International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

# An Overview of Applications of Python Programming in Engineering: Tracing of Polar Curves

**G. Gomathi Jawahar[1], K. Lakshmi Priya.[2], R. Robin[3], A. Prabhat[4], Navaneetha Krishnanan[5]**

**Submitted**: 09/05/2023 **Revised**: 18/07/2023 **Accepted**: 09/08/2023

**Abstract:** In this article, the Python programming language is used to trace polar curves and compute definite integrals. Here the Python Programme is implemented to evaluate the area and necessary SciPy and Matplotlib libraries used to trace polar curves. A step-by-step explanation on how to design a polar curve function, generate an array of angles, and use the 'plt.polar()' function to plot the curve is shown. The article also describes how to use the SciPy library's 'quad()' function to determine the polar curves.

***Keywords:*** *Python, trace, polarcurves, SciPy, 'plt.polar(), 'quad()'*

## 1.Introduction

Python functions and modules can be used to determine the area and volume of the specified surfaces. For instance, the user can load the math module and apply the formula to determine the area and volume of a circle, Python can be a useful tool for calculating the area and volume of shapes and objects since it enables rapid computations that need little code.Area and volume computations can be made using Python's built-in mathematical functions, which include square root and exponentiation.Additionally, Python supports a wide range of additional libraries, like NumPy and SciPy, which offer more sophisticated mathematical Operations for manipulating intricate shapes and objects. The python Programme have a wide range of uses in a variety of disciplines, such as mathematics, physics, engineering, economics, and technology. In addition to many other tasks, they are used to compute areas and volumes, identify the average value of a function, calculate trip distance, assess probabilities, compute work done by a force, examine population growth, solve differential equations.

The use of derivatives and integral calculus as mathematical techniques can offer answers to the problems raised by the various techniques.The computation and visualisation of geometric attributes can also be facilitated by the use of Python libraries like NumPy, SciPy, and Matplotlib.These libraries offer a wide variety of mathematical tools and functions that can be used to make 2D and 3D visualisations of geometric shapes as well as to simplify complicated calculation.

## 2. Problem Statement

Tracing a specified polar curve and determining its definite integral using Python might be the problem statement for this topic. This requires using the SciPy and Matplotlib libraries to plot the curve and calculate the definite integrals, as well as understanding the concept of polar coordinates and how to convert them to Cartesian coordinates. The objective is to precisely draw the curve and compute the definite integral using Python, and the challenge could involve many polar curves with varied degrees of complexity. For those learning about polar coordinates, drawing curves, or computing definite integrals using the Python programming language, this problem statement is appropriate. Here the solution should be accurate and efficient, taking into account any relevant mathematical formulas and units of measurement.The code should also be well-organized,documented,and easy to understand,allowing for easy modification and re use in future projects.

## 3. Methodology

However, the following can be said about a broad methodology:

1. Determine whatever shape or item needs to have its area and volume calculated.

2. Identify the form or object's pertinent dimensions, such as its length, width, height, radius, etc.

3. Identify the mathematical formulae or procedures required to compute the shape's area and volume.

[1,2,3,4,5]*Karunya Institute of Technology and Sciences,Coimbatore.*
*ORCID ID :0000-0002-0361-7116*
** Corresponding Author Email: gomathi@karunya.edu*

4. Utilising the recognised formulas or equations, write Python code to carry out the required calculations.

5. Use known numbers or mathematical proofs to confirm that the estimated results are accurate.

6. Verify the computations' units of measurement are acceptable and consistent.

7. Include any assumptions or restrictions that were made during the computations in the documentation of the code and findings.

8. If necessary, modify the code and calculations to improve accuracy or performance.

Additional steps, such as creating unique functions or importing external libraries for specialised computations, may be necessary depending on the particular form or item being taken into account.

