# Fractal Images and Chaotic Scattering

**Darwin Agunos**
*Physics Department, Cal Poly Pomona, 3801 W Temple Ave, Pomona, CA 91768*

The "Gaspard-Rice" (GR) scattering system is an excellent system that embodies many characteristics of chaotic scattering while being simple and easy to understand and simulate. One interesting way to look for a qualitative behavior of chaos in this system is to plot the exit angle θ vs input ordinate. One internet article by study engineer Fabien Dournac[1] has shown chaotic output for the GR scattering system. The Wikipedia[2] page also claims that the decay rate of particles exiting the system, γ, is- 0.739. Our purpose here is to compare the results with our GR scattering system and determine if these claim are true. We will also plot the input ordinate vs bounces in a trajectory and time spent per trajectory, verify chaotic nature and find the fractal dimensions of our results as a bonus.

## I. Background

Scattering is one of the fundamental tools for studying many physical systems. In the most general sense, scattering can be defined as the problem of obtaining a relationship between an "input" variable (or variables) characterized as the initial condition or impact parameter of a system and an "output" variable (or variables) characterizing the "final" state of a system after a collision. It may happen that there is a range of initial conditions for a scattering system where a minute change may have a significant impact on the output variable. This sensitive dependence on initial conditions signifies the appearance of chaos. Another signature of chaos are the appearance of fractals. A fractals are never-ending patterns, patterns that keep repeating as you zoom in closer to them, which are created by a feedback loop. Driven from recursion, fractals are said to be the pictures of chaos.

An excellent example chaotic scattering system is the "Gaspard-Rice" (GR) scattering system – also known simply as the "three-disc" system. The GR scattering system embodies many of the important concepts in chaotic scattering while being simple and easy to understand and simulate. The concept is simple. The system consists of point particles incident on three fixed disks equally spaced around an equilateral triangle in a two-dimensional plane (Fig.1, from Wikipedia). A point particle is shot at a fixed angle and impact parameter *b*, the y distance between the initial particle trajectory and the x axis. The particle undergoes perfect elastic collisions until it exits the scattering region at an angle θ. The output of the angle depends on the impact parameter; therefore, we can write θ = θ(*b*). The time T(*b*) (the amount of time the particle spends in the scattering region before exiting) also depends on b.

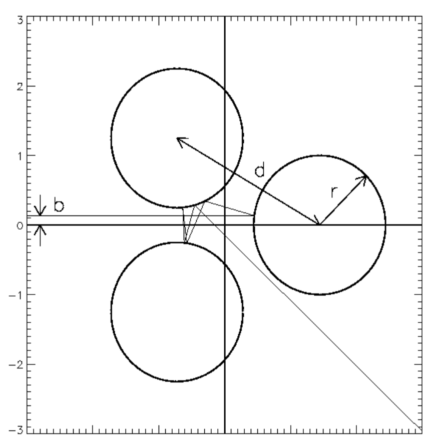


FIG. 1. Diagram of Gaspard-Rice scattering system showing major parameters (From Ref. 2.)

The chaotic nature of this system can be seen by plotting impact parameter $b$ vs exit angle θ (Fig. 2.)

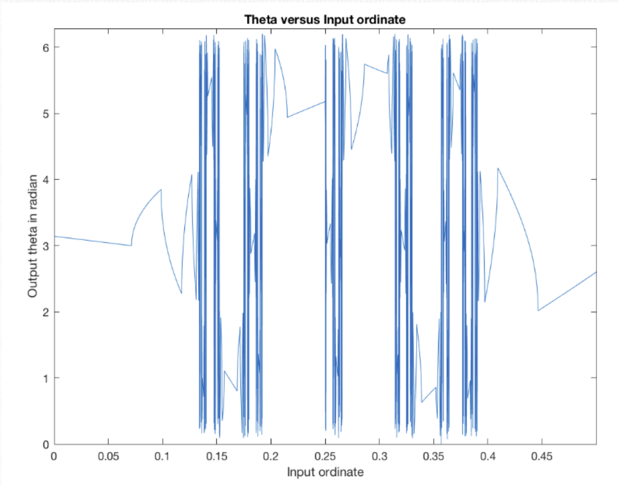


FIG. 2. θ($b$) vs $b$ for $0 < b < 0.5$. (From Fabien Dournac's coding page) [3]

