An Analysis of Wikipedia Accuracy: The Swinging Atwood's Machine

Justin Casso

California State Polytechnic University of Pomona
jrcasso@cpp.edu

Abstract

In this paper, we simulate the Swinging Atwood's Machine (SAM) system with a 4^{th}-order RK4 algorithm with two degrees of freedom. These simulations are then used to determine the accuracy of the Wikipedia article under which this topic falls under. Three major claims are individually examined and validated through careful physics, pragmatic programming, and endless expresso shots. Our findings indicate that the Wikipedia article on the Swinging Atwood's Machine is both factually accurate and reproducible.

I. Introduction

I. Brief History

The Atwood's machine facilitates a fundamental experiment that explores the precise interaction of gravitational acceleration in the laboratory frame. This experiment was first proposed by G. Atwood in the late 18^{th} century [1]. For many undergraduates, it is one of the first direct observations of the influence Earth's mass exhibits on a system of masses. A substantially more complicated system, known as the *swinging* Atwood's machine (hereafter abbreviated at *SAM*), is a much more enlightening system to analyze. The *SAM* system exhibits many intriguing, distinct, and abundantly rich trajectories.

II. Behaviors of Interest

A fundamental constant in the SAM system is the ratio of masses connected via pulleys, defined below as μ. Values of μ are enormously influential on determining distinct trajectories. There are two possibilities that both have subsequent categories: *singular*, and *non-singular* motion, with the former involving the mass (at some point during the trajectory) passing through the origin of the pulley.

III. Status Quo

It has been attempted by *Tufillaro* [2], among others, to categorize these trajectories. His senior thesis delved deep into the behavior of the *SAM* and thus deserves a great deal of attention. Since the original paper outlined in [2], *Tufillaro* has since produced at least five additional papers on the subject; although, *O. Pujol et al.* [3] has recently provided an incredibly well-rounded paper on the SAM as well. The former era of *SAM* research was conducted where computational methodologies were substantially less powerful than that applicable today. The latter research includes insight into the impact massive pulleys have on the system, which evidently is monumentally influential. With this most recent paper, the majority of questions originally posed have been wrapped and resolved into a nice package. The only thing missing is an epilogue with substantial validation through numerical techniques. Herein lies our purpose.

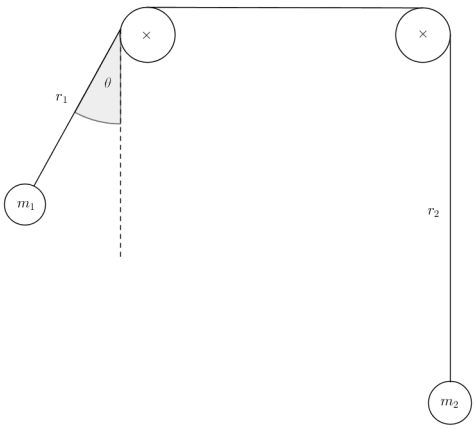
IV. The Prime Directive

Our purpose in this paper is to determine whether the *Wikipedia* article [4] claims and figures are accurate. We endeavor to verify that our findings agree with most, if not all, of the claims. The claims the article makes that we

will be testing are as follows:

- There are two coupled ordinary differential equations that govern the trajectories of the system.
- Trajectories can be non-singular, singular, bounded, unbounded, periodic, and chaotic.
- Some trajectories are visualized on the article. We will attempt to reproduce them.

Many images of trajectories are included in the article with corresponding values of μ (to wit: $\mu \equiv M/m$). We will attempt to reconstruct as many as we can, and comment on any behaviors that differ from expectations.



The setup for our system.

II. Methods

I. Equations of Motion

Before we can actually simulate anything, we need to find the equations of motion for the *SAM* system. The most straightforward way to do this is by applying the Euler-Lagrange equation to the figure above. If you are unfamiliar with the methodologies of the Lagrangian formalism, a short introduction has been included in *Appendix I*. A full derivation of the equations of motion can be found in *Appendix II*. A quick inspection of the derivation should convince the reader that we must solve these two coupled ordinary differential equations:

$$(1+\mu)\frac{d^2r}{dt^2} = r\left(\frac{d\theta}{dt}\right)^2 + g\cos(\theta) - g\mu \quad (1)$$

$$r\frac{d^2\theta}{dt^2} = -g\sin(\theta) - 2\frac{dr}{dt}\frac{d\theta}{dt} \quad (2)$$

Furthermore, the *Wikipedia* article was indeed correct in the derivation of these formulas. The first claim is evidently true.

