Rudimentary Physics with Python

An Application of Python Computer Algebra to Perform Calculus-based Kinematic Analysis

Alexander Ahmann
Manhattan, New York

Abstract

Python is "an interpreted high-level programming language for general-purpose programming."[1] As part of its "general-purpose programming," Python can be applied to solve scientific computing and numeric problems.[2,3] This paper, along with its accompanying presentation, will discuss working out the displacement function from an arbitrary non-constant acceleration function[4] with a symbolic computing library for Python known as *SymPy*.[5]

1 Background

Physicists are known for constructing highly maths-loaded models to explain the behaviour of physical entities.[6,7] The main reason is that adjectives like "slow," "very fast," "heavy," or "too long" are vague, will lead to fallacies of equivocation and goal-post shifting, ergo making any hypothesis impervious to falsification. Maths removes this ambiguity, opening up any contending hypothesis up to falsification. Working out the numeric solution to these models can be overwhelming, and therefore scientists and engineers turn to

[1] Copied verbatim from https://en.wikipedia.org/w/index.php?title=Python_(programming_language)&oldid=839618490

[2] see https://www.scipy.org/about.html

[3] Specifically, symbolic integrals; see http://docs.sympy.org/latest/modules/integrals/integrals.html

[4] see *Time Dependent Acceleration (n.d.)* article in bibliography

[5] see *Meurer et al. (2017)*

[6] see *Sharma (1982)* and *The Role of Mathematics in Physics (n.d.)*

[7] To quote an inspiration of mine: "You're going to need a lot of math, and I mean A LOT of math. Trying to do physics without math is like trying to drive without gas in the tank; you're just not gonna get very far;" see https://youtu.be/Kk8q500rYo4

scientific computing to work out tedious calculations.[8] Python, an open-source, popular,[9,10] easy to learn[11,12,13] programming language, is perhaps the most valuable weapon by which the physical scientist may attack tedious numeric problems. The basic procedure to solving classical mechanics problems on a conceptual level is straightforward. We start out by identifying a problem of concern, determining what kind of problem we are working on,[14] draw a diagram representing the problem, list formulæ relevant to the problem, substitute known quantities, and finally to work out its maths.[15] Physicists are interested in first calculating the dynamical aspects of a mechanical apparatus in order to derive an acceleration function; they then apply an integration on the acceleration function to derive its velocity function, and finally another integration on the previously discovered velocity function to derive a displacement function; the entire process can be summarised with *Eq. 1*:[16]

$$\frac{1}{m}\Big(\sum F = \vec{F}_{net}\Big) = \vec{a}(t) = \frac{d\vec{v}}{dt} = \frac{d^2\vec{x}}{dt^2} \tag{1}$$

The analyst here is interested in literally summing up all forces acting on the mechanical system and diving by the total mass m to derive the acceleration function. Anti-derivatives will be applied to derive the displacement function.

2 The task at hand

In this paper, we will explore the integration procedure for deriving a displacement function given an acceleration function. Specifically, we will be

[8] see *Gustafsson (2011); Stewart (2014)*

[9] The IEEE has conducted a number of surveys ranking programming languages by popularity. In 2017, Python ranked first(Cass, 2017). In 2015, Python ranked fourth (Cass, 2015).

[10] GitHut is a service that keeps track of GitHub pushes by programming language. On the fourth quarter of 2017, Python ranks second; see https://madnight.github.io/githut/#/pushes/2017/4

[11] Whilst "easy to learn" is subjective, some have advocated introducing amateurs and students to computer science with Python. Eric Raymond, a software developer, recommended it as a starting point; see *Raymond (2017)*

[12] see *Radenski (2017)*

[13] There are other programming languages that students may learn first, such as Java; see *Kamin et al. (2002)*

[14] Is it a kinematics problem, a momentum problem, an energy conservation problem, or any other class of problems?

[15] see *Savov (2016, p. 237)*

[16] (Savov, 2016, p. 164-165)

concerning ourselves with non-constant acceleration. Non-constant acceleration is an acceleration function where, unlike Uniform Acceleration Motion,[17] its behaviour is conditional on other factors and variables. We will work out the displacement function given the hypothetical acceleration function *Eq. 2*:[18]

$$a(t) = b + ct + dt^2 \tag{2}$$

Here, the acceleration of a body is conditional on arbitrary constants b, c, and d; and variable t represents the time elapsed. The task at hand is to work out from with Python; how shall we go about solving it computationally? Before we discuss the Python implementation of the solution, we need to express the kinematics aspect of *Eq. 1* with the more compact *Eq. 3*:[19]

$$x(s) = \int_0^s \left[v_i + \int_0^\tau a(t)\, dt \right] d\tau \tag{3}$$

SymPy is a computer algebra package for the Python programming language that can perform integration. To begin,[20,21,22] open up Python IDLE and import SymPy functions.

```
>>> from sympy import *
```

Next, we must define the acceleration function and its constants:

```
>>> b,c,d,v_i,x_i,t = symbols("b c d v_i x_i t")
>>> a = b + c*t + d*t**2
```