If we introduce many point particles with uniformly distributed impact parameters to our system, the rate at which they exit the system is exponential with decay rate γ. We can calculate the decay rate by plotting the logarithmic number of particles shot log (N) vs T($b$) where Wikipedia[2] found the decay rate to be γ = - 0.739

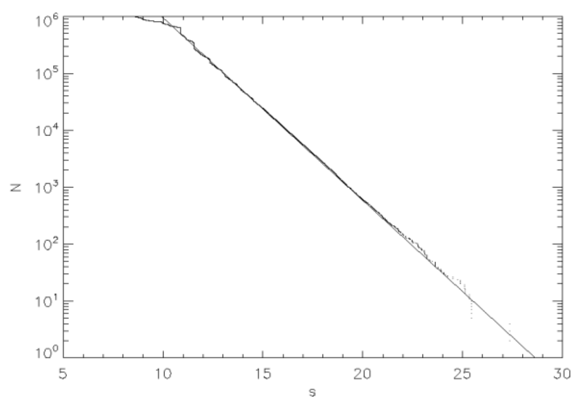


FIG. 3. Decay rate of the Gaspard-Rice scattering system (From Ref. 2.)

Our goal here is to determine if the results from Wikipedia and Dournac can be reproduced in a similar context, namely the study of the Gaspard-Rice system. We will also plot the input ordinate vs bounces in a trajectory/time spent per trajectory, verify chaotic nature and find the fractal dimensions of our results as a bonus.

## II. Methods

This project is a simple geometric optics problem solved in Python[7]. The system consists of three disks whose radii we choose to be $R_1 = R_2 = R_3 = 1$. The distances between the individual disks are d = 2.5. The disks are located at $\boldsymbol{x_1} = \frac{-d\sqrt{3}}{6}$, $\boldsymbol{y_1} = \frac{d}{2}$ (disk1 = up disk), $\boldsymbol{x_2} = \frac{d\sqrt{3}}{3}$, $\boldsymbol{y_2} = \boldsymbol{0}$ (disk2 = right disk), $\boldsymbol{x_1} = \frac{-d\sqrt{3}}{6}$, $\boldsymbol{y_1} = -\frac{d}{2}$ (disk3 = down disk) (Fig.4.).

A particle enters the system from an initial position $x_0$ and undergoes perfect elastic collisions with the three disks until it exits the system at an angle θ. We vary the height at which the particle is shot at which we will call $b$, the impact parameter. As mentioned before, the output of the angle depends on the impact parameter; therefore, we can write θ = θ($b$). The time T($b$) (the amount of time the particle spends in the scattering region before exiting) also depends on $b$.

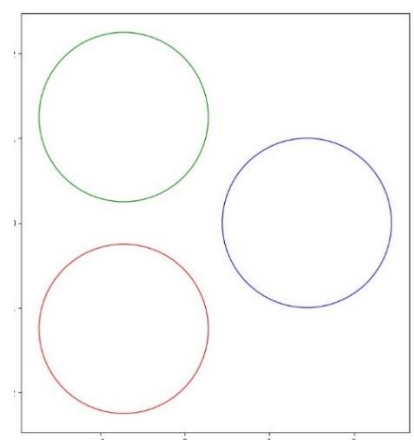

FIG. 4. Representation of disks in our coordinate system.

Now that we have set up our scattering region we make use of several computer programming techniques paired with geometry and optics. This can be broken down into four sections:

circle-line time intersections, intersection point, vector reflection and finding new angles.

