2 Motivations

Chaotic scattering is a topic of vital importance, as it gives us better techniques to predict outcomes in various fields, such as the trajectory of a comet through a solar system [8]. This paper provides a simple understanding of chaotic scattering as discussed in diverse topics of study, and introduces a simulation of a particle traversing through a region with four hills. As this program is unique from previous chaotic scattering simulations, we may gain new insights about how chaotic scattering works by studying the results we obtain. Additionally, the fractals that form in the graphs of the data may improve our understanding behind the formation of fractals in chaotic scattering systems.

1.3 Simple Example

We examine the dynamics of chaotic scattering with the help of a particle traveling through a region of potential $V(x)$. The potential is defined such that

$$V(x) = \begin{cases} 0, & \text{if } x \text{ is outside the region} \\ 0 < V < 1, & \text{otherwise} \end{cases} \tag{1}$$

where the size of the region in this example is a closed shape of area A. Figure 1 shows this shape, and the particle approaching the shape, in a trajectory parallel to the x axis. As the potential is zero outside the shape, the particle is unaffected as it moves along its initial path. However, once the particle enters the shape, it experiences different levels of force during its transit. The alterations in the force are due to the changes in potential, which varies from 0 to 1 in different regions of the shape. This changing force experienced by the particle causes its path to deviate.

The particle spends some time, T, in this region before exiting at an angle θ_s, as seen in the figure. Altering the point of entry of the particle into the region causes the value of T and θ_S to change. Sometimes, even small changes in the point of entry can result in drastically different values for both T and θ_s. These are the points where the system exhibits chaotic properties. Comparing the output parameters of T and θ_s with the different initial points of entry, we find graphs showing chaotic nature in some regions, and non chaotic in others. Chaotic nature is described as regions in which points vary rapidly with small changes, while non chaotic nature is described as regions where change is gradual and systematic [8]. Note that this example is borrowed from one provided in [10], but written for an undergraduate audience.

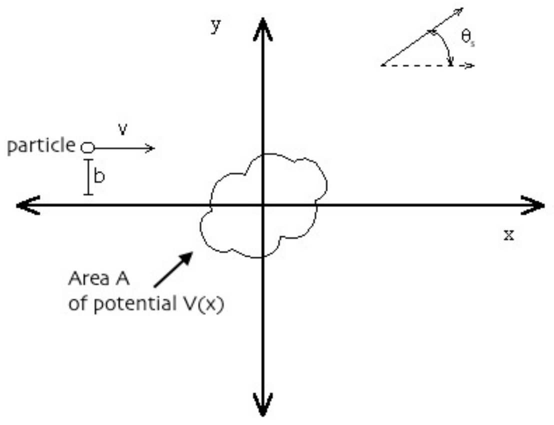


Figure 1: Schematic illustration of a scattering problem in two dimensions taken from [8].

This paper examines the scattering problem in two dimensions with four Gaussian potential hills instead of the one arbitrary potential bounded by a region. Additionally, the Gaussian potentials are that of gravity, hence the model simulates a particle rolling through a plane with four identical hills placed around one point. The model is discussed in further detail in Section 2.

2 Physics Model

2.1 Description of the Model

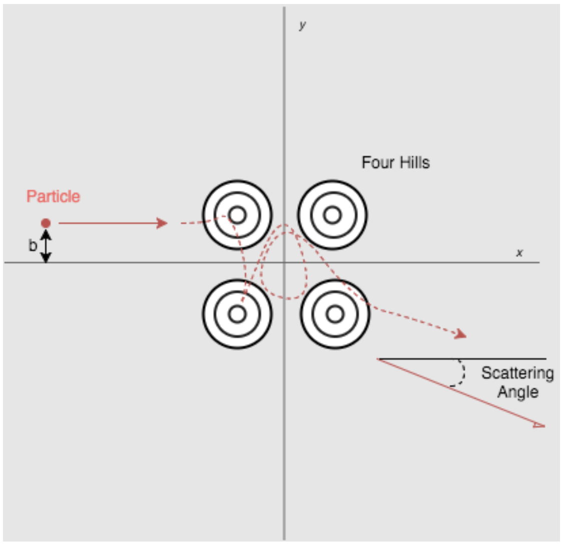


Figure 2: Schematic illustration of the scattering problem with four potential hills in the particle's path. b is the initial y position of the particle, and is known as the impact parameter. The scattering angle is made by the particle's trajectory with the horizontal axis. The dashed line shows the trajectory of the particle.

Section 1.3 discussed a scattering model with one potential. Using that, consider a particle moving towards a potential $V(x, y)$ without friction. This region's potential is different from the one mentioned in the previous section, as it is comprises four hills that have peaks at $(x, y) = (\pm 1, \pm 1)$. This system is shown in Figure 2.

As seen in Figure 2, the particle initially starts at a fixed horizontal distance, sufficiently far from the potential, so that it is unaffected by the dynamics due to the potential. Its vertical distance from the x axis is denoted as b, and is called the 'impact parameter.' This impact parameter is what defines the point of entry of the particle into the region with the potential, known as the scattering region.

After interacting with the scattering region, the particle exits the area, its trajectory forming an angle with the x axis. This angle is known as the scattering angle, and is the same as the one mentioned in Section 1.3.

It is expected that if the energy E of the particle is greater than the maximum potential energy at the hill peaks E_m, then the particle should be able to pass through the hilly region without much alteration in its trajectory, and the scattering is non-chaotic [10]. While decreasing E below the threshold energy E_m and altering b in small increments, we expect the scattering angle to vary unpredictably, demonstrating the chaotic nature of the system [10]. The functional dependence of the scattering angle and the impact parameter is interesting because it is different for the regions where the scattering is chaotic and where the scattering is non chaotic. This is discussed further in Section 5.2.

2.2 Physical Theory

Here, we will discuss the physics behind the dynamics of the system. We start by studying the the equation for the potential mapped by our four hill system,

$$\vec{V}(x, y) = x^2 y^2 \; e^{-x^2-y^2}. \tag{2}$$

This potential, as seen in Figure 3, is what causes the trajectory of our particle to deviate as it enters the scattering region. The reason the deviation actually occurs is because the potential imposes a force on the particle, which causes the particle to accelerate to the region where the potential is the lowest.

