e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Volume:06/Issue:05/May-2024 Impact Factor- 7.868 www.irjmets.com

ENERGY CONSERVATION FOR THE PROJECTILE MOTION PROBLEMS: COMPREHENSIVE STUDY AND ANALYSIS

Avni Yadav*1, S.K. Chaturvedi*2

*1Student, Amity International School, Abu Dhabi, United Arab Emirates.

*2Professor, Department Of Aerospace Engineering, Dehradun, India.

DOI : https://www.doi.org/10.56726/IRJMETS56584

ABSTRACT

Energy can neither be created nor can it be destroyed. It can only change from one form to another. This paper examines if the law of conservation of energy holds true in projectile motion.

With advances in computing, we do not require artillery guns and sophisticated equipment for testing these concepts now. A Python-based computer simulation coded with the appropriate rules and set to accept the correct parameters (Mass of the object, Angle of Projection, and Initial Velocity) can conduct an infinite number of experiments and simplify reporting. Through this paper, we test the law of conservation of energy at different points along the projectile's trajectory and for three different parameters namely Mass of the object, Angle of Projection, and Initial Velocity and present our findings.

Keywords: Energy Conservation, Simulation, Calculation, Program, Projectile Motion, Mass Of The Object, Angle Of Projection, Initial Velocity.

I. INTRODUCTION

Our society's progression thrives on advancements in science, technology and engineering, all rooted in research and scientific knowledge. The unpredictable nature of technological progress underscores the importance of physics, which influences research and development across natural sciences, biology and chemistry. Physics permeates educational systems globally and shapes our everyday experiences; hence, anyone exploring the physical world and the natural phenomena can be regarded as a physicist [1]. The essence of physics is evident in the diverse manifestations of motion observed in the universe—be it fascinating, violent, or beautiful. Motion, a fundamental characteristic, facilitates the comprehensive study of the universe. The study of motion, known as physics, was recognized as early as the fifth century BCE in ancient Greece [2].

Many students encounter difficulties in conceptualizing and mastering physical principles. This paper aims to present a practical approach to enhance comprehension of Newtonian mechanics, specifically projectile motion. Projectile motion serves as a fundamental example of two-dimensional motion extensively explored by researchers. Traditional treatments of projectile motion, devoid of aerodynamic drag, utilize differential and integral calculus or algebra and trigonometry [3]. Consider a ball thrown into the air at an angle: it ascends to its apex before descending, while simultaneously moving horizontally away from the point of release. A fundamental principle dictates that an object's horizontal displacement can be determined without accounting for its vertical motion. By decomposing vector quantities—such as acceleration, velocity, and position—into orthogonal components, variations in magnitude along specific axes are apparent. Upon release, the ball's motion can be segmented into its upward and forward components: the upward velocity constituent governs its ascent and descent, while the forward velocity component dictates its trajectory [4]. However, it is not intuitive that the energy will remain constant throughout the trajectory. Through our simulation, we will calculate the kinetic and potential energy at different points of the projectile's trajectory as well as examine the effect of Mass of the object, Angle of Projection, and Initial Velocity.

II. METHODOLOGY

Computational aspects of Simulation

The simulation has three codes for different case studies relating to projectile motion to calculate different values of an object in a projectile motion. The codes calculate velocities in the vertical and horizontal directions, height, time, trajectory, range and energy using different functions imported from the math library following the physics equations relating to each different value. To graph the motion of the three objects the trajectory is first calculated with the trajectory equation using values of angle of projection, gravitational potential constant and the initial velocity that are either calculated through the code or are inputted by the user. The codes plot the graphs, label axis, set a limit to axis and add a title to the graph plot using the functions imported from the matplotlib.plot library. The quadratic graphs are plotted evenly for energy and distance using the linspace and * function imported from the numpy library.

Physical aspects of Simulation

The first code calculates the initial velocities for the vertical and horizontal directions after the users input. After these velocities are calculated, the code calculates the height using the $v^2 = u^2 + 2as$ equation. The time is calculated with the $v = u + at$ equation. The horizontal distance is then calculated with the equation $s = ut$. To graph the projection of the object case study, the trajectory can be calculated with splitting the x^2 and the x value into 2 variables to be calculated.

The second example object code calculates the initial velocity using the $R = \frac{v_0^2 sinsin\ 2\ \theta_0}{g}$ equation for each ball. The code then calculates the trajectory with a variable calculating the x^2 and the x term value with the user inputs for the range and angle with the $y = x\left[tan\theta_0 - \frac{g}{2(v_0 cos\theta_0)^2}\ x\right]$equation.