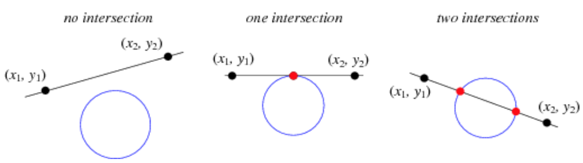


FIG. 5. Circle-line intersection (Taken from Wolfram MathWorld)[4]

Let's begin with the equations for the intersection points on a circle. A ray that collides with a disk will satisfy the two following equations:

$$x_p = x_0 + t cos(\theta) \quad (1)$$

$$y_p = y_0 + t sin(\theta) \quad (2)$$

Where $x_0$ and $y_0$ are points on the ray and $cos(\theta)$ and $sin(\theta)$ are the x and y components of velocity respectively and *t* is time of collision.

We plug equations (1) and (2) into the equation of a circle:

$$(x - x_c)^2 + (y - y_c)^2 = 1 \quad (3)$$

From here we can move onto the time intersection.

The circle-line time intersection (Fig. 5.) is essentially a collision detection algorithm. An infinite line determined by two points ($x_1$,$y_1$) and ($x_2$,$y_2$) may intersect a circle of radius r and center ($x_c$,$y_c$) at two imaginary time points (left figure), a single degenerate time point figure or two real time points (right figure).

The general equation to solve for the intersection times is the quadratic

$$t = \frac{-m \pm \sqrt{m^2 - 4pk}}{2p} \quad (4)$$

Where

$$p = 1$$

$$m = (2\cos(\theta)(x_0 - x_c) + 2\sin(\theta)(y_0 - y_c)$$

$$k = (x_0^2 + y_0^2 + x_c^2 + x_c^2 - 2x_c x_0 - 2y_c y_0 - r_c^2)$$

Solving for this equation gives us six answers (two for each disk) but the time intersection we are looking for is the minimum real number returned that is greater than 0.

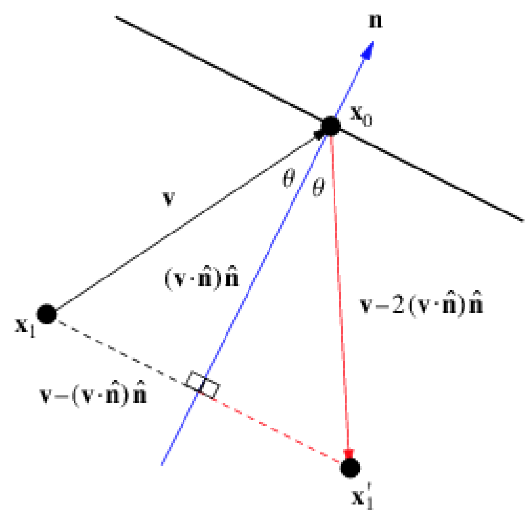


FIG. 6. Diagram of reflection (Taken from Wolfram MathWorld)[5]

The next step in our algorithm is to solve for the reflected vector. Because we are working in a system of perfect elastic collisions it is easy to solve for the reflected vector. We can take the point of contact to the disk and zoom in so we are working on a *flat* surface. From Figure 6 there is an incoming vector $v_1$ and the reflected vector is $v_1'$. We can construct another line from the normal by $Proj_{\vec{n}}(-v)$. Geometry tells us that the reflected vector is the incoming vector minus two times the projection. From here we can see that the reflected vector is

$$\vec{v_r} = \vec{v_i} - 2(\vec{v_i} \cdot \vec{n})\vec{n} \quad (4)$$

The last function we have in our program is to find the new angle we are reflecting at. This can be done easily. Since we have the components of $\overrightarrow{v_r}$ we only need to use trigonometry. The equation to solve for this angle is

$$\theta_{new} = acrtan(\frac{y_r}{x_r}) \qquad (5)$$

This algorithm is repeated until the particle has exited the scattering region (i.e. when the intersection times are either all imaginary or incredibly small) and we compute the exit angle. After that, we execute the program again using a different *b* and save the corresponding exit angle. While this algorithm is running we can also save different output variables. For each impact we save the amount of times the particle has bounced and how much time the particle has spent inside the scattering region.

