Homework Set 3: Monte Carlo Simulations and Numerical Methods for ODEs

AMS 326-1: Numerical Analysis, Spring 2025
Joe Martinez

April 22, 2025

All calculations for this assignement are done in Python, with the source code submitted with this report as hw3.py. The Python libraries used for this project are Shapely, SciPy, Numpy, matplotlib, sys, random, and math. For problems that take more than a second, a progress bar is shown in the terminal when the script is running.

1 Buffon's Varying Sized Disks (Problem 3-1)

Similar to the Buffon's Needle problem, we were tasked to find the probability a randomly thrown disk crossing at least 1, 2, 3, or 4 lines with disks of different diameters for each scenario. Our approach to the solution was to simulate each scenario using 4,444,444 disks over 1000 lines. The different sizes for disk diameter were:

$$d = \frac{1}{10}, \frac{2}{10}, \frac{3}{10}, \frac{4}{10}, \frac{5}{10}, \frac{6}{10}, \frac{7}{10}, \frac{8}{10}, \frac{9}{10}, \frac{10}{10}, \frac{15}{10}, \frac{20}{10}, \frac{30}{10}$$

Simulating our disks and lines, the lines lied on the vertical axis with a distance of 1 between consecutive lines, and we randomly sampled a horizontal x-axis value from a uniform distribution for the center of each disk. Since the lines are in the vertical axis, there was no need to randomly sample y values for each disk since it doesn't affect determining if the disk crossed a line.

Determining whether a disk crossed a line was done by iterating through the ith closets pair of lines to the left or right of the disk and calculating the distance between the center of disk to each line. For example, for the first iteration, d1 would be the distance to the 1st closest line to the left of the disk, and d2 would be to the 1st closest line to the right of the disk. The disk crosses a line if the distance to that line is smaller than or equal to the radius of the disk. We iterate through the pairs of lines until we find that both distances are larger than the radius of the disk, counting each crossed line each iteration.

Since our goal was to calculate the percentage of disks that cross as least 1,2,3 and 4 lines, the probability that a disk crossed at least n lines is

$$P(lines = n) = \frac{\sum_{i=n}^{l} \#\text{ of disks that crossed i lines}}{\text{total } \#\text{ of disks}}$$

where l is the total number of possible lines that a disk can cross, which is $\lfloor d/1 \rfloor + 1$. The pseudocode for this algorithm is seen in Algorithm 1.

Algorithm 1: Buffon Disk Monte Carlo Simulation

Input: The number of disks to throw $disks$, the diameter of each disk $disk_d$, the number of needles $needles$, and the seperation between needles $lineWidth$

Output: A array of intergers descriping how many disks landed on i lines, where i is the ith index of the array

BuffonDisks $(disks, disk\_d, needles, lineWidth)$

- $disk\_r \leftarrow disk\_d/2$;
- $maxLines \leftarrow \lfloor disk\_d/lineWidth + 1 \rfloor$;
- $diskTouchLines \leftarrow$ Initialize a new array of size $maxLines + 1$;
- **for** $i \leftarrow 0$ **to** $disks$ **do**
  - $lines \leftarrow 0$;
  - $step \leftarrow 0$;
  - $d1 \leftarrow 0$;
  - $d2 \leftarrow 0$;
  - **while** $d1 < disk\_r$ *or* $d2 < disk\_r$ **do**
    - $x \leftarrow$sample from $Uniform(0, maxLines)$;
    - $prevLine \leftarrow \lfloor x - step \rfloor$;
    - $nextLine \leftarrow \lfloor x + step \rfloor$;
    - $d1 \leftarrow x - prevLine$;
    - $d2 \leftarrow nextLine - x$;
    - **if** $d1 == 0$ *or* $d2 == 0$ **then**
      - $lines \leftarrow lines + 1$;
    - **else**
      - **if** $d1 < disk\_r$ *and* $prevLine \geq 0$ **then**
        - $lines \leftarrow lines + 1$;
      - **end**
      - **if** $d2 < disk\_r$ *and* $nextLine \leq lastLine$ **then**
        - $lines \leftarrow lines + 1$;
      - **end**
    - **end**
    - $step \leftarrow step + 1$
  - **end**
- **end**
- **for** $i \leftarrow 1$ **to** 1 **by** -1 **do**
  - $disksTouchLines[i-1] \leftarrow$ $disksTouchLines[i-1] + disksTouchLines[i]$;
- **end**

