Numerical Simulations of Action Principle

Refath Bari

February 20, 2021

Abstract

The Principle of Least Action (PLA) can be confusing to students, in part due to the Calculus of Variations, but also because of the subtleties of the actual principle. Can there exist two local minimums? Do different observers in different IRFs (Inertial Reference Frames) calculate different actions? Is action a tangible entity like velocity? In an effort to address the questions and misconceptions that can arise from the Action principle, I develop three numerical simulations that put the PLA to use: light reflecting in equal angles, light refracting in different mediums, and light moving between two points in the least time.

1 Introduction

The Action Principle can be difficult for students, in particular because of its abstract nature and lack of exposure to students. The action itself could be hard to digest, given its intangible nature, unlike velocity or even acceleration which can be understood intuitively [2]. Students may be confused as to why the Lagrangian must be T-V instead of T+V, as mechanical energy is [4]. They may be confused by the abstraction of "nearby paths" and why light can't simply travel in straight lines to minimize distance [8]. Furthermore, the Calculus of Variations is not about minimizing a variable, but a function, which can be difficult for students to comprehend [3] — especially when applied to the nearby paths of the PLA. Students are not given adequate time to understand Variational Calculus before it's applied to Physics, as PLA and COV are often simultaneously introduced, exacerbating student confusions and misconceptions [5].

In light of these difficulties, numerical simulations may help students understand the PLA. Interactive Simulations have proven to be useful tools in teaching high school and undergraduate students Physics alike, particularly in distilling abstract ideas into tangible ones [13]. For instance, OpenRelativity [15], a product of MIT's GameLab, has done exactly so with Special Relativity and PhET's Quantum Mechanics simulations [16] have done so, as well. However, few simulations have accounted for the intersection in between — the PLA. The PLA motivates the Relativistic principle of following the world-line of maximal aging, as well as the Quantum Mechanical principle of exploring all paths. This paper takes a slightly different approach in that it encourages students to engage in the actual creation of the programs contained within, so that students are motivated to understand the mechanics of the PLA, hence discovering the Euler-Lagrange Equations while replicating the PLA.

The purpose of this paper is to present three numerical simulations of the Action Principle at work, all involving light. In doing so, students leave with a better intuitive understanding of the PLA. The first simulation is that of light reflecting in equal angles off a metal [11] [9]. The second simulation involves light refracting through different materials [12]. The final simulation involves the Brachistochrone problem, solved using Johann Bernoulli's application of Snell's Law to the problem, treating a particle as a beam of light [10]. Each of the visualizations is sufficiently interactive in nature, engaging the user through user input, mouse movement, and real-time plots.

2 Simulations

2.1 Reflection of Light

The first program shows all possible trajectories for light to travel from Point A to B if it must strike a metal surface.

The program consists of three main components: dragging, bouncing, and drawing. The first component is dragging, which enables the user to move two green balls that determine the initial and final position of the light beam. To enable this component, three functions must be defined, each linked to a mouse click event. One such event is shown below.

```
def move():
    global drag, point
    if drag and len(coords) <= 2:
        point.pos = scene.mouse.pos
    else:
        base.pos = scene.mouse.pos
scene.bind("mousemove", move)
```

The second component is bouncing. A hit point on the metal is determined, on which the light beam will strike. The directional vector from the point to the initial position directs the light beam on half its journey. To detect when the light beam hits the metal surface, the program checks whether the vertical position of the beam meets the center of mass of the metal added to its width. The directional vector is recalculated from the final position to the hit point to direct the light beam on the final half of its journey.

```
light.v=v0*norm(gpoint-p1.pos)
while light.pos.y>=(base.pos.y+.25):
        rate(200)
        light.pos=light.pos+light.v*dt
        action += dt
        t=t+dt
light.v=v0*norm(p2.pos-gpoint)
```

The third component is drawing. Drawing real-time plots of the light beam's movement is simple. Every plot involves three commands: a graph, a curve, and a plot. One such example for the Angle of Incidence versus the Light Time is shown below.

```
tgraph=graph(title = 'Time vs. Angle', xtitle="Angle [Deg]", ytitle="Light
↪  Time [s]", fast=False)
f1=gcurve(graph = tgraph, color=color.blue, markers=True,
↪  marker_color=color.blue, label = 'Incident', legend = True)
f1.plot(temp[1]*180/pi,temp[0])
```

The user is able to pick and drag the initial and final position of the beam of light. This interactive element reinforces the different initial values given for Classical Newtonian Mechanics rather than the Action Principle. In the former, we are given the initial position and velocity. In the latter, we are simply given boundary conditions — the endpoints of the light beam's trajectory. As all possible trajectories from A to B are traced out on the screen, real-time plots are generated indicating the relationship between the Incidence Angle, Reflection Angle, Light Time, and Action. This element enables the student to conjecture by inspection that the Action is minimized when the Light Time is minimized and the Reflection and Incidence Angles are equal.