For the fractals we plot can analyze them in ImageJ[6], an open-source image-processing package. Box-counting can be accomplished with ImageJ's built-in tools in two steps.

1. Make Image Binary (**Process/Binary/Make Binary)**
2. Fractal Box Count (**Analyze/Tools/Fractal Box Count**)

From the resulting graph we can then determine the fractal dimensions D of our plots.

## III. Results and Discussion

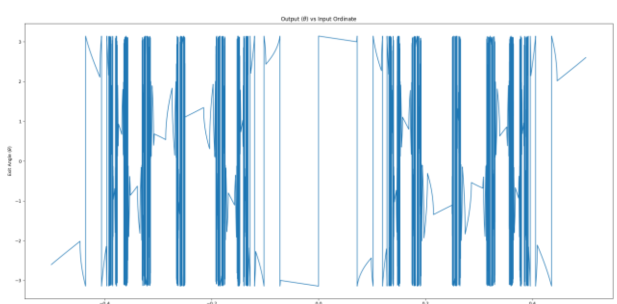

FIG. 7. θ(*b*) vs *b* for -0.5 < *b* < 0.5 for N = $10^5$ particles.

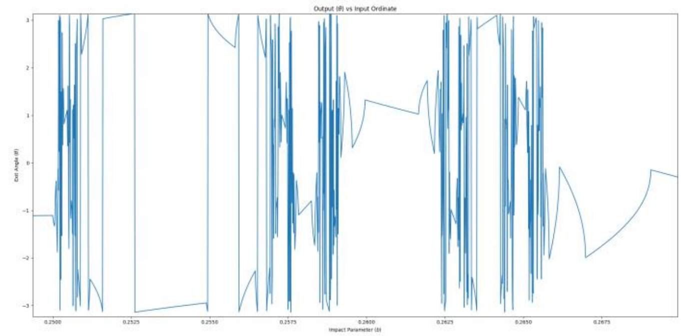

FIG. 8. θ(*b*) vs *b* for 0.2500 < *b* < 0.2675 for figure 7.

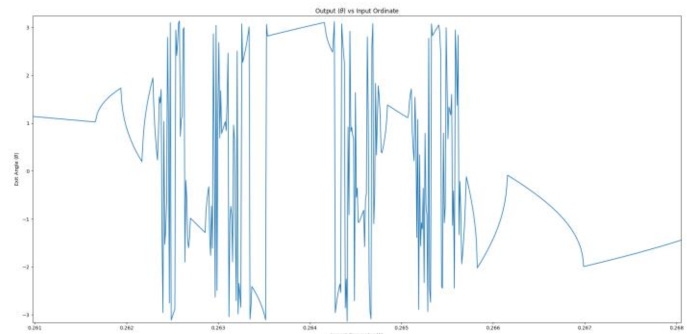

FIG. 9. θ(*b*) vs *b* for 0.262 < *b* < 0.266 for figure 7.

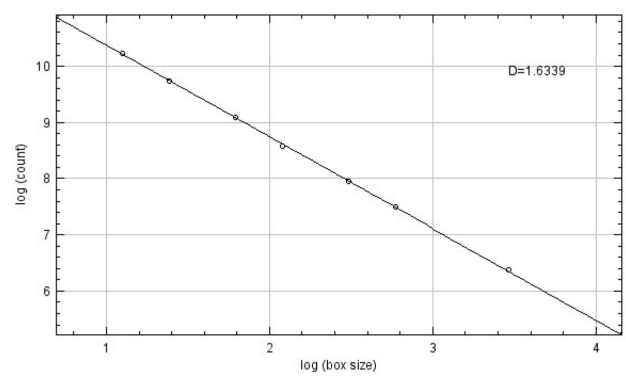


FIG. 10. Fractal dimension plot of figure 7. D = 1.6339.