Disk Diameter	Lines Crossed				
	0	1	2	3	4
0.1	0.899968	0.100032	0.0	0.0	0.0
0.2	0.799593	0.200407	0.0	0.0	0.0
0.3	0.699829	0.300172	0.0	0.0	0.0
0.4	0.600033	0.399967	0.0	0.0	0.0
0.5	0.499997	0.500003	0.0	0.0	0.0
0.6	0.400116	0.599884	0.0	0.0	0.0
0.7	0.300230	0.699770	0.0	0.0	0.0
0.8	0.199904	0.800096	0.0	0.0	0.0
0.9	0.099597	0.900403	0.0	0.0	0.0
1.0	0.0	1.0	0.0	0.0	0.0
1.5	0.0	1.0	0.500103	0.0	0.0
2.0	0.0	1.0	1.0	0.0	0.0
3.0	0.0	1.0	1.0	0.999003	0.0

Table 1: Probabilities of disks crossing specified lines

Running Algorithm 1 for our list of varying disk diameters, the probabilities can be seen in Table 1. An important note is that for a disk diameter of 1, the possibility of crossing 2 lines is incredible small, but not impossible. Same logic applies for a disk diameter of 2 with crossing 3 lines and for the a disk diamtere of 3 with crossing 4 lines. The probability distribution function (PDF) of a disk crossing 0, 1, 2, 3, and 4 lines based on disk diameters are in Figures 1, 2, 3,4, and 5 respectively.

2 Maximum Cut for a 4-leaf Rose Curve (Problem 3-2)

Given the 4-leaf "rose" graph $r = sin2\theta$, or $(x^2 + y^2)^3 = 4x^2y^2$ in Cartesian coordinates, our goal was to find the best spot to place a retangular cutter of size $1 \times \frac{1}{\sqrt{2}}$ to maximize the area of the cut.

Our approach to find a maximum area cut from a generally random starting spot was to use the L-BFGS-B algorithm, which is a limited memory numerical optimization algorithm. The BFGS-B algorithm is a quasi-Newton method that approximates the Hessian matrix to perform gradiant desent, finding the local minimum of a function and optimizing it. In this case, the function we want to optimize is the area of the cut applied on the rose graph, aiming for the maximum area. Therefore, as shown in Algorithm 3, the aim was to minimize the negative area of the cut area, hence producing a maximum area. The L-BFGS-B algorithm in this case is optimizing Algorithm 2, the calculation of the area cut from the rose graph and a defined cutter.

Using Algorithm 3 on two different occasions, starting from a random spot in the range $-0.25 \leq x \leq 0.25$ and $-0.25 \leq y \leq 0.25$ (using the heatmap for

Algorithm 2: Negative Cutter Area from Rose Curve

Input: The x, y, and angle of cutter x, y, and $angle$ respectfully, the rose curve Polygon, $rose$, and a cutter Polygon creating function, **createCutter**.

Output: The negative area of the curve

NegativeIntersectionArea $(x, y, angle, rose,$ ***createCutter***$)$

$cutter \leftarrow$ **createCutter**$(x, y, angle)$;

$intersection \leftarrow$ the intersection of $rose$ and $cutter$;

return the negative of the intersection area

Algorithm 3: Maximum Area Cut For Rose Curve Monte Carlo

Input: A polygon representing the rose curve to maximize the cut for, $rose$.