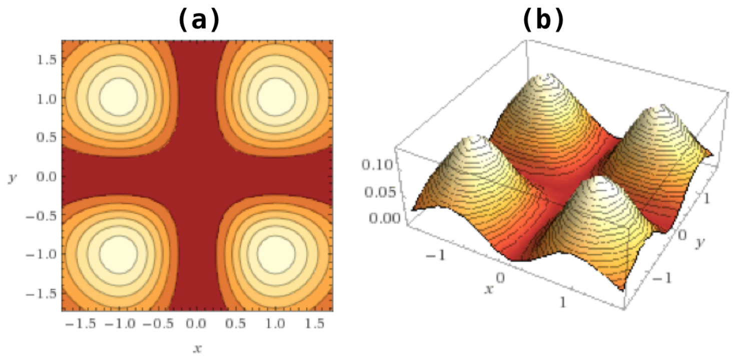


Figure 3: A contour plot (a), and a three-dimensional plot (b) created on Wolfram Alpha for the potential hills of equation 2. The potential has four maxima, at coordinates (1,1) , (-1,-1) , (-1,1) , (1,-1). At these maxima the potential energy has a value of $V(x, y) = 1/e^2 = E_{max}$.

We know from the laws of energy that a force is equal to the negative gradient of a potential, or $\vec{F} = -\nabla\vec{V}(r)$ [9]. Hence, we can calculate the force due to the potential given by equation 2 by finding its negative derivative :

$$\begin{aligned}\vec{F} &= -\nabla\vec{V}(r)\\ &= -\{2e^{-x^2-y^2}x(-1+x^2)y^2,\ 2e^{-x^2-y^2}x^2y(-1+y^2)\}\end{aligned} \tag{3}$$

According to Newton's laws, we have $\vec{F} = m\vec{a}$, hence we can state that for a unit mass $m = 1$ unit, $\vec{F} = \vec{a}$ [9]. In our system, we consider the particle as a unit mass, so the acceleration of the particle in the scattering region is

$$\vec{a} = -\{2e^{-x^2-y^2}x(-1+x^2)y^2,\ 2e^{-x^2-y^2}x^2y(-1+y^2)\}. \tag{4}$$

With this formula, we can use different numerical integration techniques to calculate the evolution of the position of our particle over time as it traverses through the scattering region. Note that the idea of using a unit mass to make calculations simpler was taken from a program written to simulate two celestial bodies in [9].

3 Numerical Integration and Computation Accuracy

3.1 Integration in Simulations

There are various ways to calculate the area underneath the curve of a function, and thereby estimate the value of a definite integral. One way is the by the 'Midpoint Rule.'

Consider the following integral $\int_a^b f(x)dx$, where $[a, b]$ is the interval of integration, $f(x)$ is the function being integrated, and dx is a small width that is the length of each subinterval. We calculate midpoints between the points separated by the distance of dx, and then evaluate the value of $f(x)$ at these midpoints, as seen in Fig 4.

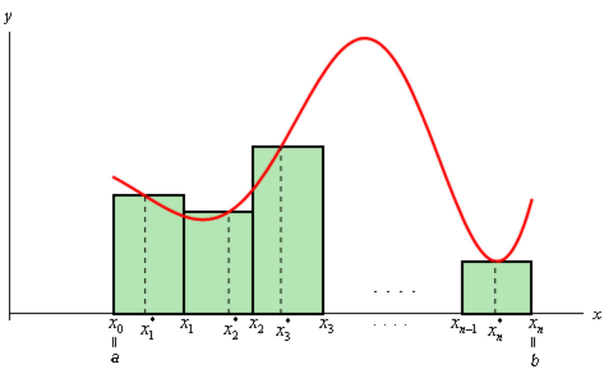


Figure 4: An integral being evaluated with the midpoint rule. The distance between intervals $[x_1, x_2], [x_2, x_3]...$ is dx, and the midpoints of these intervals are $x_1^*, x_2^*...$ These midpoints are where $f(x)$ is evaluated.

As seen in the graph, we can break the area underneath the curve of $f(x)$ into several rectangles. Each rectangle has a width of dx, and a height equal to the value of the function, $f(x)$, evaluated at the midpoints of the rectangles. Summing the areas of each of these rectangles, we can estimate the value of the definite integral.

This process of integration is very similar to what is used in the simulation. However, instead of using the 'Midpoint Rule,' the simulation uses the Euler-Cromer, and Runge-Kutta algorithms, which use different methods for obtaining the midpoint.

Additionally, the subintervals for integration in this simulation are dt, known as the time step. This is the size by which the time variable t is increased before evaluating the value of the function at time t.

3.2 Varying the Size of Time Steps

A brute force method to increase the accuracy of computations in most physical experiments is simply to take data for a larger number of points. For computational experiments such as this one, the number of data points can be increased by decreasing the size of dt, the time step used for integration, while still integrating over the same period of time [1]. However, this technique makes computations much slower, because more calculations are required [1].

Additionally, many computational physicists use variable time steps to increase integration accuracy and speed. In this simulation a variable time step is set up so that the time step becomes smaller as the speed of the particle increases. By doing this, the simulation maps the path of a particle to a more accurate degree as it travels at higher speeds. Speed increases are more likely to occur as the particle takes sharp turns. The variable time step also allows the computations to be fast, since in regions where the particle's speed is relatively constant, the time step becomes larger. This speeds up the computation while still accurately recording data for the position of the particle.

Another method to increase accuracy is finding and using better suited and more efficient algorithms for the computations. In the following section, we will discuss some numerical integration algorithms that we use for the chaotic scattering simulation.

3.3 Numerical Integration Techniques

Here we discuss the various numerical integration techniques that we can use to calculate the evolution of the position of our particle with time as it traverses through the scattering region as discussed earlier. Before we begin talking about numerical integration techniques, we first provide a background on these techniques.

3.3.1 What is Numerical Integration?

Newton and Libniz discovered calculus, and demonstrated to the world a vast new set of mathematical possibilities [4]. However, mathematicians discovered some ordinary differential equations that could not be solved using techniques provided by Newton and Libniz.

An ordinary differential equation (an ODE) is a function that expresses the rate of change of a function with respect to changing variables [4]. For example, a common type of ODE that we encounter can be expressed as $f(x, y) = dx/dy$. To solve such differential equations, mathematicians try to obtain the original function, which creates a specific rate of change with certain initial values for the variables.

Although representing ordinary differential equations in this manner makes solving them seem trivial, there are many ODEs that cannot be solved with algebra and integration. In order to solve these types of ODEs, mathematicians such as Euler, Runge and Kutta began looking at the use of small estimations to approximate the path that a function followed [4]. Hence, these methods provide us with approximations for solutions of differential equations. This process, known as numerical integration, immediately proved to be a invaluable tool for mathematicians. With numerical integration, we can approximate many more ODEs and thereby have the ability to measure the rate of change of various functions, such as temperature changes, population growth, and even velocity [4].