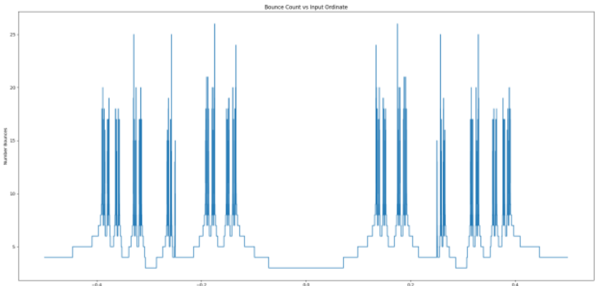


FIG. 11. Number Bounces vs *b* for -0.5 < *b* < 0.5. N =$10^5$ particles.

Plots of the exit angle/number bounces/time spent vs impact *b* are shown in Figures 7, 11 and 15 respectively. Using ImageJ's box-counting function the fractal dimension plots for Figures 7, 11 and 15 are shown in Figures 10, 12 and 16 where the fractal dimensions, D, are 1.6339, 1.4752 and 1.5074 respectively. The range of impact is the interval -0.5 < *b* < 0.5 and the number of particles shot is N = $10^5$.

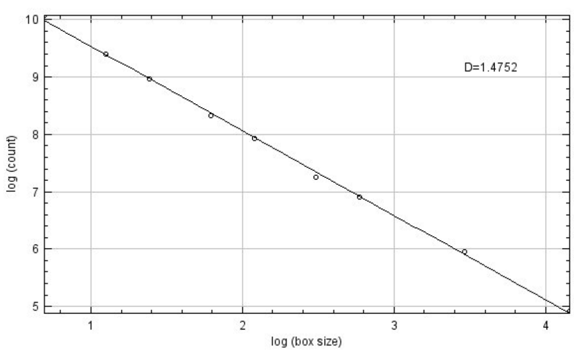


FIG. 12. Fractal dimension plot of figure 9. D = 1.4752.

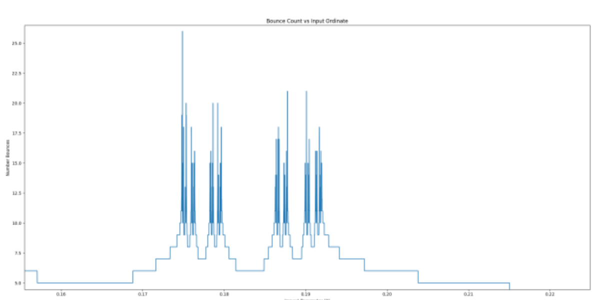


FIG. 13. Number Bounces vs *b* for 0.16 < *b* < 0.215 for figure 11.

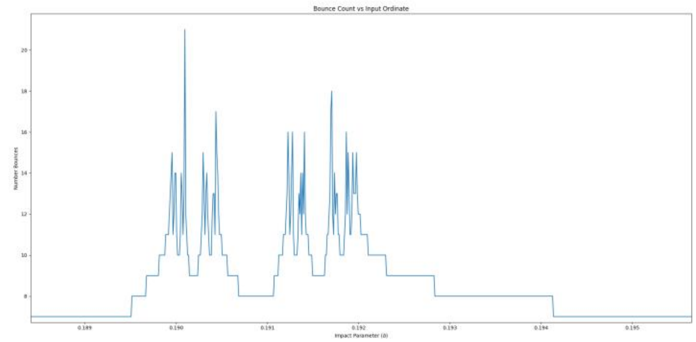


FIG. 14. Number Bounces vs *b* for 0.189 < *b* < 0.194 for figure 11.