**4. Implementation**

Program Tofindthearea of the curve $y=x^2$

```
importmath

#Define the function representing the curvedeff(x):

return x**2

# Define the range of x values for which to compute the area and volume a =0  , b =1

# Define the number of subdivisions (the higher the value, the more accurate theresult)

n =1000

# Define the width of each subdivisiondx =(b-a)/n

# Compute the area of the surface of revolutionarea=0

foriin range(n):

  x =a +i* dx

area += 2 * math.pi * x * math.sqrt(1 + (f(x) ** 2)) * dx#Computethe volumeof the surfaceofrevolution

volume =0

foriin range(n):

  x =a +i* dx

volume += math.pi * (f(x) ** 2) * dx# Printthe results

print("Area of surface of revolution: ", area)print("Volumeofsurfaceofrevolution:",volume)
```

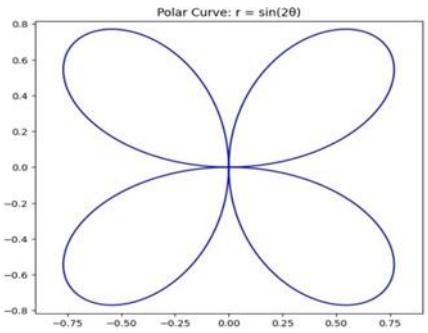
**OUTPUT**

```
Area of surface of revolution:  3.601457936527498
Volume of surface of revolution:  0.6267487815886101
>
```

**5. Tracing Polar Curves**

## 1) SIN(2θ)

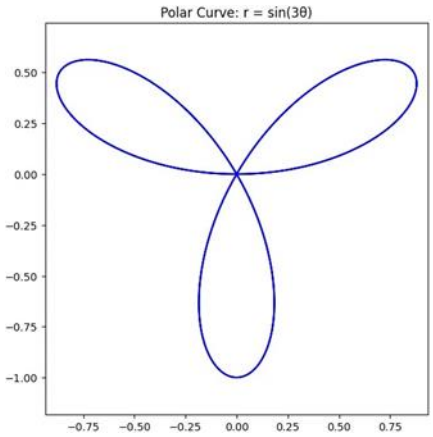
```
importnumpy asnp

import matplotlib.pyplot as pltdefpolar_func(theta):

r= np.sin(2*theta)returnr

theta_vals=np.linspace(0,2*np.pi, 1000)r_vals=polar_func(theta_vals)

x_vals= r_vals *np.cos(theta_vals)y_vals= r_vals * np.sin(theta_vals)plt.figure(figsize=(8,8))plt.plot(x_vals, y_vals, color='blue')plt.axis('equal')

plt.title('Polar Curve: r = sin(2θ)')plt.show()
```




**2)SIN(3θ)**

```
importnumpy asnp

import matplotlib.pyplot as pltdefpolar_func(theta):

 = np.sin(3*theta)returnr

theta_vals=np.linspace(0,2*np.pi, 1000)r_vals=polar_func(theta_vals)

x_vals = r_vals * np.cos(theta_vals)y_vals = r_vals * np.sin(theta_vals)plt.figure(figsize=(8,8))plt.plot(x_vals, y_vals, color='blue')plt.axis('equal')

plt.title('PolarCurve:r=sin(3θ)')

plt.show()
```

**OUTPUT**




**3) COS(2θ)**

importnumpy asnp

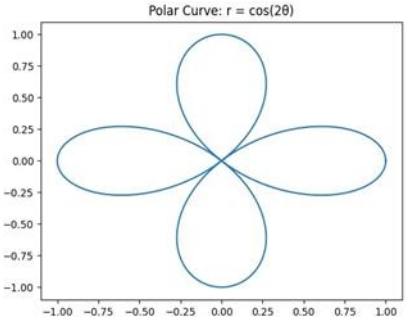
import matplotlib.pyplot as pltdefpolar_equation(theta):

r= np.cos(2*theta)returnr

theta_values=np.linspace(0,2*np.pi, 1000)

x_values = [polar_equation(theta) * np.cos(theta) for theta in theta_values]y_values = [polar_equation(theta) * np.sin(theta) for theta in theta_values]plt.plot(x_values,y_values)