II. Important Simulation Parameters

For this system, we will be using a 4^{th}-order Runge-Kutta algorithm with two degrees of freedom in the Python programming language. There are many important *algorithmic* parameters that determine the efficacy of the program, and they are as follows:

- Step-size (Δt): the step-size determines how far into the future to calculate. The smaller the step, the more accurate the calculation.
- Maximum time: the maximum time to calculate to.
- Singularity threshold: this is used to detect if the motion is singular; if the radius ever becomes less than $threshold * r_0$, the program exits early to avoid stack overflows. The trajectory is then displayed.

Equally important are the *initial conditions*:

- r_0: the initial radius of the swinging mass.
- $\dot{r}_0$: the initial radial velocity of the swinging mass.
- θ_0: the initial angle of the swinging mass.
- $\dot{\theta}_0$: the initial angular velocity of the swinging mass.
- g: the gravitational acceleration of the Earth (9.8 m/s^2).
- $\mu = m_2/m_1$: the ratio of the non-swinging mass to the swinging mass.

III. Workflows of the Programs

Included are two programs. The first is used to calculate and plot a single trajectory. The second is used to calculate, plot, and save images of a plethora of trajectories from a predefined range of μ's. The first program finds utility in calculating specific trajectories, such as those in the *Wikipedia* article. The second program is a streamlined way for producing tens of thousands of images of distinct trajectories for a given range of μ (for instance, from $\mu = 1$ to $\mu = 20$ in steps of 0.002).

At the heart, the program is quite simple: it adheres to a basic RK4 algorithm sketch. A few *try-catch* methods and error-flagging features features were necessitated to manage potential stack overflows (such as those in singular trajectories) and save data. The program begins by defining the algorithmic functions. The first function is a wrapper that handles the systematic updating of the trajectory - the general RK4 framework function. The second function is the SAM-specific function that puts those equations of motion to work, and returns trajectory values to the wrapper. The program then plots the trajectory on a polar mapping, with initial conditions noted.

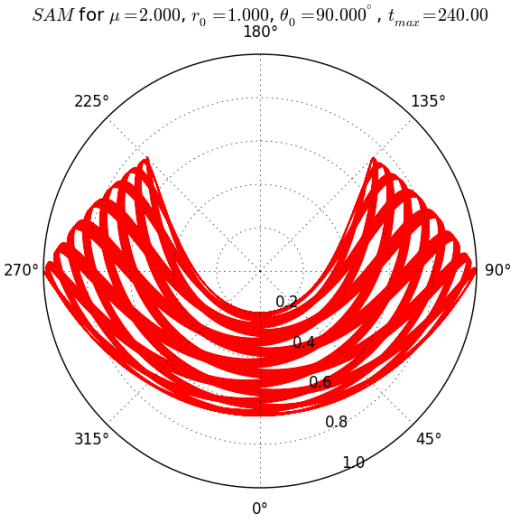
The program has some fail-safes to ensure our valuable time isn't wasted. Most often than not, the program will fully complete. The only time it will not fully complete is when a trajectory is *singular.* To solve the coupled differential equations, we had to divide by $r(t)$ - see *Eq. 2*. Thus, as $r \to 0$, things start to blow up. At each step of the algorithm, the program checks to make sure there isn't a stack overflow. If we come within range of the singularity threshold (as is intended), the program will exit prematurely and plot the trajectory up to the point of cessation, thus saving our valuable data. The amount of time calculated is then printed to indicate how close to the maximum time the program calculated to before exiting prematurely. If there *is* a stack overflow (quite rare), that means we've calculated too close to the singularity (i.e. we've started to divide by very-close-to-zero), and the program instructs the user to increase the singularity threshold. Plotting procedures are abandoned in this circumstance because the results are almost always. This is an unavoidable circumstance that can only be rectified by increasing the singularity threshold. Generally speaking, it's a safe practice to have your singularity threshold set to at most 0.01.