Because of the usefulness of numerical integration, many mathematicians have worked on forming different techniques for it. Additionally, since the behavior of each ODE varies, different numerical integration methods are required to undertake different problems. This consequently gives rise to many mathematicians creating their own numerical integration techniques, each having their own costs and benefits [4].

We now discuss two popular numerical integration techniques used to calculate quantities such as the trajectories of astronomical bodies. These are the techniques that have been used in our simulation to calculate the trajectory of the particle.

3.3.2 Euler-Cromer

In the late 18th century, Leonhard Euler, the man who discovered the famous relation, $e^{\imath\pi} = -1$, was also one of the mathematicians of his time to recognize the beauty of numerical integration [3]. With his studies, he produced a simple numerical integration technique. The numerical integration method works by first calculating the slope at a given point, and then recalculating it after stepping over a fixed interval [4]. These can be easily calculated if an initial starting point is given. This process is then repeated at a new point. Plotting the slopes calculated at the different points helps us approximate the graph of the original function from which we obtained the ODE.

In the case of the chaotic scattering problem, we have a differential equation defining the acceleration of the particle as the second derivative of the position of the particle as seen below [1].

$$a(t) = \frac{d^2}{dx^2}x(t)$$

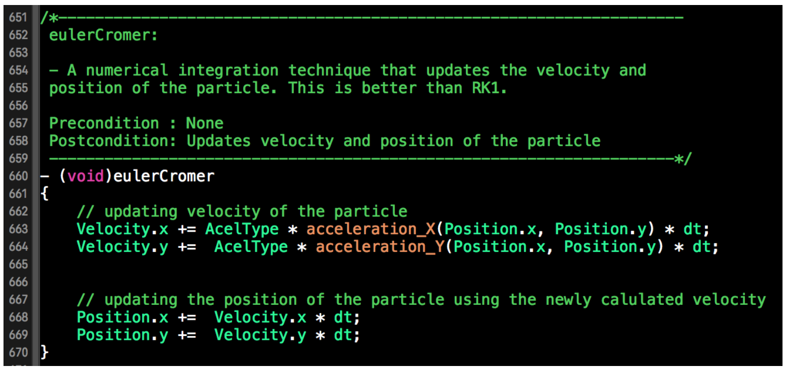
This equation can be broken into two differential equations, relating position $x(t)$ to its derivative, velocity $v(t)$ and relating velocity to its derivative, acceleration $a(t)$:

$$x_{n+1}(t) = x_n(t) + v_n(t)\triangle t$$
$$v_{n+1}(t) = v_n(t) + a_n(t)\triangle t$$

These coupled differential equations can then be solved simultaneously using Euler's Method [1]. This process of calculating the position of a moving particle is very similar to how astrophysicists plot the trajectories of planets, and is what inspired this work [1].

Cromer, a mathematician who was implementing this method, altered the process by first updating the velocity, and then the position (using the newly calculated velocity) [3]. This means that the position is determined based on a later velocity, and hence is more accurate than the method Euler created. This modified method is known as the Euler-Cromer method.

Fig 5 contains the code for the implementation of the Euler-Cromer algorithm.



```
/*-----------------------------------------------------------------------
 eulerCromer:

 - A numerical integration technique that updates the velocity and
 position of the particle. This is better than RK1.

 Precondition : None
 Postcondition: Updates velocity and position of the particle
 ----------------------------------------------------------------------*/
- (void)eulerCromer
{
    // updating velocity of the particle
    Velocity.x += AcelType * acceleration_X(Position.x, Position.y) * dt;
    Velocity.y +=  AcelType * acceleration_Y(Position.x, Position.y) * dt;


    // updating the position of the particle using the newly calulated velocity
    Position.x +=  Velocity.x * dt;
    Position.y +=  Velocity.y * dt;
}
```

Figure 5: Implementation of the Euler-Cromer algorithm in the simulation. As seen, we are updating the velocity, position and then the time.

3.4 Four Stage Runge-Kutta

A century after Euler created his numerical integration algorithm, Carl David Tolmé Runge and Martin Wilhelm Kutta created a new set of numerical integration schemes known as the Runge-Kutta methods [3]. Like the Euler-Cromer algorithm, sets of numerical integration techniques can be applied to a large set of problems. These methods calculate each point by first calculating trial steps at the mid point of the time step [2]. These midpoints help remove lower-order error terms, and make the calculation of each point more accurate, as the point is calculated by taking the average of several midpoints, each with different weights [2].

As you increase the order of the Runge-Kutta method, you get more mid points. Hence the higher the order of the Runge-Kutta method, the more the number of mid points that we use to calculate the value of one data point [2]. This makes the approximation more accurate; however, it also makes the time taken for the approximation longer. Additionally, since the number of mid points that are present between a time step increases as we increase the order of the Runge-Kutta method, the higher order algorithms can be accurate even with a large time step [2].

The most popular Runge-Kutta method is the forth order method, known RK4. The RK4 method is preferred since it has a higher degree of accuracy than the lower order algorithms, but at the same time does not take as long as other higher order algorithms [2]. The step code for the Runge-Kutta 4 algorithm, along with the implementation of the code for this algorithm can be found in Appendix 7.1.

3.5 Integration Accuracy Checks

As this is a system in which there are no external forces, energy is contained in the system. Hence, this provides us with a convenient tool to study the accuracy of integration: the energy of the particle. If the energy of the particle is constant, then we know that the integration is accurate. We calculate this energy by summing the potential energy and the kinetic energy of the particle at a given point. The total energy is given by

$$Energy = \frac{1}{2}\, mass \cdot velocity^2(x, y) + potential(x, y),$$

where *potential*(x, y) is the potential energy of the particle, as described in equation 2.

4 Software

Here, we discuss the software used to implement the model of the simulation.

