Calculation of Area under the curve using Monte Carlo simulations

by-

Prajual Pillai(193100069)

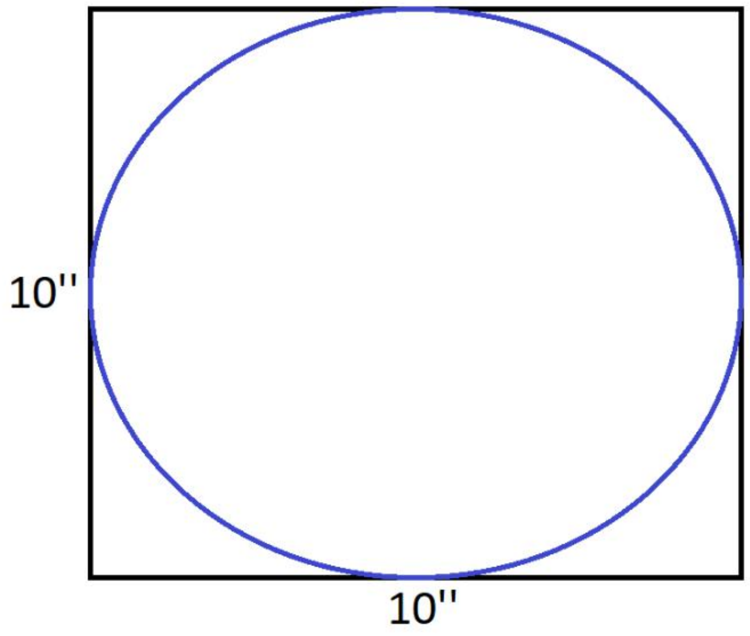
Saurabh Mandaokar(193100081)

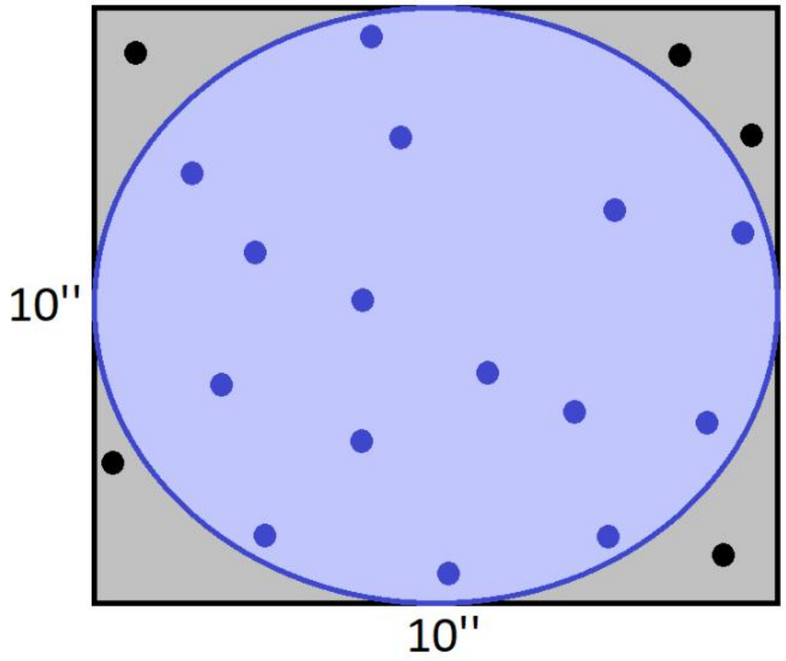
Problem Statement

- The calculation of area under a specific curve is very important for modern day problems.
- Generally integration is performed but, numerical methods are preferred in the absence of closed formed integrals.
- It is not very easy to perform integration computationally.
- Prior knowledge of a numerical method is necessary for numerical methods.
- We have decided to use the same principle that Monte-Carlo(MC) simulation works with, in order to calculate the area under a given curve

