International Journal on Recent and Innovation Trends in Computing and Communication ISSN: 2321-8169
Volume: 5 Issue: 3 141 – 143

Study of Kinematic Analysis Using Python Programming Language: A Visualization Approach

Dr. Nirmal Sahuji
Department of Physics
Vidya pratishthan's Kamalnayan Bajaj Institute of Engineering and Technology,
MIDC, Baramati 413133, India
Email: sahujink@gmail.com

Abstract: Kinematics as a field of study is often referred to as the geometry of motion. To describe motion, kinematics studies the trajectories of points, lines and other geometric objects and their differential properties such as velocity and acceleration to investigate the variety of means by which the motion of objects can be described. The variety of representations that one can investigate includes verbal representations, numerical representations, and graphical representations. In planar mechanisms, kinematic analysis can be performed either analytically or graphically. In this paper we first discuss analytical kinematic analysis.

In the present paper, graphical representation of kinematical equation is studied. Analytical kinematics is a systematic process that is most suitable for developing into a computer program. Plotting few points does not provide us with a complete picture of the distance traveled over time. For this we need more points. Program written in Python calculates as many points as required almost instantaneously and help to see the better perspective of the conceptual language.

Keywords- *kinematics, visualization, trajectories of points*

*****

I. INTRODUCTION

In scientific application, data is represented as vectors or variables. To perform such task, a computer language needs fast and efficient array storage, retrieval and operations [1]. Moreover interaction of user with application is desired through GUI. Cost effective end product is also one of the biggest challenge. One of the fast upcoming language which supports all this feature is Python. Python is widely used in scientific and numeric computing.

As Python is a popular general-purpose programming language, it has many advanced modules for building interactive applications. Numpy provides powerful numerical arrays objects, and routines to manipulate them.

Similarly,modules SciPy using which it become possible to build a full-fledged scientific application. Scipy is a high-level data processing routines used for Optimization, regression, interpolation, etc

Another feature is IPython a powerful interactive shell that features easy editing and recording of a work session, and supports visualizations and parallel computing. A package Matplotlib supports 2-D visualization and Mayavi supports 3-D visualization.

It is often required to reduce complexity of the program and to focus more on physics rather than computer programming. Using control structure tends the user to know computer programming. However, rich python libraries help to reduce this burden [2].

II. COURSEWORK

The kinematic equations are a set of four equations that can be utilized to predict unknown information about an object's motion if other information is known. The equations can be utilized for any motion that can be described as being either a constant velocity motion (an acceleration of 0 m/s/s) or a constant acceleration motion. They can never be used over any time period during which the acceleration is changing. Each of the kinematic equations include four variables. If the values of three of the four variables are known, then the value of the fourth variable can be calculated. In this manner, the kinematic equations provide a useful means of predicting information about an object's motion if other information is known. For example, if the acceleration value and the initial and final velocity values of a skidding car is known, then the displacement of the car and the time can be predicted using the kinematic equations. The four kinematic equations that describe an object's motion are:

$$\mathbf{d = v_i.t + \frac{1}{2} a.t^2}$$

$$\mathbf{d = v.t + 0.5.a.t^2}$$

$$\mathbf{v_f^2 = v_i^2 + 2.a.d}$$

 ISSN: 2321-8169
141 – 143

$$v_f = v_i + a.\ t$$

$$d = \frac{vi+vf}{2}.\ t$$

The symbol d stands for the displacement of the object. The symbol t stands for the time for which the object moved. The symbol a stands for the acceleration of the object and the symbol v stands for the velocity of the object; a subscript of i after the v (as in vi) indicates that the velocity value is the initial velocity value and a subscript of f (as in vf) indicates that the velocity value is the final velocity value.

Each of these four equations appropriately describes the mathematical relationship between the parameters of an object's motion. As such, they can be used to predict unknown information about an object's motion if other information is known [3].

In this example students use the equation to calculate the distance traveled by an object projected at a given initial velocity under the gravitational force on earth. Here, a Python program is used to quickly calculate the distance traveled every second for twenty seconds and capture the results in a file to facilitate graphing. Students can then run the program using the force of gravity from other planets as well, and see how this affects the shape of the graph.
The equations used to describe the relationship between distance, time, velocity, and acceleration in moving objects are called kinematic equations.distance=initial_velocity*time+0.5*acceleration*time^2

The study of kinematics is based on purely mathematical functions [4]. For instance, rotation can be represented by elements of the circle in the given plane. Other algebras are used to represent the shear mapping of classical motion in absolute time and space and to represent the Lorentz transformations of relativistic space and time. By using time as a parameter in geometry, mathematicians have developed a science of kinematic geometry. Kinematic analysis is the process of measuring the kinematic quantities used to describe motion. In engineering, for instance, kinematic analysis may be used to find the range of movement for a given mechanism, and, working in reverse kinematic synthesis designs a mechanism for a desired range of motion.