4.1 Programming Language and APIs

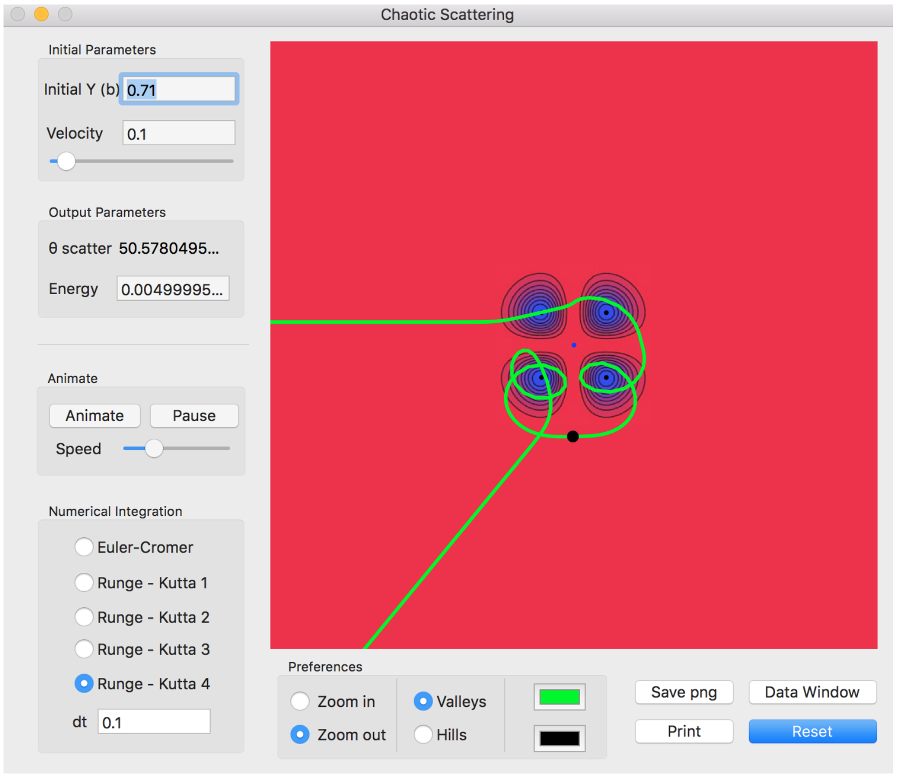
We implement the model discussed in section 2 in a Coca application for OS X. The program is in Objective C, and we rely on the various tools available via the Coca API to run it. Xcode is the IDE we use to construct the simulation, as it provides a straightforward interface builder, which permits users to easily connect buttons, text fields, and other items in the interface to back-end functions that make them work.

4.2 Interface Design and Interacting with the Program

The application consists of a window with a view of a two-dimensional plane, displaying the four hills, and the particle at a distance from the hills, as seen in Figure 6. The trajectory of the particle can be adjusted by clicking on the view, and dragging the initial path of the particle along the y axis. Alternately, the starting position of the particle can be set within the 'Initial Y (b)' text field. The 'b' is the symbol for the impact parameter used in comparable simulations for chaotic scattering. As discussed earlier, it is this variable that causes chaotic scattering at certain values.

The particle's velocity, another input parameter, can be changed within the text field marked 'Velocity,' or with the horizontal slider underneath it. Changing this will in turn change the energy of the particle. Increasing the energy, or the initial velocity of the particle while keeping the 'b' constant, can demonstrate how having a different velocity can cause different trajectories to emerge. This is discussed in Section 5.3.

The user may also change the numerical integration technique used to calculate the path of the particle by choosing between the radio buttons for the Euler-Cromer



Initial Y (b) 0.71

Velocity 0.1

θ scatter 50.5780495...

Energy 0.00499995...

[✓ Runge - Kutta 4]

dt 0.1

[✓ Zoom out]

[✓ Valleys]

🟩

■

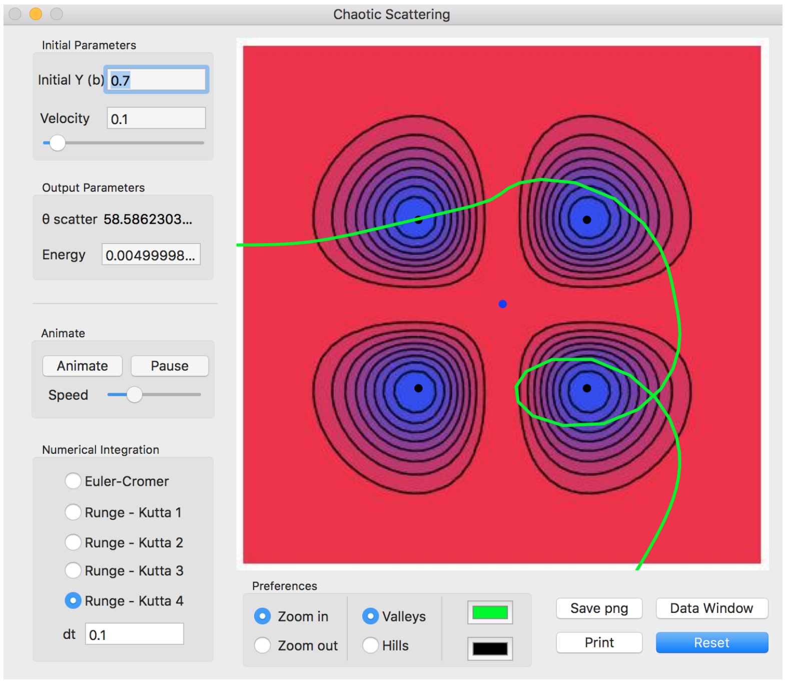
Figure 6: A screen shot of the application. The blue lines represent the potentials of the system while the green line traces the path of the particle.

algorithm, and various Runge-Kutta algorithms. With this option, users can observe how numerical integration techniques alter the path of the particle.

Once the integration algorithm and input values are chosen, the simulation generates a path as shown in Figure 6 and 7. On clicking the 'Animate' button, an animation of the particle traversing the potentials can be seen.

After the particle finishes its transit through the hilly region, it exits the system at an angle from the x axis. This angle is the scattering angle and is displayed in the the 'θ scatter' text field. Below that text field, the energy of the particle as it exits the system is displayed.

The 'Data Window' button opens a new window as seen in Figure 8 with options for archiving data in text files. The 'Archive Data for b' option runs the simulation for a set of initial y positions and records the various scattering angles

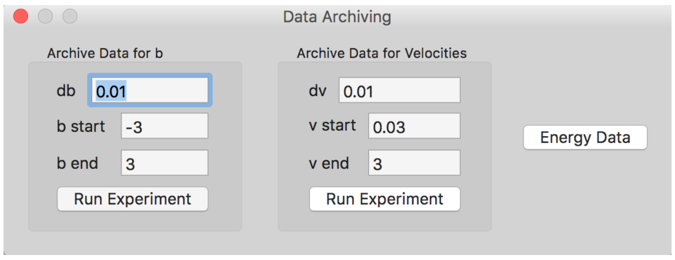


Initial Y (b) 0.7
Velocity 0.1
θ scatter 58.5862303...
Energy 0.00499998...
Euler-Cromer
Runge - Kutta 1
Runge - Kutta 2
Runge - Kutta 3
[✓ Runge - Kutta 4]
dt 0.1
[✓ Zoom in] Zoom out
[✓ Valleys] Hills
🟩
⬛

Figure 7: A similar screen as in Fig 6, however with its initial y position altered by a factor of 0.01. Additionally, this is a zoomed in version. This trajectory clearly differs by a great amount from the first one, displaying the chaotic nature of the system.

of the particle from the system. This data when plotted displays interesting fractal patterns. The 'Archive Data for Velocities' runs the simulation for different initial velocities at the same y position. Like the previous data run, this also records the scattering angle. The 'Energy Data' button allows the user to save the value of the energy of the particle at each point in its path. This is useful to tell us if the integration is accurate or not.