Output: The maximum area of a cut from curve, and both the location fo the cutter and a plot of the cutter are outputted

MaxRoseCut *(rose)*

$init\_x \leftarrow$ sampled from $Uniform(-0.25, 0.25)$;

$init\_y \leftarrow$ sampled from $Uniform(-0.25, 0.25)$;

$init\_theta \leftarrow$ sampled from $Uniform(0, 2\pi)$;

$init\_params \leftarrow [init\_x, init\_y, init\_theta]$;

$bounds \leftarrow [(-0.25, 0.25), (-0.25, 0.25), (0, 2\pi)]$;

$results \leftarrow$
L-BFGS-B(**NegativeIntersectionArea**, $init\_params, bounds, rose$);

output graph of best cutter from results;

return max area from results;

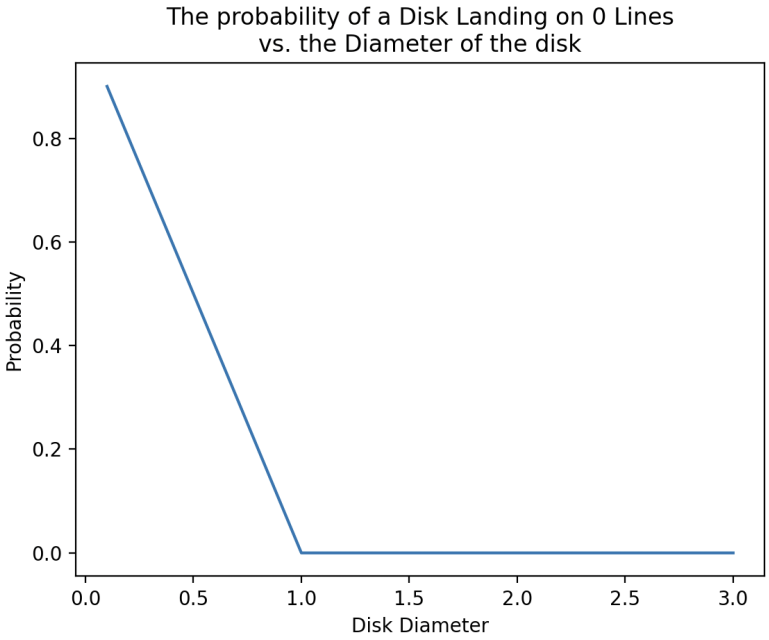

Figure 1: PDF of crossing 0 lines based on disk diameter

maximum area), two maximum area cuts were found. Figure 6 shows a maximum cut of area 0.5878186487084814, placing the cutter at $x = 2.557233 \times 10^{-6}$, $y = -2.768929 \times 10^{-5}$, and an angle of $3.141593297595777 \approx 2\pi = 0^o$. Figure 7 shows a maximum cut of area 0.5878186629229423, placing the cutter at $x = 1.403466 \times 10^{-9}$, $y = -7.215794 \times 10^{-6}$, and an angle of $4.712389002212077 \approx 2\pi + \frac{\pi}{2} = 90^o$.

3 The Trajectory of a Plane (Problem 3-3)

A plane is flying starting from (a, 0) to an airport at (0,0) with a constant speed of v_0 always heading towards the airport, and a constain wind of speed of w in the positive y direction. Both of these fact combined create an ordinary differential equation detailing the change in y position with respect to the change in x position of the plane:

$$\frac{dy}{dx} = \frac{y}{x} - k\sqrt{1 + (\frac{y}{x})^2}$$

where $k = \frac{w}{v_0}$. Given the initial values of $a = 100$, $w = 44$, and $v_0 = 88$, our goal is to map out the plane's trajectory using this ordinary differential equation.

The Runge-Kutta 3 (RK3) method uses slopes of two predicted future point to calculate an averaged weighted slope to numerically estimate a point at a

