Comparing Different Numerical ODE Methods When Applied to a Single Pendulum

Adarsh Iyer

December 17th, 2021

1 Abstract

The goal of this project is to compare different methods of simulating a single pendulum with an analytical solution. There were 3 main stages of this project: The mathematical calculations, the writing of the simulation code, and the construction of the plots. The final product of this is a jupyter notebook with an easy-to-use function to simulate a single pendulum using multiple simulation methods with various starting conditions.

2 Introduction

This investigation was motivated by the ever-increasing number and complexity of systems modeled by differential equations that are present in the world. The economy, physics simulations, video games, and many more systems are increasingly described by differential equations. However, as these systems become more complex, they will tend to not have exact analytic solutions. The need to approximate these systems is increasingly necessary in the modern world. The choice of the single pendulum is motivated by it's simple physical nature and the existence of an exact solution to it, placing it as a top candidate for a physical system that can be simulated and exactly calculated.

3 Background

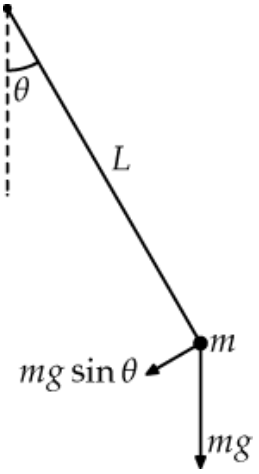
3.1 The Simple Pendulum

The simple pendulum is a massless rod connected to a point mass, with one end fixed and the other free to move, subjected to a uniform gravitational field. A diagram is shown below in Figure 1.

Observing that the acceleration is totally in the $\hat{\theta}$ direction, we can write this system as a differential equation.

$$\ddot{\theta} + \frac{g}{L}\sin\theta = 0 \tag{1}$$

Figure 1: Example of a simple pendulum



For small θ, $\sin\theta$ is often approximated to just θ, resulting in the equation

$$\ddot{\theta} + \frac{g}{L}\theta = 0 \tag{2}$$

This is the well-known simple harmonic oscillator, which has a solution in the form

$$\theta(t) = \theta_0 \cos\left(\sqrt{\frac{g}{L}}t\right) \tag{3}$$

This is, however, very inaccurate, and does work for larger values of θ.

3.2 Analytical Solution

The inner workings of the analytical solution to the simple pendulum are beyond the scope of this paper. However, this paper will briefly cover some of the relevant theory. The most common form of the exact solution appears in the form of a solution to an elliptic integral of the first kind, which is not an easily implementable form in python. Thankfully, there are other ways of expressing the solution. According to this paper "Exact solution for the nonlinear pendulum" [1], the pendulum can be expressed as a function of the Neville theta functions. The particular functions that are relevant to the pendulum are

$$\theta_n(z,m) = \frac{\sqrt{2\pi}}{2(1-m)^{1/4}\sqrt{K(m)}}\left(1 + 2\sum_{k=1}^{\infty}(-1)^k(q(m))^{k^2}\cos\left(\frac{\pi z k}{K(m)}\right)\right) \tag{4}$$

$$\theta_s(z,m) = \frac{\sqrt{2\pi}q(m)^{1/4}}{m^{1/4}(1-m)^{1/4}\sqrt{K(m)}}\sum_{k=0}^{\infty}(-1)^k(q(m))^{k(k+1)}\sin\left(\frac{(2k+1)\pi z}{2K(m)}\right) \tag{5}$$