The most challenging portion of creating this program is determining the potentials of the four hills, and making the particle follow the potentials near the hills. Additionally, making the animation push a particle forward through the path was an intricate problem. However, the fact that this simulation provides new insights to the chaotic scattering problem makes the investigation worthwhile. Calculating the scattering angle was also confusing at first. The following section describes the process by which the program calculates the scattering angle.


Data Archiving

Archive Data for b

db	0.01
b start	-3
b end	3

Run Experiment

Archive Data for Velocities

dv	0.01
v start	0.03
v end	3

Run Experiment

Energy Data

Figure 8: The window for data archiving.

4.3 How the Scattering Angle is Calculated

The program has a loop that starts the particle at an initial position off the screen. The distance between the position where the particle starts, and the center of the region with the hills, is used as the radius of a circle. In the program, the area enclosed by this circle is considered to be the entire universe where the particle can exist. After the particle leaves its initial point, interacts with the scattering region and exits the screen, it eventually reaches the boundary of its circular universe, which is present off the screen. The coordinates of where the particle touches the circle (x, y) and the point right before it touches the circle (p, q) are recorded and can be see in Figure 9.

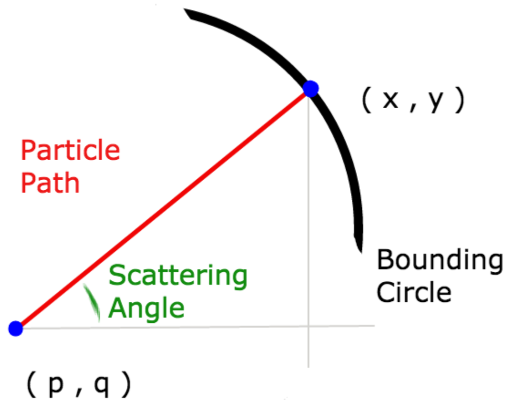


Figure 9: The points (x,y) and (p,q) are the last two points of the particle's trajectory that are calculated. The bounding circle is the boundary of the simulation's universe. Using these two points, we can calculate the scattering angle.

With these values, we can calculate the scattering angle with the formula:

$$\theta_s = \tan^{-1}\left(\frac{y-q}{x-p}\right).$$

5 Results and Analysis

5.1 Integration Accuracy Results

As discussed in Section 3.5, by the law of energy conservation, the energy of the particle in the system must remain constant at all times. Hence, we can consider only those trajectories where the particle has constant energy as physically realistic. On running the program at an initial y position of 0.71, and velocity of 0.1, we discovered that the particle's energy remained constant with calculations made with the Runge-Kutta 4 algorithm. Hence, we concluded that this algorithm's calculations were the most accurate. We can see the plot of the energy in Figure 10.

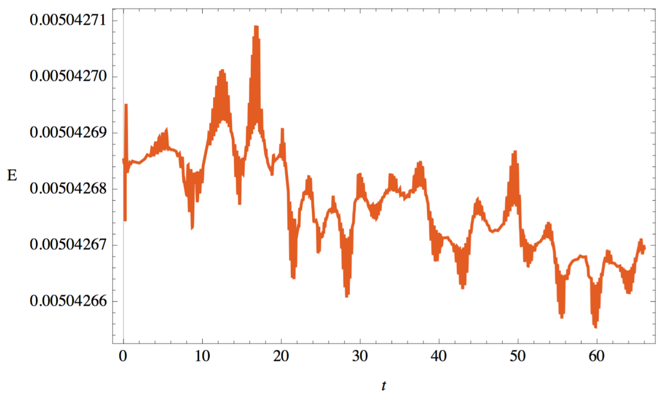


Figure 10: A zoom in of the energy of the particle as it traverses through the scattering region. Although the energy appears to be fluctuating, it is not changing by a large enough amount to affect the trajectory of the mass. The path of the particle is calculated with Runge-Kutta 4.

Although this looks like the energy is fluctuating by large amounts, this is only because the graph's vertical scale is very small. If we increase the size of the vertical scale, we see that the energy remains constant, as seen in Figure 11.

Hence, as we were able to observe that the energy of the particle fluctuated by very small amounts with the Runge-Kutta 4 algorithm, these fluctuations would

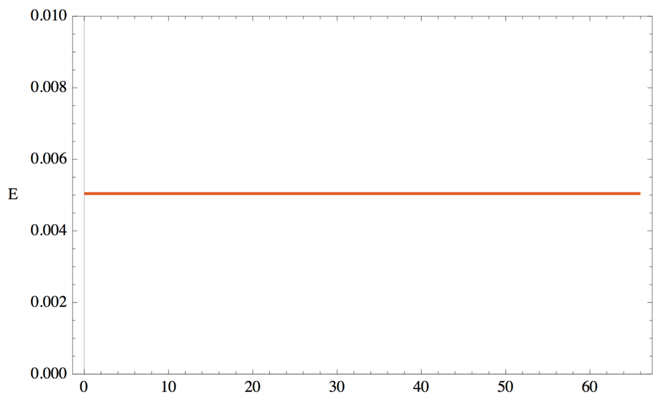


Figure 11: The energy of the particle as it traverses through the scattering region. The path of the particle is calculated with Runge-Kutta 4.

not affect the calculations on a relevant scale. Thus, we decided to take all of our data using this algorithm.

5.2 Chaotic, Non Chaotic and Fractal Regions in Plots

We ran the simulation and recorded the values of the scattering angle for impact parameters in the range $b = -5$ to $b = 5$, with intervals of $b = 0.001$ and plotted the data in Figure 12(a). From this plot, it is easy to distinguish between the non-chaotic regions–regions where the graph appears smooth (for example, between the range $b = -5$ to $b = -3$)– and the chaotic regions, the regions on the graph where there are great variations in small time steps. Looking at the nonchaotic region, we can see that increasing the impact parameter in this region does not alter the scattering angle by much. This is most likely because the path is not altered. On the other hand, looking at the chaotic regions, we can see that the scattering angle changes drastically with the impact parameter, and the graph appears to be fluctuating rapidly.