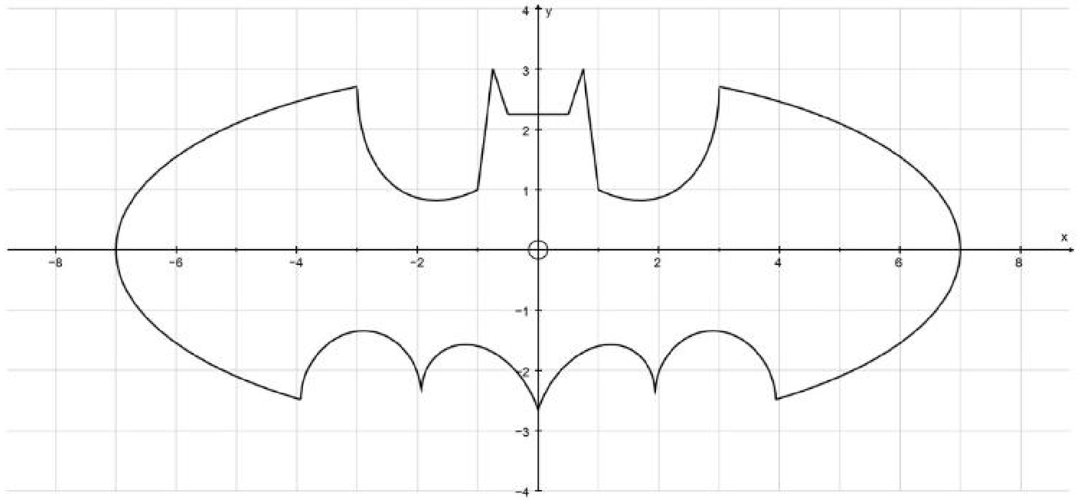
Methodology

- Monte-Carlo(MC) is just a way for estimating parametrs by generating random numbers.
- We will utilize the same concept to calculate the area of a curve.
- For example the curve shown here.
- The curve is enclosed in a rectangular box.
- Random points are generated.
- The total area of the box is calculated.
- Only points which fall within the curve are considered.





- The ratio of the valid points is calculated.
- The same ratio of the area of the box is considered.
- The method works well with curves that are not easy to depict mathematically.
- For example the image shown here.



Implementation

- We have considered two senarios.
  - One, where the curve is open and area is calculated within a range
  - Second, a simple closed aurve is considered.
- For both the cases random numbers are generated.
- For the first case, it is checked whether the geerated point lies between the curve and x-axis.
- For the second case, it is checked whether the point lies within the closed curve.
- The number of points generated is a hyper-parameter and can be tuned appropriately.

- For open curves the following method the code is as shown here.
- The code shown here is for the segregation of the points as valid and invalid.
- The generated random numbers are uniformly distributed.
- Hyper-parameter tuning was tested by vaying the number of points generated and results were obtained accordingly.

```
arr_in = []
points_in = 0
x_in, y_in = [],[]
for i in arr:
    if i[1] >= 0:
        if i[1] <= f(i[0]):
            arr_in.append(i)
            x_in.append(i[0])
            y_in.append(i[1])
            points_in += 1
    else:
        if i[1] >= f(i[0]):
            arr_in.append(i)
            x_in.append(i[0])
            y_in.append(i[1])
            points_in += 1

plt.plot(x_in,y_in,'*')
plt.plot(x,y,'r')
plt.grid()

area = (points_in/n) * box_area
rel_err = ((area/actual(a,b)) - 1) * 100
print(f' box_area : {box_area} \n enclosed area : {area} \n required area {actual(a,b)} \n\n points inside: {points_in} \n total
print(f'relative error is {rel_err} %')
```

- For closed curves the segregation is a bit simpler.
- We just check whether the point is inside the curve or not.
- Limits of the curve must be metioned.
- For both of the methods we have considered some predefined functions.

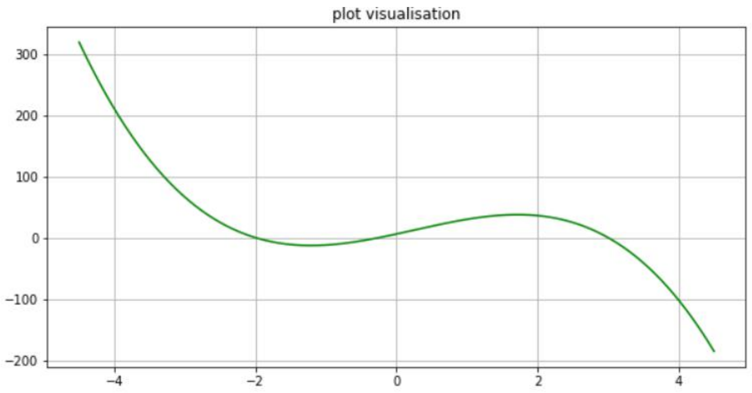
```
def closed(x,y):
    return((x**2)+(y**2)-9)
    #return(((x**2)/4)+(y**2)-1)
```

```
x_lim1,x_lim2 = -4,4
y_lim1,y_lim2 = -4,4

x_gen_c = np.sort(np.random.uniform(-4,4,100))
y_gen_c = np.random.uniform(-4,4,100)

arr = []
for i in range(len(x_gen_c)):
    arr.append([x_gen_c[i],y_gen_c[i]])

points=0
x1_c,y1_c=[],[]
for i in arr:
    if closed(i[0],i[1])<=0:
        x1_c.append(i[0])
        y1_c.append(i[1])
        points+=1
```

