AMS 326-1: Exam 3

Numerical Analysis, Spring 2025
Author: Joe Martinez
SBU ID: 112416928

April 29, 2025

All of the problems solved were done in Python, with the use of the random, math, shapely, numpy, and mathplotlib libraries.

1 Problem 2: Buffon's Needles on a Quadrifolium

Given the quadrifolium expressed in polar coordinates as $r = sin(2\theta)$, and a box $[-1, -1] \times [1, 1]$ enclosing it, our goal was find the probability of tossing randomly tossed needles of varying length crossing our "rose" or quadrifolium curve. If a needle crossed the curve twice, it was only counted once. Our approach to simulate this was to run a Monte Carlo simulation with 444444 needle with the list of varying lengths being

$$L = \frac{1}{10}, \frac{1}{5}, \frac{1}{4}, \frac{1}{3}, \frac{1}{2}, \frac{1}{1}$$

Being a Monte Carlo simulation, we randomly sampled x and y values for each needle a uniform distribution in the range of $[-1, 1]$, given the box enclosing the curve to represent the center of the needle. For the orientation of the needle, we sampled an angle from a unfirom distribution in the range of $[0, 2\pi]$. Having the angle and the length of the needle, we create a line that represented the needle by addding and subtracting the distance from the center of the needle to its endpoints. This was done by as shown below, where θ represents the angle and l represents the length of the needle:

$$\Delta x = \frac{l}{2} cos(\theta)$$

$$\Delta y = \frac{l}{2} sin(\theta)$$

$$\text{Line: } [(x - \Delta x, y - \Delta y), (x + \Delta x, y + \Delta y)]$$

Each needle and the "rose" curve were presenteded using hte LineString object in the Shapely Python library. The code to generate the rose curve can

be seen in Figure 2. Using the *shapely::intersection(curve1 ,curve2)* function, we found if a given needle intersected the "rose" curve. We would count the number of needles that cross the curve and divide this by the total number of needles generated, producing our probability for a given length. This was done for each length. The algorithm for calculating the probability given a length of a needle is in Figure 1.

```
def simulate_needles(l, num_needles):
    print(f"\nSimulating Needles:\n  length {l}\n  number: {num_needles}")
    rose_poly = rose_curve()

    # Needles
    crossing = 0
    count = 0
    percentUnit = (int)(num_needles / 100)
    while (count < num_needles):
        x = random.uniform(-1, 1)
        y = random.uniform(-1, 1)
        theta = random.uniform(0, 2 * math.pi)
        deltaX = (l/2) * math.cos(theta)
        deltaY = (l/2) * math.sin(theta)
        needle = LineString([(x-deltaX, y-deltaY), (x+deltaX, y+deltaY)])

        crossPoint = shapely.intersection(rose_poly, needle)
        if not crossPoint.is_empty:
            crossing += 1
        count += 1

        if (count % percentUnit == 0):
            print(f"Progress: {(int)((count / num_needles)*100)}%\r", end="")

    print("Progess: 100%")
    print('total crossing: ', crossing)
    print('probability: ', (crossing / num_needles))
    return (crossing / num_needles)
```

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

The following are the probabilities we found

```python
def rose_curve(k: float = 2, a: float = 1, num_points: int = 1000) -> Polygon:
    """
    This function creates a rose curve of the form r = a * cos(k * theta)

    Parameters:
    k (float): 'k' number
    a (float): 'a' number
    num_points (int): number of points to create curve with

    Returns:
    Polygon: A rose curve that fits the specifications
    """
    theta = np.linspace(0, 2 * np.pi, num_points)
    r = a * np.sin(k * theta)
    x = r * np.cos(theta)
    y = r * np.sin(theta)

    points = list(zip(x, y))
    return LineString(points)
```

Figure 2: PDF of crossing 0 lines based on disk diameter

Needle Length	Probability of Crossing
0.1	0.1416
0.2	0.2725
0.25	0.3341
$\frac{1}{3}$	0.4307
0.5	0.6031
1	0.7972