where

$K(m)$ is the complete elliptic integral of the first kind

$$q(m) = e^{-\pi\frac{K(1-m)}{K(m)}}$$

Using these functions, the analytic solution to the pendulum is

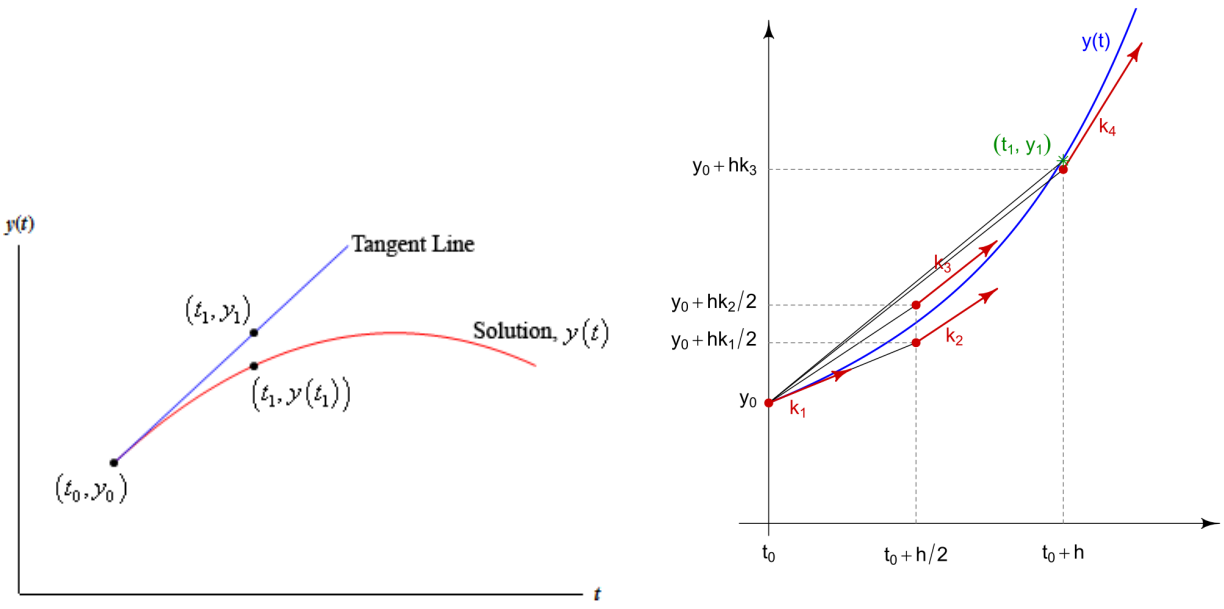
$$2\arcsin\left\{\sin\frac{\theta_0}{2}sn\left[K\left(\sin^2\frac{\theta_0}{2}\right)-\omega_0 t;\sin^2\frac{\theta_0}{2}\right]\right\}, \tag{6}$$
$$\text{where } sn=\frac{\theta_s}{\theta_n}$$

While this is very complicated, it is feasible to implement this into python code.

3.3 Runge-Kutta Method

The use of the Simple Harmonic Motion approximation is virtually unusable to model real life pendulums over long periods of time. The Runge-Kutta family of approximation methods is a much more accurate approach to simplifying the calculations of a pendulum. The principle behind the Runge-Kutta is to use linearizations at points nearby each other to approximate the change over a small step interval, and then simply sum these changes to achieve the next value. The simplest form of this is the Runge-Kutta 1, also known as Euler's Simple Method, which uses one linearization at each step. While this is much more accurate than Simple Harmonic Motion, the error still grows quite large. However, Runge-Kutta 2 is much more accurate, Runge-Kutta 3 even more, and Runge-Kutta 4 the most, using 4 different linearizations at beginning, middle, and end points to calculate each step. This accuracy is quite high. A difference between Runge-Kutta 1 and 4 is shown below.

Figure 2: Runge-Kutta 1 (left) vs Runge-Kutta 4 (right) [3] [4]



4 Code

The implementation of the code was tedious, but not wholly challenging to achieve. The form of the analytical solution using the Neville theta functions is optimal because it allows the elliptic integral to be computed directly instead of being an equation to solve. This function is convenient provided by the SciPy python library as `scipy.special.ellipk`. The infinite sums are approximated by using a sum to k=10,000. The full implementation can be found in the GitHub repository [2].

The simulation code for the Runge-Kutta methods is set up so that the step function can be substituted for whatever step algorithm is desired, so it is easy to pass in the desired step algorithm, which in this case are the Runge-Kutta 1,2,3 and 4 methods.

Graphing these simulations was simple, as the algorithm simple takes in a time array and returns an array of theta positions at each time. MatPlotLib was used to plot these graphs.

5 Results

A full spread of plots can be found in the GitHub repository [2].

In figure 3 below, the top left graph is the analytic solution, and the other 5 are the respective errors of each simulation method to the analytic solution. Even at only 20°, the differences between the methods is apparent. SHM is off by .6 a radian at times, while RK4 sits at a maximum error of 10^{-11} radians. It is clear that the Runge-Kutta approximation methods are superior to SHM, and that RK4 is incredibly accurate over long spans of time.

Further more, observing figure 4, where the initial angle is 179°, the RK1 method seemingly gains energy with each rotation, completing over a full rotational swing. The SHM solution demonstrates some peculiar cyclical patterns to its error, however it is very inaccurate, at a maximum of 6 radians off from the real solution. Still, Runge-Kutta 4 is proven to be incredible accurate with a maximum of only 10^{-8} radians difference from exact solution.