The third example object code calculates the initial velocity in both directions, the height, Gravitational Potential Energy using the mgh equation and the Kinetic Energy using the $\frac{1}{2}mv^2$ equation at different times in the projectile using the inputs from the user. To plot the curves of energy against time, the suvat equations have been used to create quadratic functions for the GPE and KE.

$$GPE = \left(-\frac{1}{2}mg^2\right)t^2 + (mgu\ sin\ sin\ \theta\ )t$$

$$KE = \left(\frac{1}{2}mg^2\right)t^2 - (mug\ sin\ sin\ \theta\ )t + \left(\frac{1}{2}mu^2\right)$$

III. MODELING AND ANALYSIS

Code for projectile motion of a firework:

```
import math

# User Input for Initial Velocity, Angle that object is projected at and the Mass of object
u=int(input('Enter the initial velocity: '))
angle=int(input('Enter the angle of projection: '))
mass=int(input('Enter the mass of the object: '))

# Value of Gravity constant
g=9.8

# Calcute Initial Velocity in the vertical and horizontal vector
horizontal_velocity=round(u*math.cos(math.radians(angle)),2)
vertical_initial_velocity=round(u*math.sin(math.radians(angle)),2)
print('Initial Vertical velocity: ', vertical_initial_velocity, 'm/s')

#Calculate Height using v2=u2+2as equation
height =round(((vertical_initial_velocity**2)/(2*g)),2)
print('Height: ', height, 'm')

# Calculate time using v=u+at equation
t=round(((vertical_initial_velocity)/(g)),2)

# Calculate Horizontal distance using s=ut equation
xh=round((horizontal_velocity*t),2)
print('Horizontal velocity: ', horizontal_velocity, 'm/s')
print('Horizontal distance: ', xh, 'm')
print('The gravity constant: 9.81 m/s**')
```

e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Volume:06/Issue:05/May-2024 Impact Factor- 7.868 www.irjmets.com

```
# Calculate trajectory to plot as a graph for distance
trajectoryax=round(math.tan(math.radians(angle)),2)
trajectoryax2=round(g/(2*(((vertical_initial_velocity)*(math.cos(math.radians(angle))))**2)),5)

import matplotlib.pyplot as plt
from matplotlib.pyplot import *
from numpy import *

x=linspace(-1,xh,5000)
y=((-1*trajectoryax2)*x**2)+(trajectoryax*x)
plt.plot(x,y)
plt.xlabel("x (m)")
plt.ylabel("y (m)")
fig, ax=plt.subplots()
ax.set_xlim(xh/2)
plt.show()
```

Input:

```
Enter the initial velocity: 70
Enter the angle of projection: 75
Enter the mass of the object: 10
```

Output:

```
Initial Vertical velocity:  67.61 m/s
Height:  233.22 m
Horizontal velocity:  18.12 m/s
Horizontal distance:  125.03 m
The gravity constant: 9.81 m/s**
```

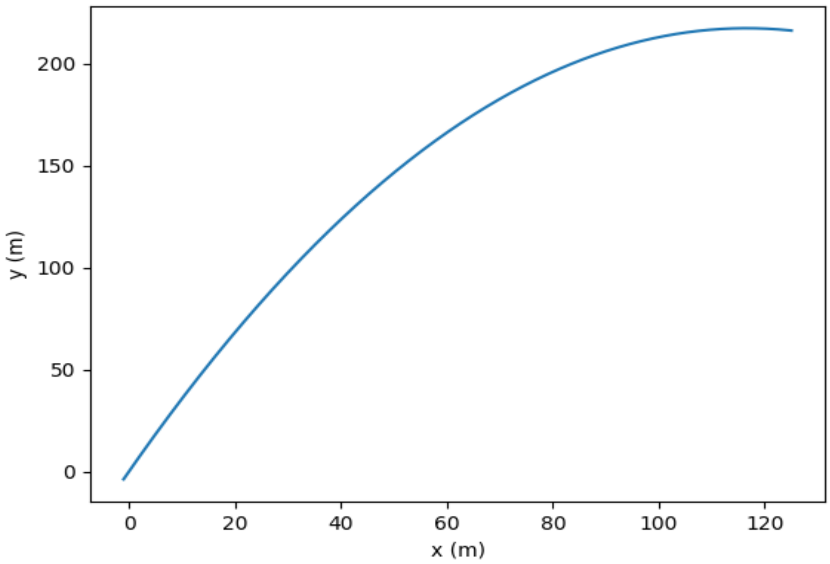

Figure 1: Projectile Study to calculate the x and y values. Initial velocity (70 m/s) and Angle (75°).