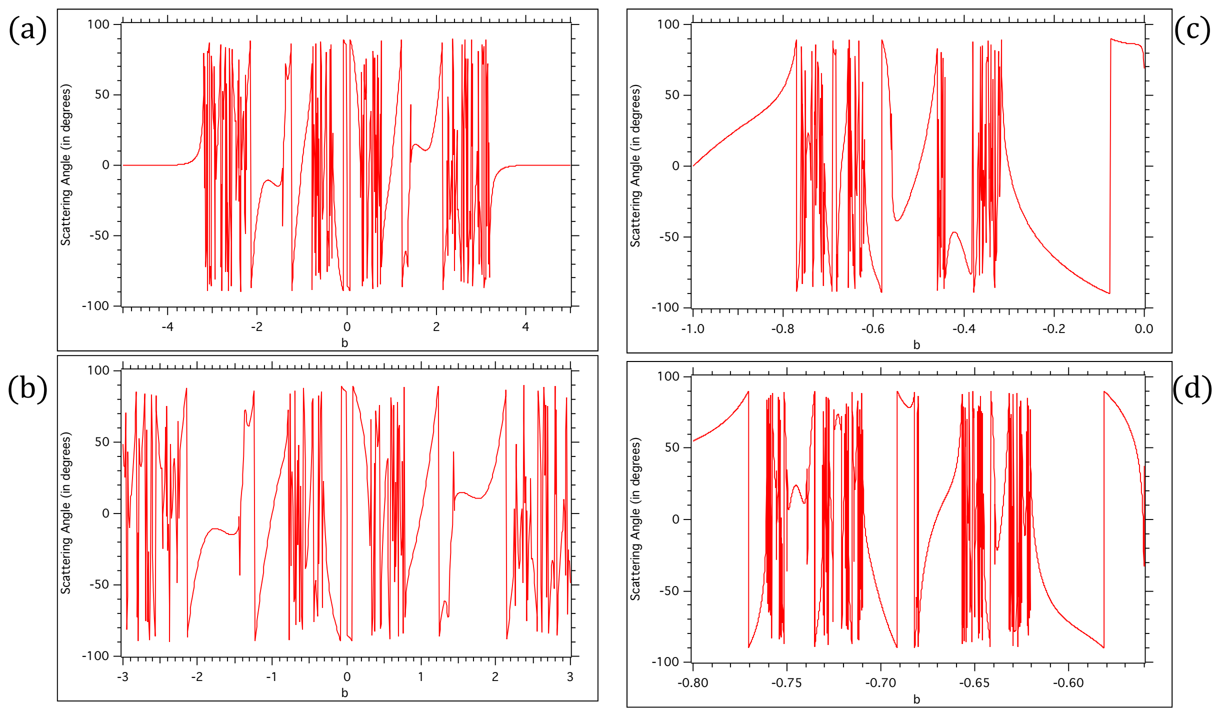


Figure 12: Plot (a) is of the scattering angle versus the impact parameter for a velocity of 0.10 . Plots (b), (c), and (d) show successive expansions of a small region in (a).

As it is difficult to clearly distinguish the pattern of this chaotic trend, we pick such a region and expand the graph along the horizontal axis as seen in Figure 12(b), which shows the region between $b = -3$ to $b = 3$. Clearly, the regions with rapid fluctuation still remain. Figures 12(c) and 12(d) show further expansions along the x axis; however, even these still display fluctuating regions.

This suggests that there are ranges for the impact parameter for which the variations in the plots show a fractal nature. Quantitatively, what may be happening is that right before a region where the graph shows chaotic nature, the particle might have been interacting with two hills, and may have been looping around some of them, as seen in Figure 6. A very small change in the impact parameter may have caused the particle to change the number of loops around the hills. This would be more than enough to drastically change the output parameter, which in this case is the scattering angle.

5.3 Changing Initial Velocity for a Fixed Impact Parameter

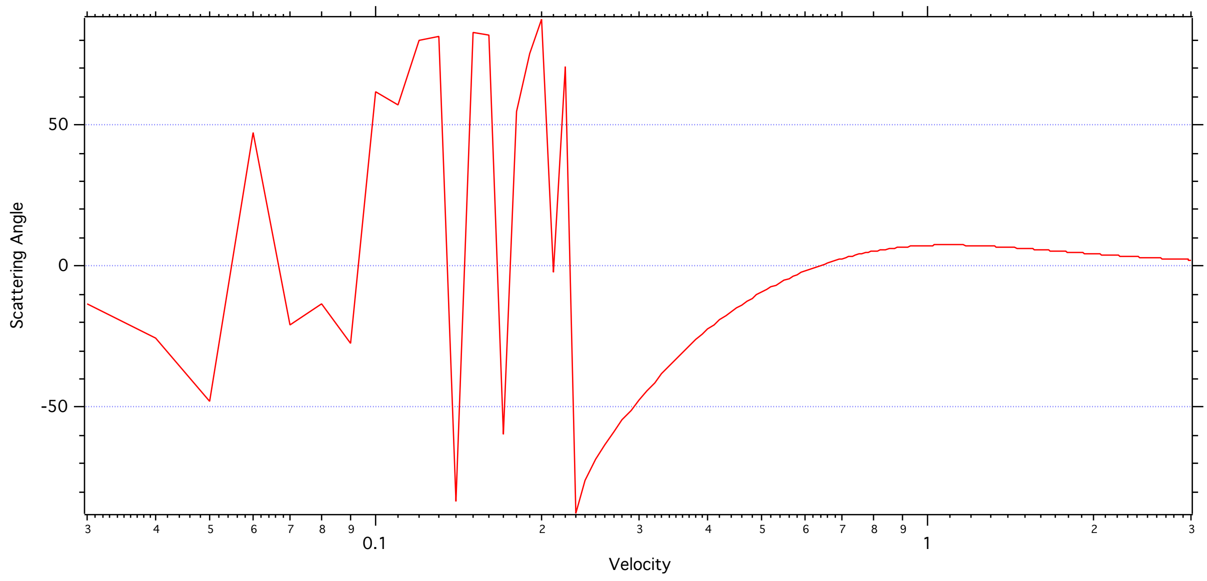


Figure 13: Plots of the scattering angle versus velocity, for an impact parameter of $b = 0.71$.

We noted earlier that changing the velocity while keeping the impact parameter fixed changes the trajectory of the particle. To gain a better understanding of this phenomenon, we plotted the scattering angle against the velocity of the particle as seen in Figure 13. Studying the graph, we can see that at smaller velocities, the scattering angle changes rapidly. After a velocity of 0.23, the scattering angle appears to increase gradually, and then as the velocity increases even further, the angle appears to gradually approach zero. This makes sense, as at higher velocities, the energy of the particle increases, and hence, it is less affected by the hills.

