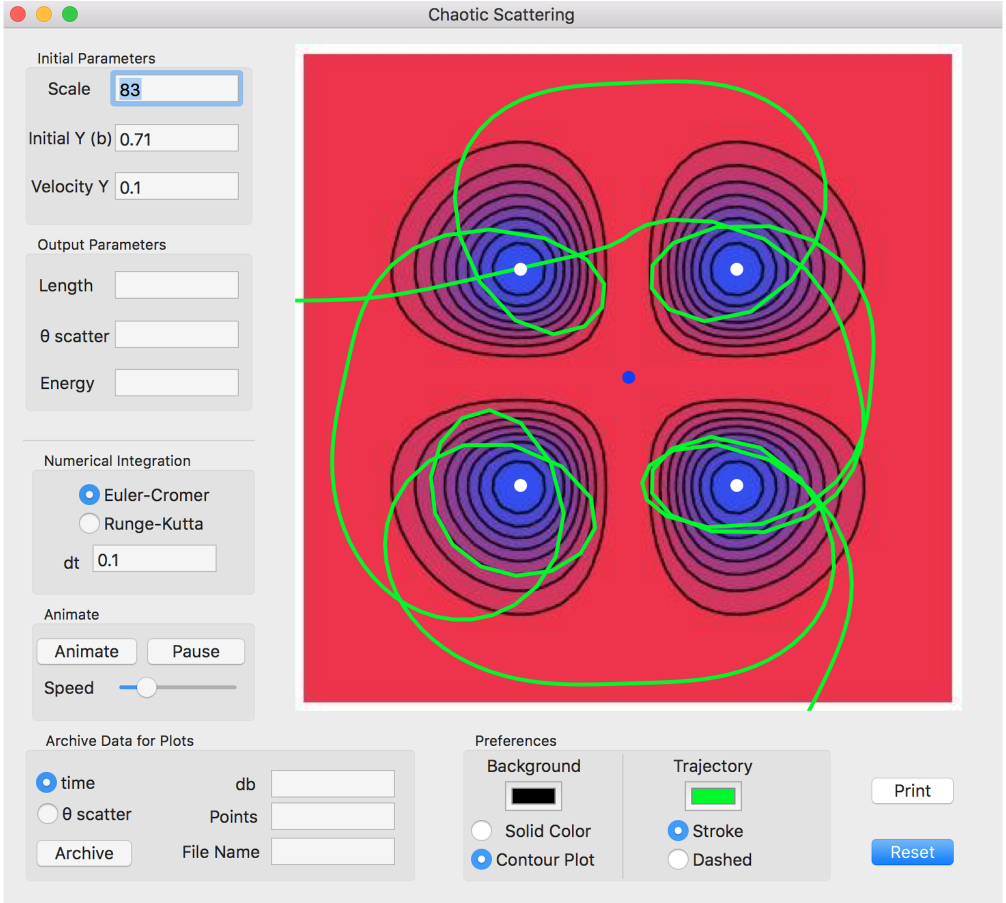
Software Description
for Chaotic Scattering for a Sliding Mass in a Complex Topography

This project relies on using numerical integration algorithms such as Euler-Cromer, and the adaptive four stage Runge-Kutta, to calculate the trajectory of a particle entering a region with four hills. Depending on the initial y position of the particle, the particle exits the region at a particular angle with respect to the x axis. The behavior of the particle‘s initial position to the exit angle is analysed.


Chaotic Scattering

Initial Parameters

Scale 83

Initial Y (b) 0.71

Velocity Y 0.1

Numerical Integration

● Euler-Cromer
○ Runge-Kutta

dt 0.1

Archive Data for Plots

● time
○ θ scatter

Preferences

Background: ○ Solid Color ● Contour Plot

Trajectory: ● Stroke ○ Dashed

Figure 1: 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.