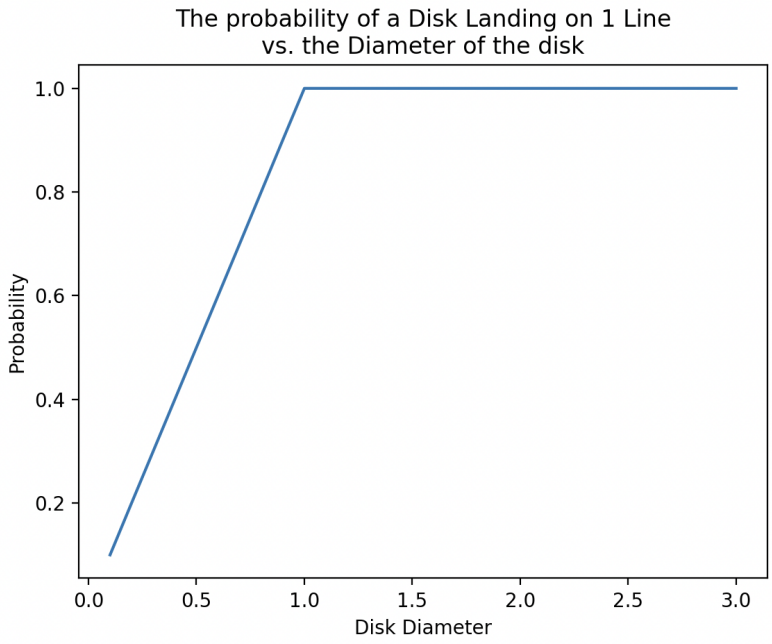

Figure 2: PDF of crossing 1 lines based on disk diameter

h independent variable distance away from the starting independent variable point. This method can be used iteratively, plugging in new points into the method to estimate even furthur points from the stating point.

Inputting our intial values and start from the point (100, 0), and using an h value of -10^{-4}, we found the trajectory of the plane to be as shown in Figure 8.

Algorithm 4: Runge-Kutta 3 Method for Plane Trajectory

Input: The starting x and y coordinates of the plane, x and y, the differential equation defined as dy/dx, $f(x, y)$, and the incremental step h

Output: A list of values x and y values indicating the airplane's trajectory, as well as a plot of the trajectory

SimulatePlane $(startX, startY, h)$

- $x \leftarrow startX$;
- $y \leftarrow startY$;
- $trajectory \leftarrow$ initialize new array;
- **while** $x \geq 0$ *and* $y \geq 0$ **do**
  - $k_1 \leftarrow f(x, y)$;
  - $k_2 \leftarrow f(x + \frac{h}{2}, y + \frac{h}{2}k_1)$;
  - $k_3 \leftarrow f(x + h, y - hk_1 + 2hk_2)$;
  - $y \leftarrow y + \frac{h}{6}(k_1 + 4k_2 + k_3)$;
  - $x \leftarrow x + h$
  - **if** $x \geq 0$ *and* $y \geq 0$ **then**
    - append to $trajectory$ (x, y);
  - **end**
- **end**
- **output** Plot of $trajectory$;
- **return** $trajectory$;