5.4 Effect of Initial Velocity on the Scattering Angle

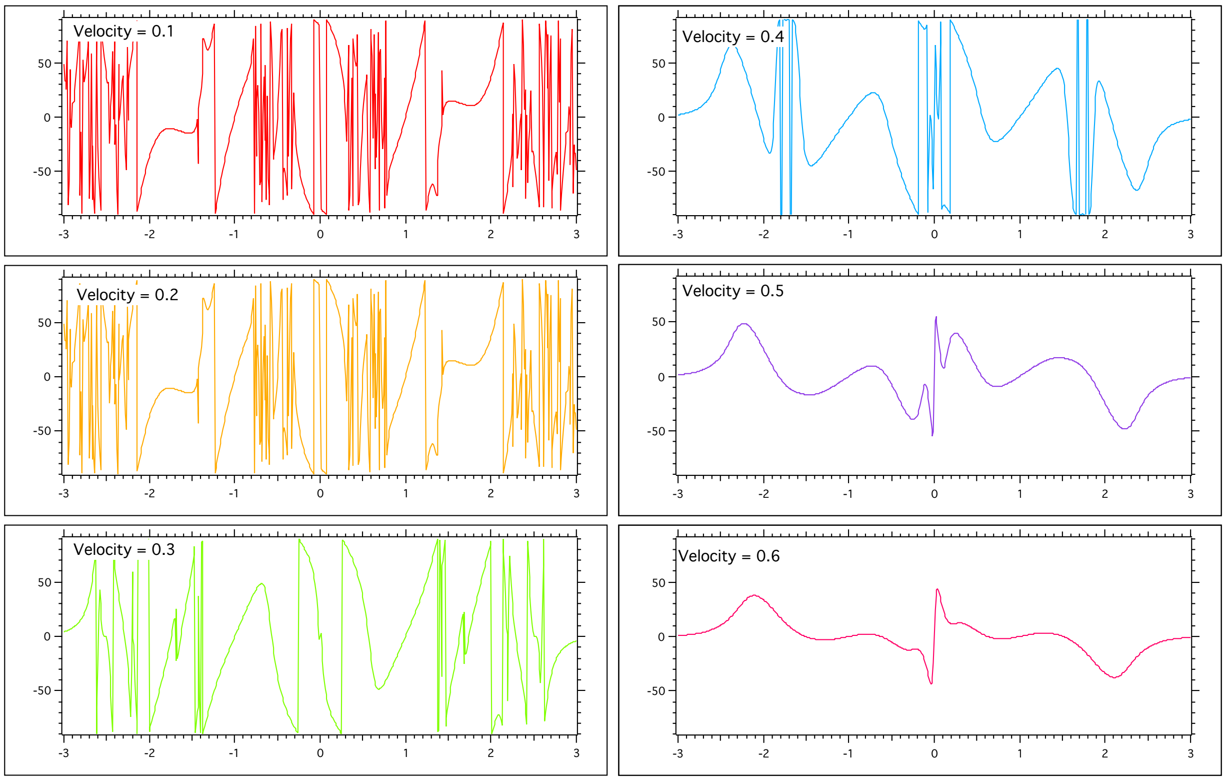


Figure 14: Plots of the scattering angle versus the impact parameter for velocities of 0.10 to 0.60.

We decided to study how changing the velocity affects the relationship between the scattering angle and the impact parameter. We plotted the data for velocities of 0.1 to 0.6, as seen in Figure 14. These graphs are very interesting, as we can see that at certain velocities, there seems to be less chaotic nature. This makes sense as we know that at higher velocities, the energy of the particle is higher, and hence the particle is not affected by the scattering region at these velocities.

6 Conclusions and Future Work

Many new advances in the study of chaotic scattering have been made in the past decade. However, there is still much left to be understood from chaotic systems. Although we now have much more data than in the past, we still do not understand certain aspects of the dynamics in chaotic scattering regimes. There is still much work left to be done with this simulation as well. The next task a researcher could investigate is duration a particle spends within the scattering region. This is also supposed to have fractal properties in its data. After collecting data for both the time and the scattering angle, the researcher could possibly try to determine what is known as the 'fractal dimension' of the system. With this, one can create an equation which helps describe fractal systems for chaotic scattering regimes. This could help further the studies of chaotic scattering and thereby help our understanding of various systems in nature. Recognizing the chaotic, fractal nature of our world can give us new insight, power, and wisdom.

7 Appendix

7.1 Step Code and Implementation of RK4

The step code for the Runge-Kutta 4 algorithm, taken from [2], can be seen below:

$$
\begin{aligned}
\triangle x_1 &= f(t,\ x(t))dt \\
\triangle x_2 &= f\left(t + \frac{dt}{2},\ x(t) + \frac{\triangle x_1}{2}\right)dt \\
\triangle x_3 &= f\left(t + \frac{dt}{2},\ x(t) + \frac{\triangle x_2}{2}\right)dt \\
\triangle x_4 &= f(t + dt,\ x(t) + \triangle x_3)dt
\end{aligned}
$$

$$
\begin{aligned}
\triangle x &= (\triangle x_1 + 2\triangle x_2 + 2\triangle x_3 + \triangle x_4)\frac{1}{6} \\
x &= x(t) + \triangle x
\end{aligned}
$$

Here, we have

- $\triangle x_1$ is the increment based on the slope at the beginning of the interval, using $f(t)$. This is also known as Euler's method.
- $\triangle x_2$ is the increment based on the slope at the midpoint of the interval, using $f(t + \frac{dt}{2})$ and $x(t) + \frac{\triangle x_1}{2}$.
- $\triangle x_3$ is again the increment based on the slope at the midpoint, but now using $f(t + \frac{dt}{2})$ and $x(t) + \frac{\triangle x_2}{2}$.
- $\triangle x_4$ is the increment based on the slope at the end of the interval, using $f(t + dt)$ and $x(t) + \triangle x_3$.

Although this appears difficult to follow, the final $\triangle x$ value is evaluated by taking the weighted average of the other $\triangle x_i$ values. For a more detailed explanation of how this algorithm works, refer to [2].