2.2 Refraction of Light

The second program shows all possible trajectories for light to refract from Point A to B if it must strike the metal surface.

Similar to the first program, the second program also involves three components: dragging, refraction, and iteration. The first component is dragging, which enables the student to determine the light beam's initial position in the first material and final position in the second material. This involves defining a click event and a function linked to the event. One such event is shown below.

```
def down():
    global drag, point
    if len(coords) < 2:
        point = sphere(pos=scene.mouse.pos, color=color.red)
        coords.append(point.pos)
    drag = True
scene.bind("mousedown", down)
```

The second component is refraction, which adjusts the direction of the light beam such that it reaches the final point. To do so, the directional vector is recomputed, as is the new speed of light to account for the index of refraction.

```
light.v=v_new*norm(aim)
thetar=acos(light.v.y/v_new)

while light.pos.y < (glass.pos.y+5) and light.pos.y > p2.pos.y:
                          rate(200)
       light.pos=light.pos+light.v*dt
```

```
        t=t+dt
        return(t,thetai,thetar)
```

The third component is iteration, which involves incrementing the hit point on the surface of the second material by some constant, until all possible trajectories for the light beam are generated. To do so, a for loop asks the time function to increase the horizontal argument by some arbritrary constant. Note that as this constant approaches 0, the accuracy of the incidence and refraction time increases, but so does the computation time.

```
for i in range (1,12):
    temp=lighttime(vector((glass.pos.x-5)+x,glass.pos.y+5,glass.pos.z))
          print("Incident = ", temp[1]*180/pi," Refraction =",180 -
          ↪  temp[2]*180/pi," Time = ",temp[0]," s")
    x=x+dx
```

Like the previous program, the user can select and drag the initial and final position of the beam of light. As it traces out every possible trajectory, this reinforces the PLA's tendency to examine every path and select the one of least action, as opposed to the frame-by-frame nature of Classical Newtonian Mechanics, where position and momentum is given at the present and asked to find at some time in the future. Once again, real-time plots of the Incidence Angle, Refraction Angle, Light Time and Action are traced. This element not only enables the student to identify the path of least action, but also thereby conjecture Snell's Law.

2.3 Brachistochrone

The third program shows the time for a given path between two points. A numerical simulation is used to approximate the times for a given path between two points. To do so, the visualization involves three components: Partition, Approximation, and Modulation. The first component is partitioning, which involves breaking up a drawn path into its component points. Each pair of consecutive points are then taken and used to create a cylinder.

```
for i in range (starts[len(starts)-1],len(scene.objects)-1):
            newPart = sphere(radius = 0.1, pos = scene.objects[i].pos)
            partsOfCurve.append(newPart)
```

The second component is approximation, which involves estimating each cylinder as an inclined plane with $a = g\sin(\theta)$. θ is taken by the ratio of the horizontal component of the wire to the magnitude of its length. The integrals of this function is taken to get the magnitude of the velocity and position functions. The direction of these functions is set by taking the norm of cylinder on which the particle is traveling.

```
for i in range(0,len(partsOfCurve)-1):
            string = cylinder(pos = partsOfCurve[i].pos, axis =
            ↪  partsOfCurve[i+1].pos - partsOfCurve[i].pos, radius = 0.1,
            ↪  color = color.magenta)
```

```
            wholeWire.append(string)
ball.a=g*sin(wholeWire[numParts-1].axis.x/mag(wholeWire[numParts-1].axis))
 ↪  *norm(wholeWire[numParts-1].axis)
```

The third component is modulation, which involves a "hand-off" process. This process boils down to a recursive algorithm that takes the previous velocity and uses it as the seed for the initial velocity of the ball's next trajectory.

```
if ball.pos.y > wholeWire[j].pos.y:
                    ball.a =
                     ↪  g*sin(wholeWire[j].axis.x/mag(wholeWire[j].axis))
                     ↪  *norm(wholeWire[j].axis)
                else:
                    ball.a = -1 *
                     ↪  g*sin(wholeWire[j].axis.x/mag(wholeWire[j].axis))
                     ↪  *norm(wholeWire[j].axis)
```