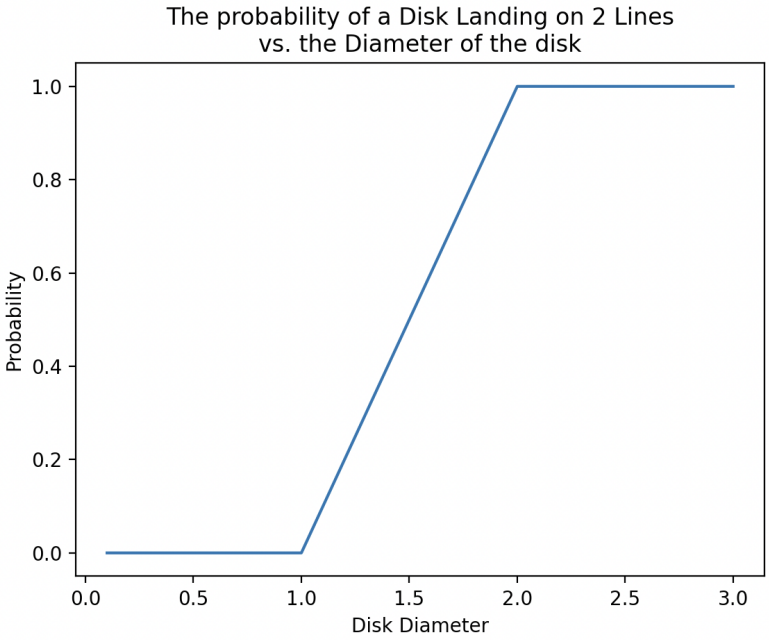

Figure 3: PDF of crossing 2 lines based on disk diameter

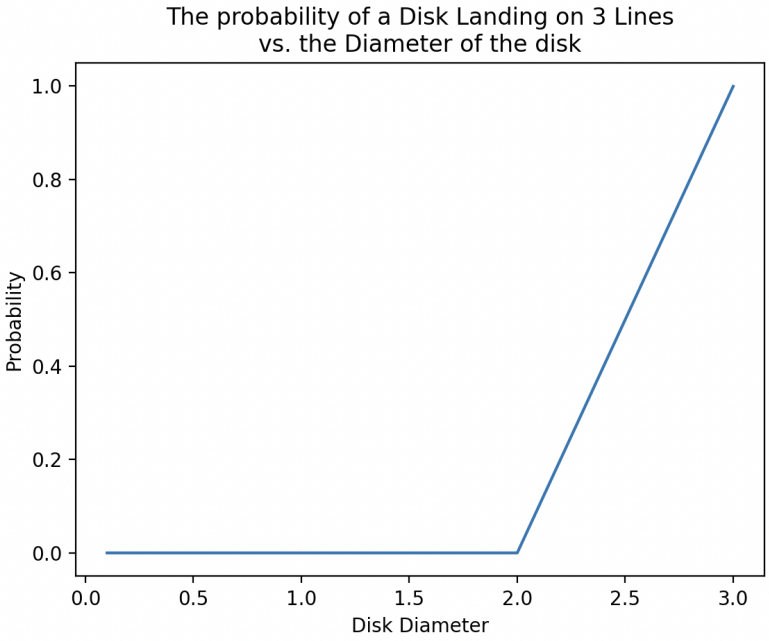

Figure 4: PDF of crossing 3 lines based on disk diameter

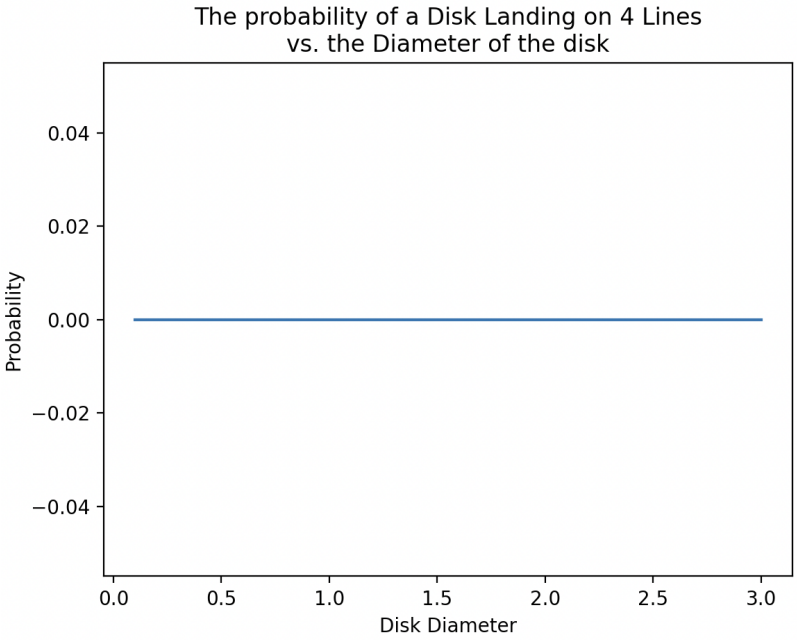

Figure 5: PDF of crossing 4 lines based on disk diameter