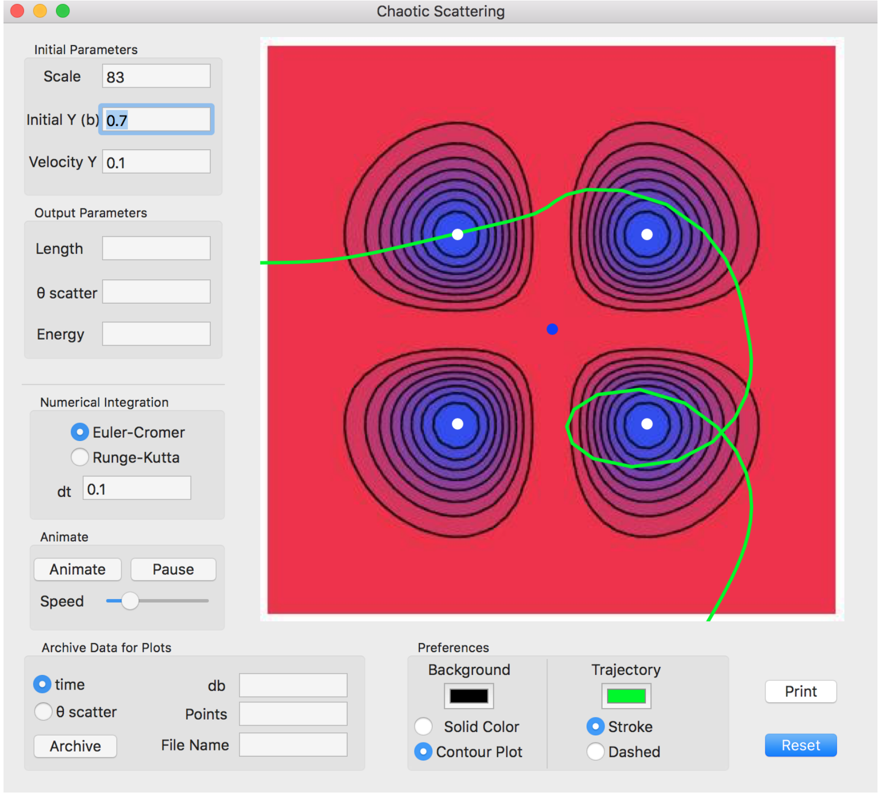
We implement this model in a Coca application for the OS X. The program is in Objective C, and we rely on the various tools available via the Coca API to run it. Xcode is the IDE we use to construct the simulation, as it provides a straightforward method of creating the interface for the application.

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 1. The trajectory of the particle, can be adjusted by changing the initial position 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. The particle’s velocity, another inpt parameter can be changed within the text field marked ‘Velocity Y’. Initially, the potential hills are present in the coordinates $(\pm 1, \pm 1)$. The scale input parameter shifts the centers of the hills outwards, and makes the hills bigger. Hence, it allows the user to enlarge the transit of the particle. This will later be implemented in the form of a magnifying glass that the user can use to zoom into a region of the particle's path.

Once these input values are provided, the simulation can either generate a path as shown in the figures, or display an animation of a particle traversing the potentials with the 'Animate' button. Additionally before plotting the trajectory, there is an option of using switching between the Euler-Cromer algorithm, and the four stage Runge-Kutta algorithm. Thereby, users can observe how the two numerical integration techniques alter the path of the particle.

After the particle finishes its transit, it exits the system at a specific time, and angle from the x axis. The 'Archive' button runs the simulation for a set of initial Y positions and records the various exit times and angles of the particle from the system. This data when plotted displays interesting fractal patterns which we can print with the simulation and analyse.

The most challenging portion of this program is determining the potentials of the four hills, and making the particle follow the potentials near the hills. Additionally, making the animation to 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 worth while.



Chaotic Scattering

Initial Parameters

Scale 83

Initial Y (b) 0.7

Velocity Y 0.1

Output Parameters

Length

θ scatter

Energy

Numerical Integration

● Euler-Cromer

○ Runge-Kutta

dt 0.1

Animate

Speed

Archive Data for Plots

● time

○ θ scatter

db

Points

File Name

Preferences

Background ■

○ Solid Color

● Contour Plot

Trajectory 🟩

● Stroke

○ Dashed

Figure 2: A similar screen as in Fig 1, however with its initial y position altered by a factor of 0.01. This clearly differs by a great amount from the first one, displaying the chaotic nature of the system.