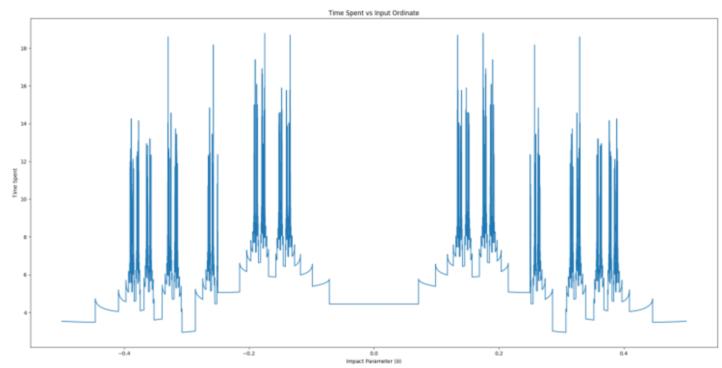


FIG. 15. T(*b*) vs *b* for -0.5 < *b* < 0.5 for N = $10^5$ particles.

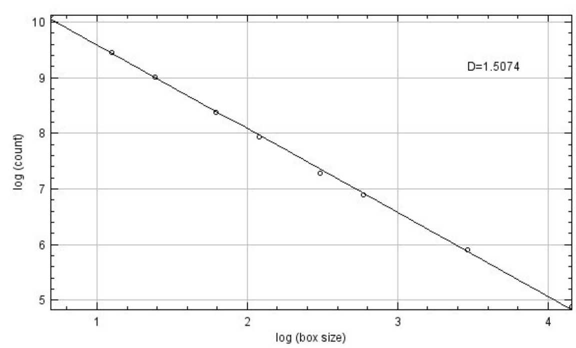


FIG. 16. Fractal dimension plot of figure 11. D = 1.5074.

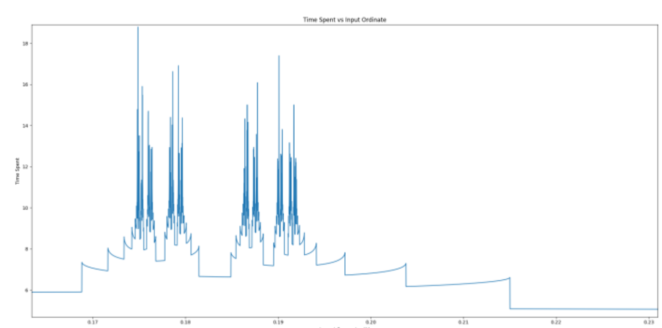


FIG. 17. T(*b*) vs *b* for 0.17 < *b* < 0.215 for figure 15.

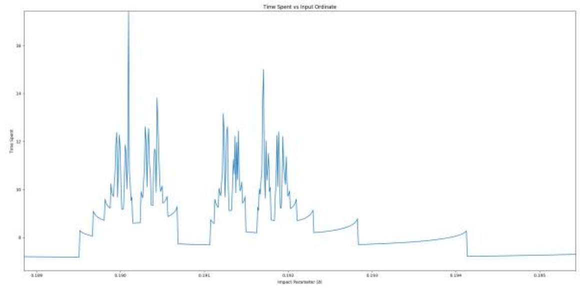

FIG. 18. T($b$) vs $b$ for $0.189 < b < 0.194$ for figure 15.

As we can see from Figures 8, 9, 13, 14, 17 and 18 the plots (Fig. 7, 11, 15) exhibit a fractal nature. What is interesting is that the original fractal images are also *mirror images* of each other. The fractal image from – 0.5 to 0 and the fractal image from 0 to 0.5 are flipped.

From our own findings we can verify that the GR scattering system exhibits chaotic behavior in certain intervals. One such trajectory can be found on Figure 19. The next step is to do a comparison between our findings and compare them to Dournac and Wikipedia.