Finally, we will make use of SymPy's *integrate()* function to express Eq. 3 and work out the displacement function of the acceleration function:

```
>>> x_i + integrate(v_i + integrate(a, t), t)
```

You should have the following output:

```
>>> b*t**2/2 + c*t**3/6 + d*t**4/12 + t*v_i + x_i
```

This output can be expressed in terms of *Eq. 4*:

$$x(s) = \frac{bt^2}{2} + \frac{ct^3}{6} + \frac{dt^4}{12} + v_i t + x_i \tag{4}$$

[17](Garcia, 2014, p. 21-36)

[18]Example from *Time Dependent Acceleration (n.d.)* article in bibliography

[19](Savov, 2016, p. 344)

[20]It is assumed that you have *Python* installed. If not, see https://www.ics.uci.edu/p̃attis/common/handouts/pythoneclipsejava/python.html

[21]It is assumed that you have *SymPy* installed. If not, see https://github.com/sympy/sympy/wiki/Download-Installation

[22]Alternatively, you can use SymPy Live: http://live.sympy.org/

3 Conclusion

This marriage of physics and computer science will demonstrate itself to be useful. A general application would obviously be deriving the solution to tedious general physics problems and engineering.

A particular application of this, that would perhaps interest a forensic investigator, would be video analysis.[23,24] Python can take the results of raw kinematic results and compare them to the predictions of a physics model. Another application of kinematics analysis would involve biomechanical analysis of animals.[25] One rather interesting subject of biomechanics that may be of interest to the biologist or sports-bettor is Thoroughbred racehorse analysis.[26]

Python and its symbolic computing library has provided us with a means by which to solve numerical problems that are not necessarily conceptually difficult, but rather tedious and time consuming.

References

Allain, R. (2016). *Physics and video analysis.* Bristol (Temple Circus, Temple Way, Bristol BS1 6HG, UK): IOP Publishing.

Cass, S. (2015, July 20). The 2015 Top Ten Programming Languages. Retrieved May 4, 2018, from https://www.csee.umbc.edu/courses/undergraduate/202/spring16_marron/lectures/l01/the_2015_top_ten_programming_languages.pdf

Cass, S. (2017, July 18). The 2017 Top Programming Languages. Retrieved May 4, 2018, from https://spectrum.ieee.org/computing/software/the-2017-top-programming-languages

archive: https://archive.fo/Hoyy0

Garcia, N. (2014). *Physics for computer science students: With emphasis on atomic and semiconductor physics.* Springer.

[23] Assuming that the physics model is sound, a statistically significant deviation from the model would most likely imply a doctored video.

[24] (Allain, 2016)

[25] (McCaw, 2014)

[26] An early researcher into quantitative biomechanical analysis of the Thoroughbred racehorse is Harry Laughlin; see *Laughlin (1934)*

Gustafsson, B. (2011). *Fundamentals of scientific computing.* Berlin: Springer.

Kamin, S. N., Mickunnas, M. D., & Reingold, E. M. (2002). *An Introduction to computer science using Java.* Boston, MA, McGraw-Hill.

Laughlin, H. H. (1934). Racing Capacity in the Thoroughbred Horse. Part I. *The Scientific Monthly, 38,* 310-321. Retrieved May 9, 2018, from http://www.jstor.org/stable/15639

McCaw, S. T. (2014). *Biomechanics for dummies.* Hoboken: Wiley.

Meurer, A., Smith2, C. P., Paprocki3, M., Kirpichev5, S. B., Rocklin3, M., Ivanov7, S., ... Skillman SW. (2017, January 02). SymPy: Symbolic computing in Python. Retrieved May 4, 2018, from https://peerj.com/articles/cs-103

Radenski, A. (2006). "Python first": A Lab-Based Digital Introduction to Computer Science. *ACM SIGCSE Bulletin, 38(3),* 197. doi:10.1145/1140123.1140177

Raymond, E. S. (2001). How To Become A Hacker (Last Revision: October 6, 2017). Retrieved May 4, 2018, from http://www.catb.org/esr/faqs/hacker-howto.html

Savov, I. (2016). *No bullshit guide to math & physics.* Montréal, Québec: Minireference.

Sharma, C. S. (1982). The Role of Mathematics in Physics. *The British Journal for the Philosophy of Science, 33, 3rd ser.,* 275-286. Retrieved May 4, 2018, from http://www.jstor.org/stable/687225

Stewart, J. (2014). Python for Scientists. *Cambridge: Cambridge University Press.* doi:10.1017/CBO9781107447875

The Role of Mathematics in Physics. (n.d.). Retrieved May 4, 2018, from http://www.batesville.k12.in.us/physics/phynet/aboutscience/role_of_math.htm Published by *The Batesville Community School Corporation*

Time Dependent Acceleration. (n.d.). Retrieved May 3, 2018, from http://hyperphysics.phy-astr.gsu.edu/hbase/avari.html