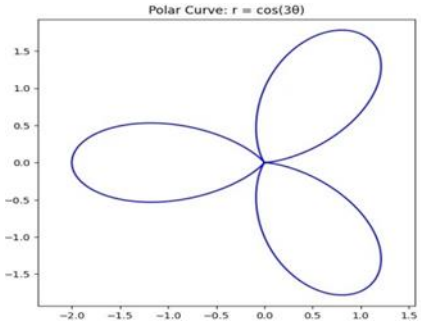
plt.show()




theta_vals = np.linspace(0, 2*np.pi, 1000)r_vals=polar_func(theta_vals)

x_vals = r_vals * np.cos(theta_vals)y_vals = r_vals * np.sin(theta_vals)plt.figure(figsize=(8,8))plt.plot(x_vals, y_vals, color='blue')plt.axis('equal')

plt.title('Polar Curve: r = cos(3θ)')plt.show()




**4)COS(θ)**

importnumpy asnp

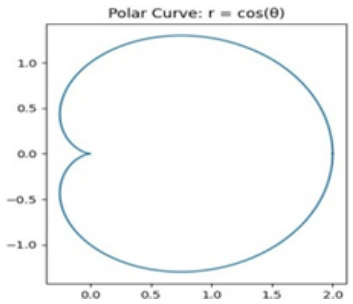
importmatplotlib.pyplotasplt

theta = np.linspace(0, 2*np.pi, 1000)r=1 +np.cos(theta)

x = r*np.cos(theta)y = r*np.sin(theta)plt.plot(x,y)

plt.gca().set_aspect('equal')plt.title("Polar Curve: r = cos(θ)")plt.show()

Output




## 6. Conclusion

In conclusion, the Python programme is implemented to evaluate the area and necessary SciPy and Matplotlib libraries used to trace polar curves. Here python Programme is applied to trace the curves $\sin 2\theta$, $\sin 3\theta$, $\cos 2\theta$, $\cos\theta$, A step-by-step explanation on how to design a polar curve function, generate an array of angles, and use the 'plt.polar()' function to plot the curve is shown. The article also describes how to use the SciPy library's 'quad()' function to determine the polar curves.

**References:**


[1] A. Meurer et al., "SymPy: Symbolic computing in python," PeerJ, 2017, doi: 10.7287/peerj.preprints.2083

[2] R. A. McCleery, A. Sovie, R. N. Reed, M. W. Cunningham, M. E. Hunter, and K. M. Hart, "Marsh rabbit mortalities tie pythons to the precipitous decline of mammals in the Everglades," Proc. R. Soc. B Biol. Sci., 2015, doi: 10.1098/rspb.2015.0120.

[3] S.Van Der Walt et al., "Scikit-image: Image processing in python," PeerJ, 2014, doi: 10.7717/peerj.453.

[4] J. Demšar et al., "Orange: Data mining toolbox in python," J. Mach. Learn. Res., 2013.

[5] A. Gramfort et al., "MEG and EEG data analysis with MNE-Python," Front. Neurosci., 2013, doi: 10.3389/fnins.2013.00267.

[6] D. Stanikova et al., "Python Cookbook," Carcinogenesis, 2012, doi: 10.1093/carcin/bgs090

[7] F. Pedregosa et al., "Scikit-learn: Machine learning in Python," J. Mach. Learn. Res., 2011.

[8] D. Kuhlman, "A Python Book: Beginning Python, Advanced Python, and Python Exercises," A Python B.,2009.

[9] Sashank, Y. T. ., Kakulapati, V. ., & Bhutada, S. . (2023). Student Engagement Prediction in Online Session. International Journal on Recent and Innovation Trends in Computing and Communication, 11(2), 43–47. https://doi.org/10.17762/ijritcc.v11i2.6108

[10] Alejandro Garcia, Machine Learning for Customer Segmentation and Targeted Marketing , Machine Learning Applications Conference Proceedings, Vol 3 2023.

[11] Pandey, J.K., Ahamad, S., Veeraiah, V., Adil, N., Dhabliya, D., Koujalagi, A., Gupta, A. Impact of call drop ratio over 5G network (2023) Innovative Smart Materials Used in Wireless Communication Technology, pp. 201-224.