For our simulation for recreating Dournac's results (Fig. 2.) the range of impact is the interval $0 < b < 0.5$ and the number of particles shot is $N = 10^5$. Figure 2 gives the fractal dimension to be D = 1.488 when analyzed in ImageJ (Fig. 20.). When analyzing our results for recreating Dournac's simulation (Fig. 21.) we found the fractal dimension to be D = 1.5539 (Fig.22.). This is a 4.24% difference. ( $D_{Dournac} = D = 1.488$)

In this work we recreated the Wikipedia results, for the decay rate for particles exiting the system (Figure 23), although we used $N = 10^5$ instead of $N = 10^6$. We found the resulting decay rate to be $\gamma = -0.72962$. This is a 1.2856 % difference ( $\gamma_{wiki} = -0.739$)

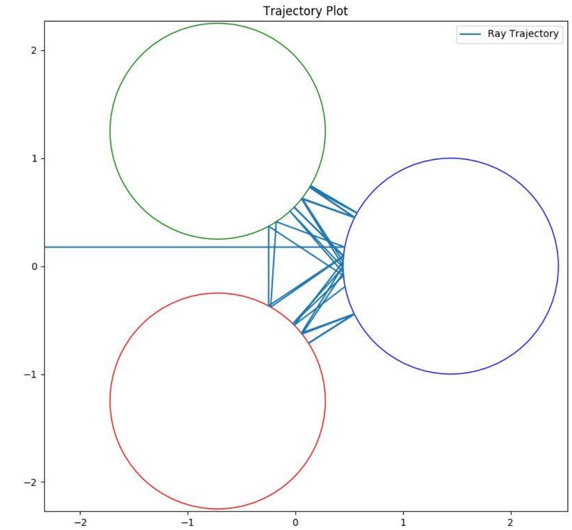


FIG. 19. Trajectory of Particle. Interval from $0 < b < 0.5$. $N = 10^6$ $b = 0.17606767606767604$ (Exact number given due to sensitivity of conditions) Number Bounces = 32

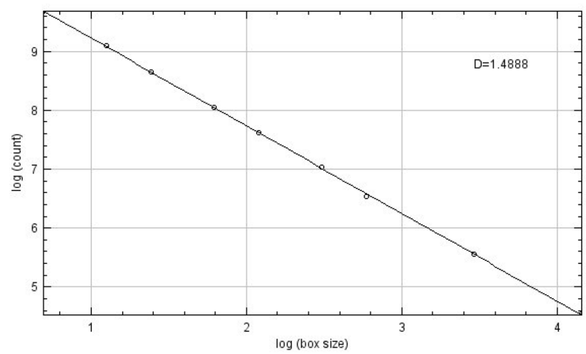


FIG. 20. Fractal dimension plot of figure 2. D = 1.488

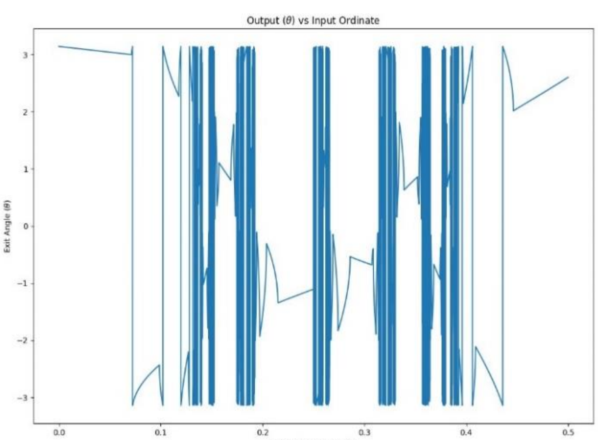


FIG. 21. θ($b$) vs $b$ $0 < b < 0.5$ for $N = 10^5$ particles.