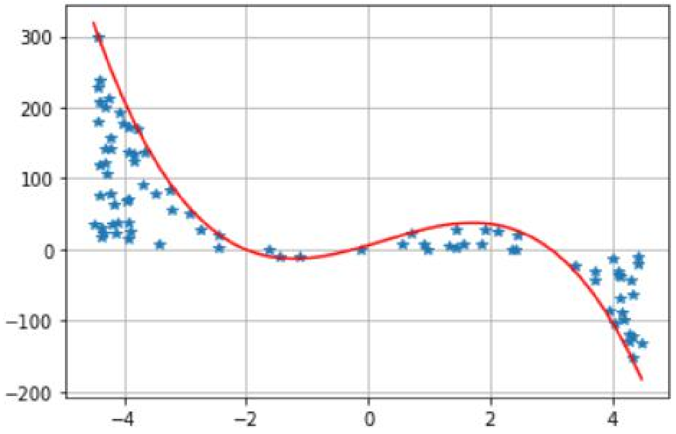
```
box_area_c = (x_lim2-x_lim1)*(y_lim2-y_lim1)
```

```
area_c = points*(box_area_c)/(len(x_gen_c))
```

Outputs



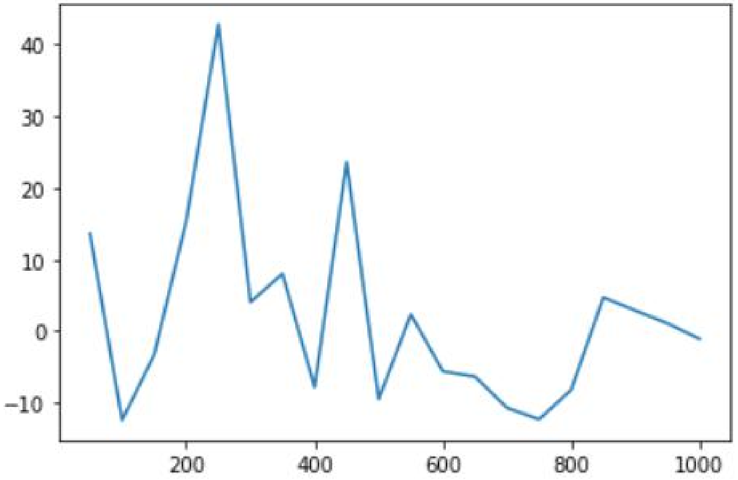
Original Curve



The selected points

```
box_area : 4497.106525534883
enclosed area : 472.1961851811627
required area 496.5287639111765

points inside: 84
total points = 800
```

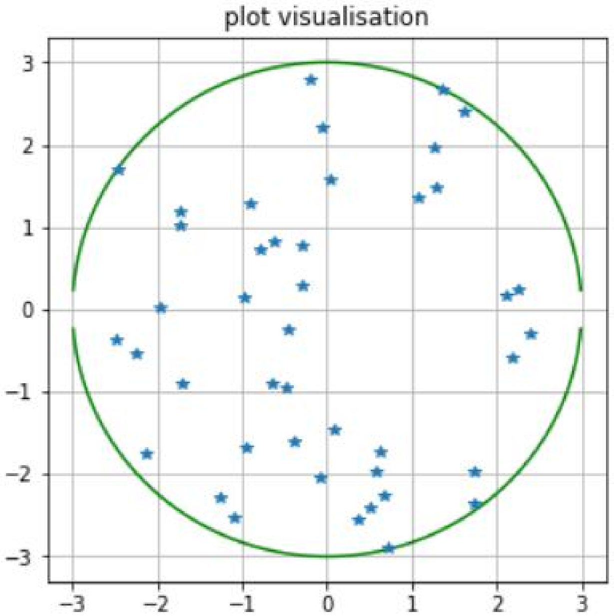
The answer we get after running the code for open curves.



Variation relative error with no. of random numbers

```
average area is 519.6582099889805
required area is 496.5287639111765

relative error for avg. area is 4.65822883967737
```

When the avg. of all those areas is calculated.



Closed curve

```
actual area of circle:  28.274333882308138
area calculated by us:  29.44
```

The ans. for a circle when we applied the method for closed curves.

Observations

- We saw that just increasing the number of points doesn't increase the accuracy of the problem.
- As the number of points are randomly generated the area calculated by the method varies.
- The average of several calculations gets us closer to the actual area.
- No prior knowledge of formulas were required to implement the code.
- The generated points are not supervised. Just the limits and distribution.
- The code doesn't require any special intstructions.
- The functions for which the area is to be calculated can be selected from the function 'f'.
- For closed curves only circle is considered and the area calculation for comparison is done using the standard formula, though ellipse was also tried and similar results were obtained.
- For open curves reimann's numerical method is used for comparison.