e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Volume:06/Issue:05/May-2024 Impact Factor- 7.868 www.irjmets.com

Code for projectile motion of two golf balls:

```
import math

# User Input for the range of both balls and the angle that ball is projected at
r=int(input('Enter the range: '))
r1=int(input('Enter the range: '))
anglea=int(input('Enter the angle: '))
anglea1=int(input('Enter the angle: '))

#Value for Gravity Constant
g=9.80

# Calculate Initial velocity of balls using Range equation
initial_velocity=round(math.sqrt((r*g)/(math.sin(math.radians(2*anglea)))),2)
initial_velocity1=round(math.sqrt((r1*g)/(math.sin(math.radians(2*anglea1)))),2)
print('Initial velocity of first: ', initial_velocity)
print('Initial velocity of second: ', initial_velocity1)

# Calculate trajectory using trajectory equation
trajectoryx=round(math.tan(math.radians(anglea)),2)
trajectoryx2=round(g/(2*(((initial_velocity)*(math.cos(math.radians(anglea))))**2)),5)
print('Trajectory of Secondhole: ', trajectoryx, 'x -', trajectoryx2, 'x2')
trajectoryx1=round(math.tan(math.radians(anglea1)),2)
trajectoryx21=round(g/(2*(((initial_velocity1)*(math.cos(math.radians(anglea1))))**2)),5)
print('Trajectory of Secondhole: ', trajectoryx1, 'x -', trajectoryx21, 'x2')
```

```
import matplotlib.pyplot as plt
from matplotlib.pyplot import *
from numpy import *

x=linspace(-1,r,5000)
y=((-1*trajectoryx2)*x**2)+(trajectoryx*x)
y1=((-1*trajectoryx21)*x**2)+(trajectoryx1*x)
plt.plot(x,y)
plt.plot(x,y1)
plt.xlabel("x (m)")
plt.ylabel("y (m)")
plt.title('Golf Ball')
fig, ax=plt.subplots()
ax.set_ylim(0)
plt.show()
```

Input:

```
Enter the range: 90
Enter the range: 90
Enter the angle: 30
Enter the angle: 70
```

Output:

```
Initial velocity of first:  31.91
Initial velocity of second:  37.04
Trajectory of Secondhole:  0.58 x - 0.0064 x2
Trajectory of Secondhole:  2.75 x - 0.03053 x2
```

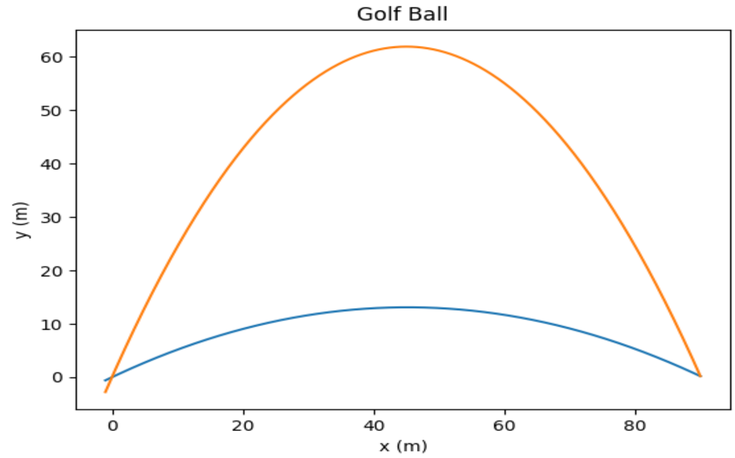

Figure 2: Projectile Study to calculate x and y values for different angles. Range (90m, 90m) and Angle (30°, 70°)

Code for the energies of an object along the projectile motion:

```
import math
from matplotlib import pyplot as plt

# User Input for Initial Velocity, Angle that object is projected at and the Mass of object
u=int(input('Enter the initial velocity: '))
angle=(int(input('Enter the angle of projection: ')))
mass=int(input('Enter the mass of the object: '))

# Value of Gravity constant
g=9.81

# Calcute Initial Velocity in the vertical and horizontal vector
horizontal_velocity=u*math.cos(math.radians(angle))
vertical_initial_velocity=u*math.sin(math.radians(angle))

# Calculate time using v=u+at equation
t=int(2*vertical_initial_velocity)/(g)
height =(vertical_initial_velocity*t)-(0.5*g*(t*t))
print('Height: ', height)
```

```
import matplotlib.pyplot as plt
from matplotlib.pyplot import *
from numpy import *

plt.xlabel('Time (s)')
plt.ylabel('Energy (J)')
plt.title('Projectile Motion Conservation of Energy')
x=linspace(0,t,5000)
y=((0.5*mass*g*g)*x**2)-((mass*u*g*(vertical_initial_velocity/u))*x)+(0.5*mass*u*u)
y1=(-1*(0.5*mass*g*g)*x**2)+((mass*g*u*(vertical_initial_velocity/u))*x)
initial=0.5*mass*u*u
plt.axhline(y=initial, xmin=0.048, color='green', label='Total Energy')
plt.plot(x,y, label='KE')
plt.plot(x,y1, label='GPE')
plt.legend()
fig, ax=plt.subplots()
ax.set_ylim(0)
plt.show()
```