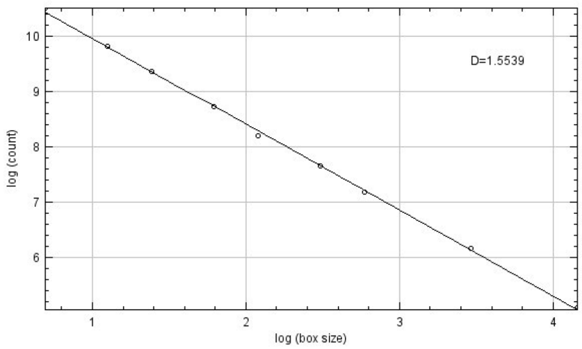


FIG. 22. Fractal dimension plot of figure 2. D = 1.5539

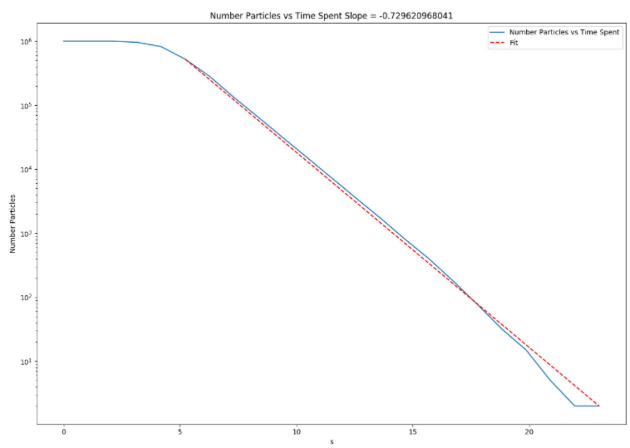


Fig 23. Number Particles vs Time Spent. $\gamma$ = -0.72962 N = $10^6$

## IV. Conclusion

This work leads to two key findings. First, we verify that the Gaspard-Rice scattering system exhibits strong sensitivity to initial conditions signifying chaos. For the three different plots (exit angle/bounce count/time spent vs impact) we displayed, a minute change in the input ordinate have shown significant impact on the output. This is a very strong indicator of chaos. We have also verified Dournac's claim by comparing our results with his. While it is true that our fractal dimensions vary it is not a substantial amount. The only attribute I can think of is that we may have solved this geometric optics problem differently. Dournac solved it in Mathematica while this work was solved in Python.

Our second finding is verifying that Wikipedia is a reliable source for describing the nature of chaotic scattering, mainly the section on decay rate. Comparing our results, our decay rates differed by a meager 1.2856 %. While there is a decent amount of references made to credited papers, upon exploration the figures we explored are the work of Peter Mills whose thesis is *Noisy Chaotic Scattering*. Since we do not know how Peter Mills solved this problem will attribute this difference to a possibility of a different algorithm being used to work out the scattering system.

Future work possible would be to include another disk in the system or change the radii. We can compare how this would affect the fractal dimensions of our results.

## VI. Acknowledgements & References

I would like to thank Cal Poly Pomona for the use of their resources and facilities. I would like acknowledge Professor Alexander Small for his guidance in this work as well as Bo Shrewsbury for advice on the coding algorithm developed in Python.

1. Dournac, F. (n.d.). Chaotic scattering in Gaspard-Rice's system. Retrieved May 12, 2018, from https://dournac.org/info/chaotic_scattering
2. Mills, P. (2018, January 03). Chaotic scattering (Decay Rate). Retrieved May 12, 2018, from https://en.wikipedia.org/wiki/Chaotic_scattering
3. Dournac, F. (n.d.). Chaotic scattering in Gaspard-Rice's system (Section 2, Figure 5). Retrieved May 12, 2018, from https://dournac.org/info/chaotic_scattering
4. Weisstein,Eric W. "Circle-Line Intersection." From *MathWorld*--A Wolfram Web Resource. http://mathworld.wolfram.com/Circle-LineIntersection.html
5. Weisstein,Eric W. "Reflection." From *MathWorld*—A Wolfram Web Resource. http://mathworld.wolfram.com/Reflection.html
6. https://imagej.nih.gov/ij/
7. https://www.enthought.com/product/canopy/