With manual calculations, we can eaily get data as

- distance traveled after 1 second = 75.1 meters
- distance traveled after 2 seconds = 140.4 meters
- distance traveled after 3 seconds = 195.9 meter

Plotting with only few point does not provide us with a complete picture of the distance traveled over time. For this we need more points. Hence, we can program Python to calculate as many points as we want almost instantaneously.

```
def Distance(init_velocity, end_time, accel):
    "calculation for distance traveled at every second"
    time = 0
    while time <= end_time:
        d = init_velocity * time +.5 * accel * time**2
        print time,d
        time += 1

Distance(80,25,-9.8)
```

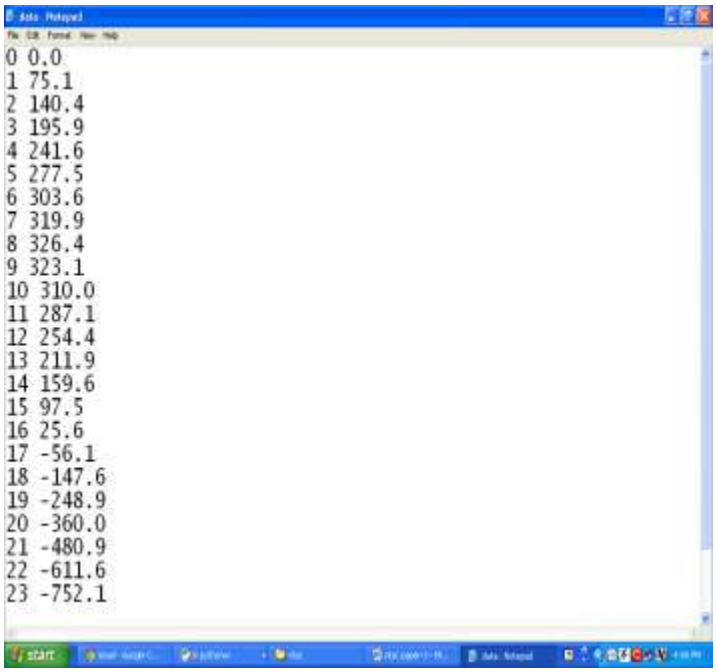

Fig.1: Data captured in text file: data.txt

Following program helps to plot graph.

```
from pylab import plot,show
from numpy import loadtxt
import matplotlib.pyplot as plt
data = loadtxt("data.txt",float)
x = data[:,0]
y = data[:,1]
plt.xlabel('Time in second')
plt.ylabel('Distance')
plt.title('distance-time mapping of the ball')
plot(x,y)
show()
```

 ISSN: 2321-8169
141 – 143

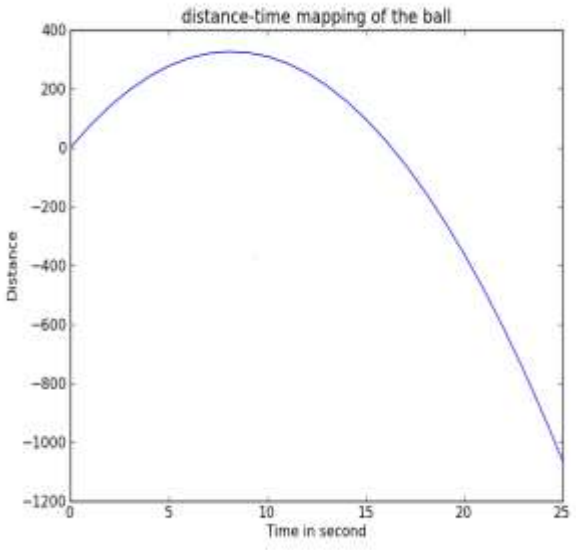

Fig: Pattern of distance travelled by the ball at every second.

III. CONCLUSION

The motion of the ball along with its trajectory is obtained. The visualization clearly indicates how far the ball travels. It gives complete picture of the distance traveled by projectile over time.

IV. ACKNOWLEDGEMENT

Author is grateful to the Principal, Vidya Pratishthan;s Kamalnayan Bajaj Institute of Engineering and Technology, Baramati for his constant support and encouragement.

REFERENCES

[1] Ting-Sheng Weng, Meng-Hui Hsu and Der-Ching Yang, "Dynamic teaching of kinematics of particles and python," International Journal of e-Education, e-Business, e-Management and e-Learning, Vol. 3, No. 4, August 2013.

[2] P. J. Cornwell, "Teaching Dynamics Using Modern Tools," Computers in Education Journal, Oct.-Dec. 1996.

[3] Arnd Backer, "Computational physics education with python," Computing in science and engineering, vol. 9, pp. 30-33, June 2007.

[4] Fernando Perez, Brian E. Granger, "IPython : A system for interactive computing," Computing in science and engineering, vol. 9, pp. June 2007.