Input:

```
Enter the initial velocity: 70
Enter the angle of projection: 75
Enter the mass of the object: 100
```

IV. RESULTS AND ANALYSIS

The first two examples show a simulation of motion for a firework and two golf balls. Their path is outputted through the code which displays the motion in x and y components. Taking the same Initial Velocity (70 m/s), Angle (75°) and a Mass of (100 kg), the code then calculates and graphs the Gravitational Potential Energy (Orange curve) and the Kinetic Energy (Blue curve).

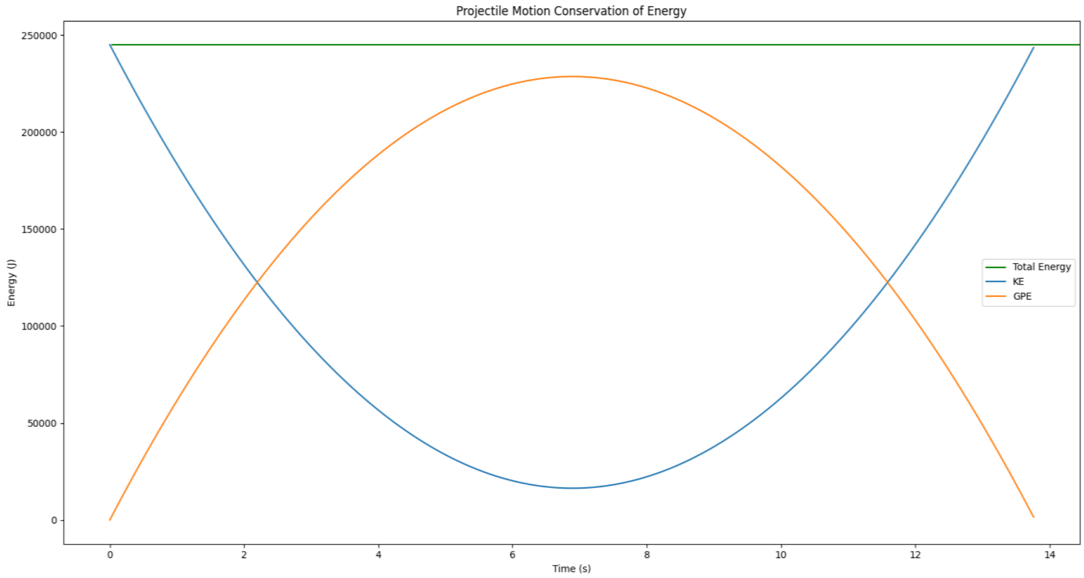

Figure 3: Projectile Study for Gravitational Potential Energy, Kinetic Energy and Total Energy for a projectile motion. Initial Velocity (70 m/s), Mass (100 kg) and Angle (75).

The graph for the Gravitational Potential Energy increases from 0 at the ground level to the maximum value of 245 kJ at the highest point and then decreases back to 0. The Kinetic Energy decreases from maximum energy of 245 kJ at ground level to approximately 16 kJ at the highest point and then increases back to 245 kJ. The point of intersection of energies is approximately at 125 kJ. The Total Energy remains constant at 245 kJ as represented by the horizontal line when adding Gravitational Potential Energy and Kinetic Energy thus proving that energy is conserved.