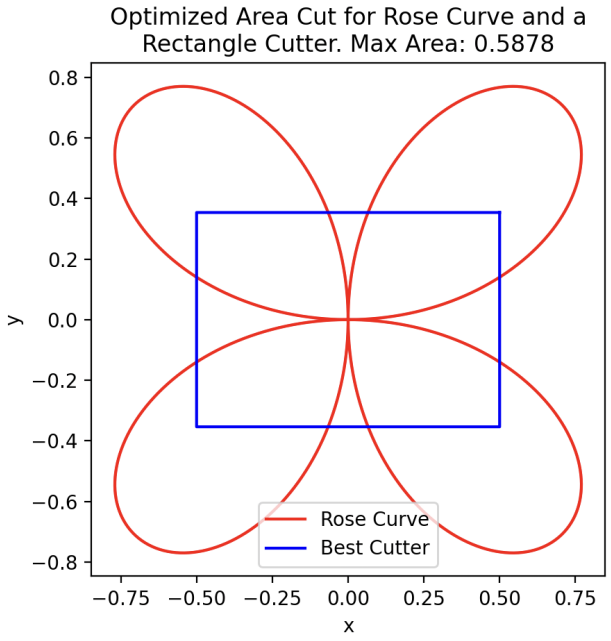

Figure 6: Maximum area cut with an angle of 0 degrees

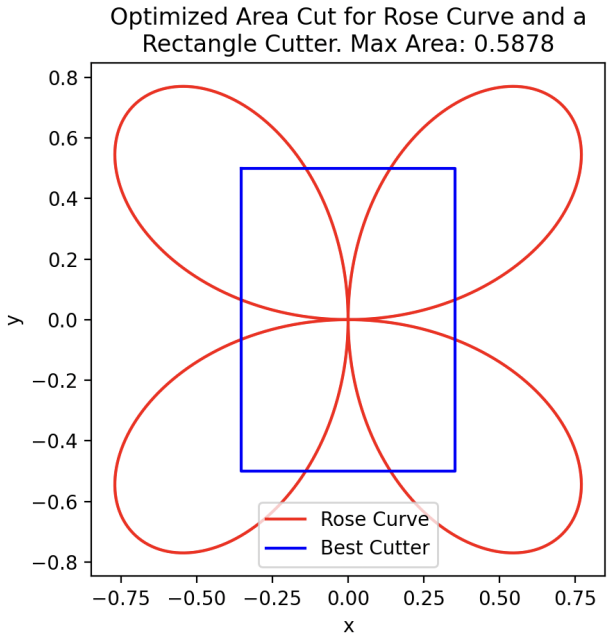

Figure 7: Maximum area cut with an angle of 90 degrees

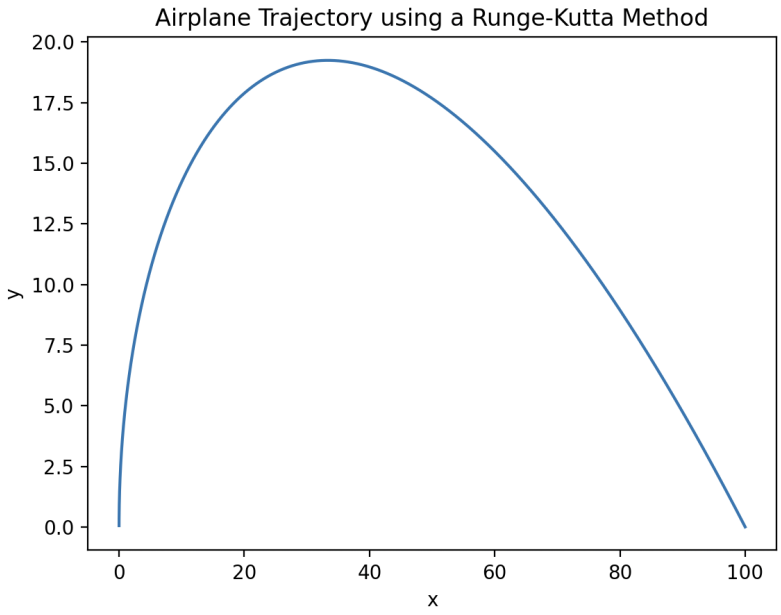

Figure 8: The trajectory of the plan estimated with the Runge-Kutta 3 Method, starting at point (100, 0)