The user can select any two points to denote the endpoints of the particle's trajectory. They can then draw an arbitrary number of paths between the two paths, with a time and action generated for each path. Each path also involves an animation with real-time graphs of the particle's Kinetic and Potential Energy and its Action. Given enough time and trail-and-error, the user should realize the goldilocks balance between gaining acceleration and ceding horizontal distance occurs in the case of a Brachistochrone, which is formed by tracing the trajectory of a point on the circumference of a cycloid.

3 Conclusion

In summary, the three numerical simulations outlined above are designed to address previously-known student difficulties with the Principle of Least Action. The programs use mouse interactivity, keyboard input, GlowScript animations, and graphical plots to engage students. Each of the three programs give a different perspective of the Principle of Least Action, through the reflective and refractive properties of light and through the Brachistachrone problem. Because of their interactive and interdisciplinary nature, these programs should help students intuitively understand the Action Principle to a deeper extent.

4 Acknowledgements

The author is grateful to Dr. Daniel Kabat for his constant help through the process of creating the visualizations and conducting the research. The author is also appreciative of insightful discussions with Dr. Bruce Sherwood.

References

[1] Stöltzner, Michael. "The principle of least action as the logical empiricist's Shibboleth." Studies in History and Philosophy of Science Part B: Studies in History and Philosophy of Modern Physics 34.2 (2003): 285-318.

[2] Bunn, James H. "Availing the Physics of Least Action." New Literary History, vol. 26, no. 2, 1995, pp. 419–442. JSTOR, www.jstor.org/stable/20057290. Accessed 20 Feb. 2021.

[3] Jourdain, Philip E. B. "The Nature and Validity of the Principle of Least Action." The Monist, vol. 23, no. 2, 1913, pp. 277–293. JSTOR, www.jstor.org/stable/27900429. Accessed 20 Feb. 2021.

[4] Bouzounieraki, Despoina. "The Principle of Least Action: An Explorative Investigation of Learning Difficulties and Teaching Strategies." University of Copenhagen, 1 June 2017, www.ind.ku.dk/english/research/didactics-of-physics/Master_Thesis_-_Despoina_Bouzounieraki.pdf.

[5] Feynman, Richard P. "The principle of least action in quantum mechanics." Feynman's Thesis—A New Approach To Quantum Theory. 2005. 1-69.

[6] Siburg, Karl Friedrich. The principle of least action in geometry and dynamics. No. 1844. Springer Science & Business Media, 2004.

[7] Coopersmith, Jennifer. The lazy universe: An introduction to the principle of least action. Oxford University Press, 2017.

[8] Feynman, Richard P., Robert B. Leighton, and Matthew Sands. "The principle of least action." The Feynman lectures on physics 2 (1964): 19-1.

[9] Robinson, Enders, and Dean Clark. "Fermat and the principle of least time." The Leading Edge 6.2 (1987): 34-37.

[10] Erlichson, Herman. "Johann Bernoulli's brachistochrone solution using Fermat's principle of least time." European journal of physics 20.5 (1999): 299.

[11] Helfgott, Harald, and Michel Helfgott. "A noncalculus proof that Fermat's principle of least time implies the law of refraction." American Journal of Physics 70.12 (2002): 1224-1225.

[12] Annila, Arto. "Least-time paths of light." Monthly Notices of the Royal Astronomical Society 416.4 (2011): 2944-2948.

[13] Jin, Zehao, Joshua Yao-Yu Lin, and Siao-Fong Li. "Learning Principle of Least Action with Reinforcement Learning." arXiv preprint arXiv:2011.11891 (2020).

[14] Helfgott, M., & Simonsen, L. M. (1998). Using Technology (Instead of Calculus) To Derive the Law of Reflection for Parabolic Mirrors from Fermat's Principle of Least Time. Mathematics and Computer Education, 32(1), 62-73.

[15] Sherin, Zachary W., et al. "Visualizing relativity: The openrelativity project." American Journal of Physics 84.5 (2016): 369-374.

[16] McKagan, S. B., et al. "Developing and researching PhET simulations for teaching quantum mechanics." American Journal of Physics 76.4 (2008): 406-417.