Energies for Mass = 100 kg, Initial Velocity = 70 m/s, Angle = 75° (Corresponding to Figure 3)

	Gravitational Potential Energy (J)	Kinetic Energy (J)	Total Energy (J)
At launch	-	245,000	245,000
50% height	114,294	130,706	245,000
Max height	228,588	16,412	245,000
50% height	114,294	130,706	245,000
Landing Point	-	245,000	245,000

Energies for Mass = 50 kg, Initial Velocity = 70 m/s, Angle = 75°

	Gravitational Potential Energy (J)	Kinetic Energy (J)	Total Energy (J)
At launch	-	122,500	122,500
50% height	57,147	65,353	122,500
Max height	114,294	8,206	122,500
50% height	57,147	65,353	122,500
Landing Point	-	122,500	122,500

Energies for Mass = 100 kg, Initial Velocity = 35 m/s, Angle = 75°

	Gravitational Potential Energy (J)	Kinetic Energy (J)	Total Energy (J)
At launch	-	61,250	61,250
50% height	28,574	32,676	61,250
Max height	57,147	4,103	61,250
50% height	28,574	32,676	61,250
Landing Point	-	61,250	61,250

Energies for Mass = 100 kg, Initial Velocity = 70 m/s, Angle = 37.5°

	Gravitational Potential Energy (J)	Kinetic Energy (J)	Total Energy (J)
At launch	-	245,000	245,000
50% height	45,397	199,603	245,000
Max height	90,795	154,205	245,000
50% height	45,397	199,603	245,000
Landing Point	-	245,000	245,000

The tables prove the Energy is conserved for any value of Mass, Initial Velocity and Angle of Projection.

V. CONCLUSION

The graph for Gravitational Potential Energy and Kinetic Energy shows a simulation of how energies change during the projectile path. This research proves that Energy is conserved for a projectile motion through computational calculations and through a graph to visually see Energy Conservation outputted through a computer program. This follows the law of conservation stating that 'Energy can neither be created nor destroyed'.

ACKNOWLEDGEMENTS

I want to express my sincere gratitude to Dr. Sudhir Kumar Chaturvedi as he has taught me many valuable points related to Physics and enabled enrichment in my academic journey beyond the school curriculum.

VI. REFERENCES

[1] Schiller C. Motion Mountain-vol. 1-Fall, Flow and Heat-The Adventure of Physics.` Christoph Schiller; 2013.

[2] Calderon CT, Mohazzabi P. Average speed in projectile motion and in general motion of a particle. Journal of Applied Mathematics and Physics. 2018 Jul 27;6(07):1540.

[3] Bloomfield LA. How things work: the physics of everyday life. John Wiley & Sons; 2015 Dec 15.

[4] Adams S, Allday J. Advanced Physics. Oxford University Press; 2013. Serway RA, Jewett JW. Physics for scientists and engineers. Cengage learning; 2018.

[5] Cuoco AA, Goldenberg EP. A role for technology in mathematics education. Journal of Education. 1996 Apr;178(2):15-32.

[6] Chabay R, Sherwood B. Computational physics in the introductory calculus-based course.

[7] American Journal of Physics. 2008 Apr;76(4):307-13.

[8] Wells M, Hestenes D, Swackhamer G. A modeling method for high school physics instruction. American journal of physics. 1995 Jul;63(7):606-19.

[9] Landau ID, Zito G. Digital control systems: design, identification and implementation. Springer Science & Business Media; 2007 May 11.

[10] Snyder LG, Snyder MJ. Teaching critical thinking and problem solving skills. The Journal of Research in Business Education. 2008 Apr 1;50(2):90.

[11] AN, YJ, Haynes L, D'Alba A, Chumney F. Using educational computer games in the classroom: Science teachers' experiences, attitudes, perceptions, concerns, and support needs.

[12] Contemporary Issues in Technology and Teacher Education. 2016 Dec;16(4):415-33. Beichner R, Chabay R, Sherwood B. Labs for the Matter & Interactions curriculum. American Journal of Physics. 2010 May;78(5):456-60.

[13] Bufasi E, Lakrad K, Improving Teaching Techniques Using Visual Python: A Case Study In Physics Laboratories. International journal of scientific & technology research. 2019 Dec; 8(12).

[14] Chabay R, Sherwood B. Restructuring the introductory electricity and magnetism course. American Journal of Physics. 2006 Apr;74(4):329-36.

[15] Scherer D, Dubois P, Sherwood B. VPython: 3D interactive scientific graphics for students. Computing in Science & Engineering. 2000 Sep;2(5):56-62.

[16] Knight J. Science of Everyday Things-Real Life Physics, vol. 2. Thomson Learning, Michigan. 2002:45-50.Council NR, GS Committee. Learning to think spatially.

[17] Weintrop D, Beheshti E, Horn M, Orton K, Jona K, Trouille L, Wilensky U. Defining computational thinking for mathematics and science classrooms. Journal of Science Education and Technology. 2016 Feb 1;25(1):127-47.

[18] Salgado RB. VPython applications for Teaching Physics. American Astronomical Society Meeting Abstracts 2006 Dec (Vol. 209, pp. 60-02).