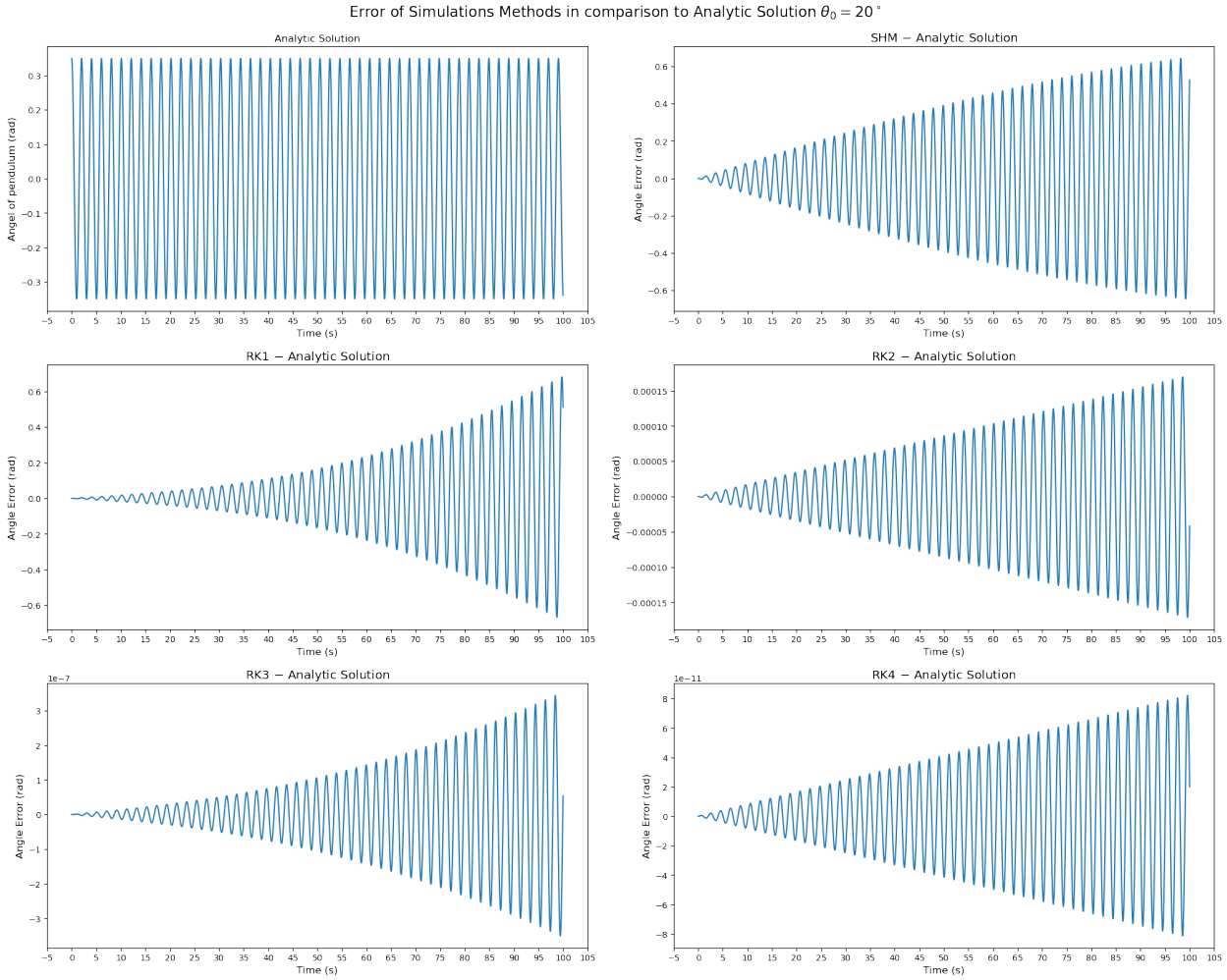


Figure 3: Simulation error at $\theta_0 = 20°$

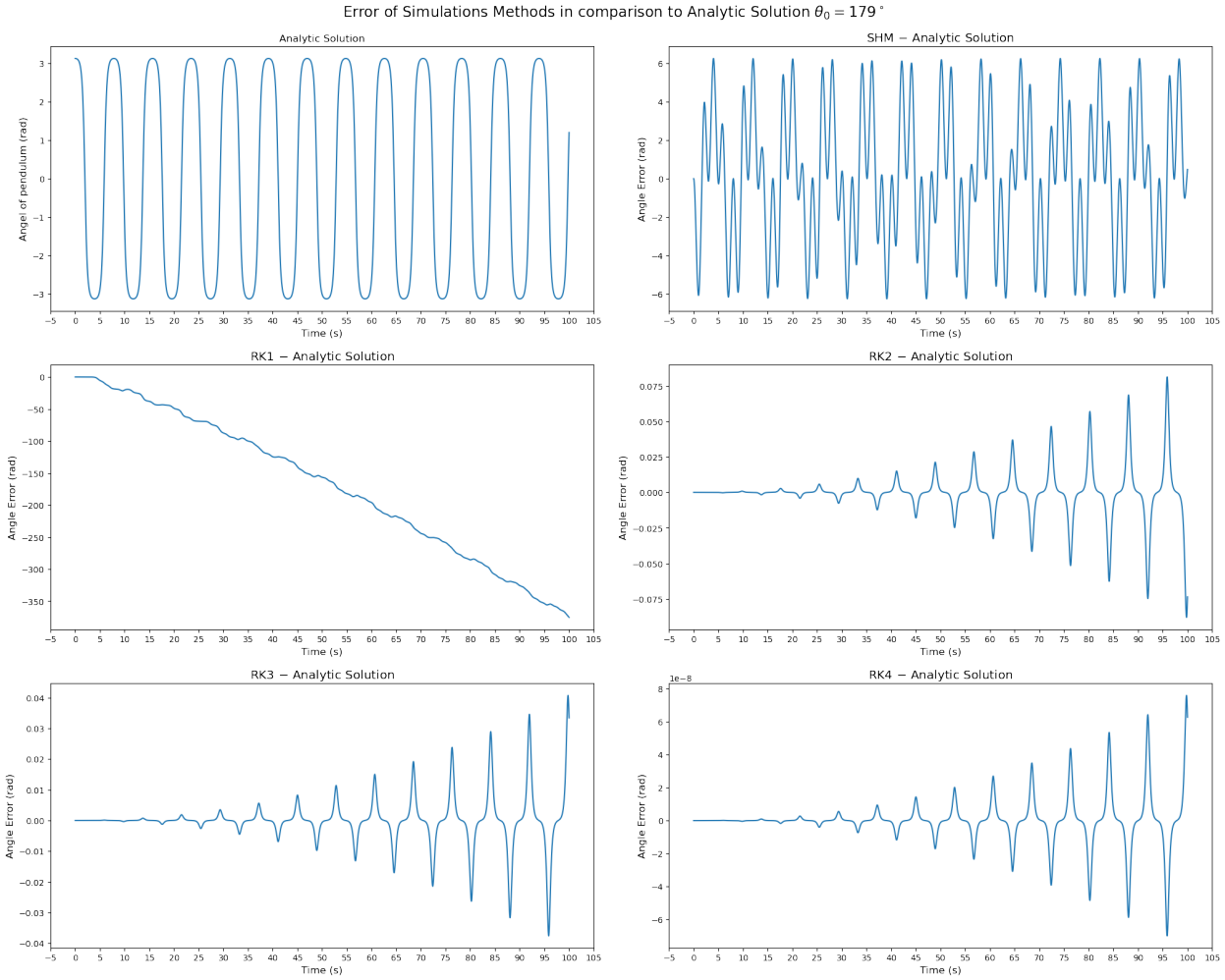


Figure 4: Simulation error at $\theta_0 = 179°$

6 Conclusion

This investigation has shown that certain approximation methods for simple pendulums are incredibly inaccurate, while others are extremely accurate. Specifically, it has shown the Runge-Kutta method to be the superior method for approximating this system, as well as a simpler implementation than the analytical solution. It would be logical to assume that Runge-Kutta 4 is in many scenarios generally applicable, as the merits that make it more accurate than Euler's Simple Method are intrinsic to its computation, and the use of more data points in the calculation can only increase its accuracy. If one were to make a pendulum simulation, this study advises them to use Runge-Kutta approximations, as they lose virtually no accuracy whilst saving enormously on computational resources and time.

References

[1] Augusto Beléndez, Carolina Pascual, DI Méndez, Tarsicio Beléndez, and Cristian Neipp. Exact solution for the nonlinear pendulum. *Revista brasileira de ensino de física*, 29:645–648, 2007.

[2] Adarsh Iyer and Jeffrey Kim. pendulum-simulations-77. `https://github.com/AdarshI/pendulum-simulations-77`, 2021.

[3] Paul. `https://tutorial.math.lamar.edu/classes/de/eulersmethod.aspx`.

[4] Hilber Traum. `https://commons.wikimedia.org/w/index.php?curid=64366870`.