Below is the code for the implementation of the RK4 step code shown above.

```
/*--------------------------------------------------------------------------
 RK4:

 - A numerical integration technique, called Runge-Kutta 4, that
 updates the velocity and position of the particle. This is
 better than Euler Cromer and the previous RK's.

 Precondition : None
 Postcondition: Updates velocity and position of the particle
 -------------------------------------------------------------------------*/
- (void) RK4
{
    // a() -> acceleration
    // v   -> velocity
    // x   -> position

    double v = velocity;
    double x = position;

    double  dv1 = a(t,x,v) * dt;
    double  dx1 = v*dt;
    // ----------------

    double  dv2 = a( t+dt/2.0, x + dx1/2.0, v +dv1/2.0) * dt;
    double  dx2 = (v+dv1/2.0) * dt;
    // -----------------

    double  dv3 = a( t+dt/2.0, x + dx2/2.0, v +dv2/2.0) * dt;
    double  dx3 = (v+dv2/2.0) * dt;
    // -----------------

    double  dv4 = a( t+dt, x + dx3, v +dv3) * dt;
    double  dx4 = (v+dv3) * dt;
    // ----------------

    double  dv = (dv1 + 2*dv2 + 2*dv3 + dv4) / 6.0;
    double  dx = (dx1 + 2*dx2 + 2*dx3 + dx4) / 6.0;
    // -----------------

    velocity += dv;
    position += dx;
    t +=dt;
}
```

7.2 The XML Interface Builder

Figure 15 has the interface for the simulation.

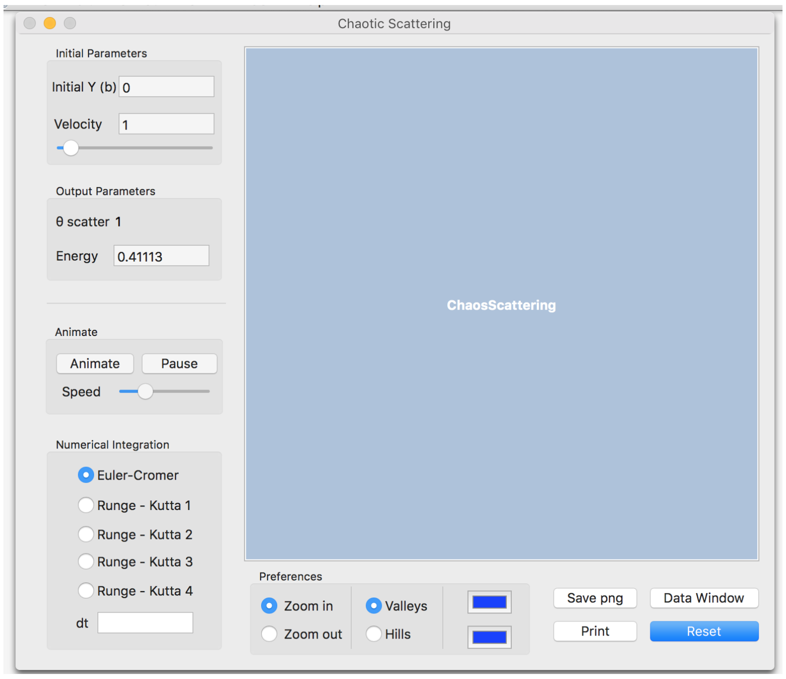
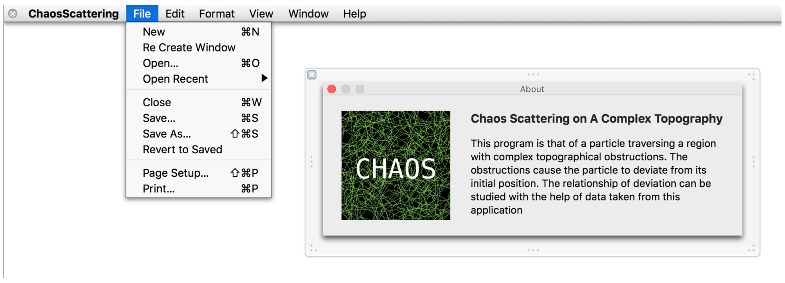


Figure 15: A screen shot of the XML Interface.



Chaos Scattering on A Complex Topography

This program is that of a particle traversing a region with complex topographical obstructions. The obstructions cause the particle to deviate from its initial position. The relationship of deviation can be studied with the help of data taken from this application

Figure 16: A screen shot of the About box for the application.

Figure 16 is that of the ‘About’ box of the program, which contains the icon for the program, and text about the application.

Figure 17 is that of the ‘Data Entry’ box of the program, which appears when the user want to take data for an experiment. It is created when the user clicks the 'Data Window' button.

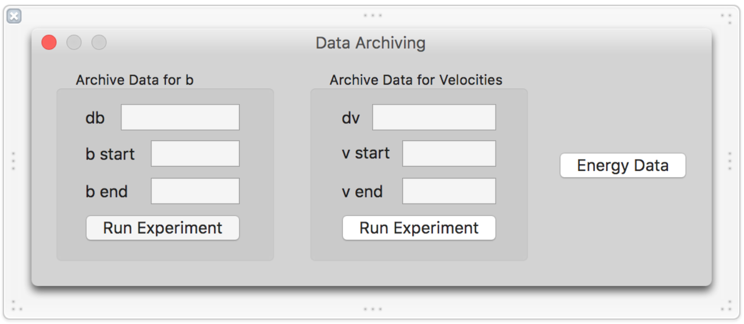


Figure 17: A screen shot of the data entry window of the program.

References

[1] Konstantinos N Anagnostopoulos. *Computational Physics, Vol I: A Practical Introduction to Computational Physics and Scientific Computing*, volume 1. Konstantinos Anagnostopoulos, 2014.

[2] DG Bettis. Runge-kutta algorithms for oscillatory problems. *Zeitschrift für angewandte Mathematik und Physik ZAMP*, 30(4):699–704, 1979.

[3] Alex Clement, Amy Clement, Greg Fenchel, Jeff Fenchel, Jayson Lynch, and NM Supercomputing Challenge. Comparing numerical integration methods. 2008.

[4] Earl A Coddington. *An introduction to ordinary differential equations*. Courier Corporation, 2012.

[5] Harvey Gould and Jan Tobochnik. More on fractals and chaos: Multifractals. *Computers in Physics*, 4(2):202–207, 1990.

[6] Jorge V José and Eugene J Saletan. *Classical dynamics: a contemporary approach*. Cambridge University Press, 1998.

[7] Edward N Lorenz. *The essence of chaos*. University of Washington Press, 1995.

[8] Edward Ott and Tamás Tél. Chaotic scattering: An introduction. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 3(4):417–426, 1993.

[9] Tao Pang. *An introduction to computational physics.* Cambridge University Press, 2006.

[10] Tolga Yalçinkaya and YingCheng Lai. Chaotic scattering. *Computers in Physics*, 9(5):511–539, 1995.