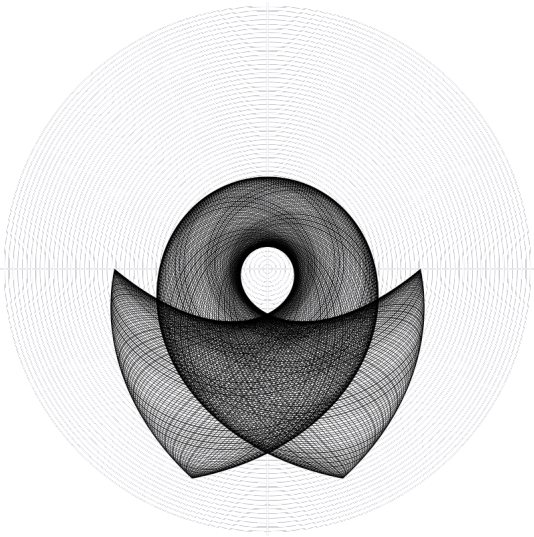
IV. Final Remarks

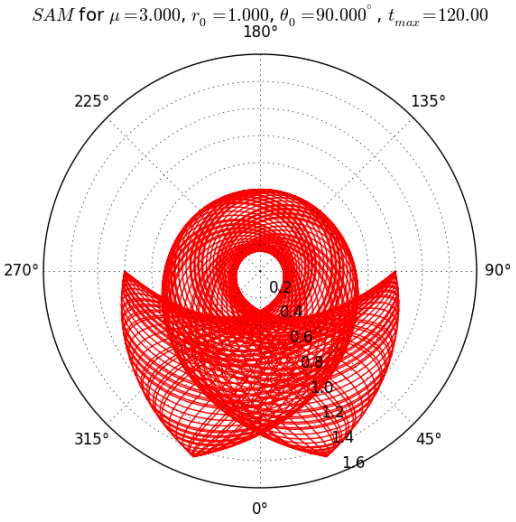
The second program is very powerful. It took about 3 hours on an Intel i3 (2nd generation) processor to produce about 10,000 images of distinct trajectories. Be aware that these images can eat memory ridiculously fast; it would be wise to produce only a few hundred images at a time to learn the subtleties of operation before attempting to produce thousands. Be sure to have the correct file path for the save destination. As an added feature, the second program also prints the total calculation time so you can benchmark performance.

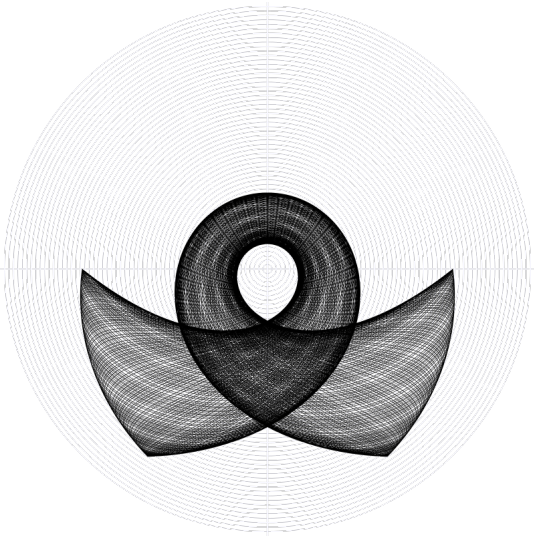
Wikipedia Images

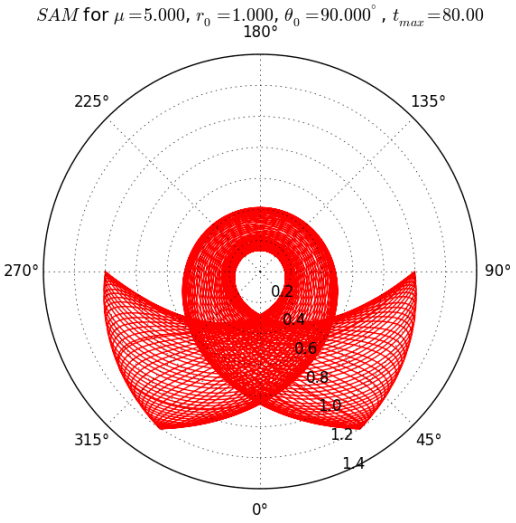
My Images

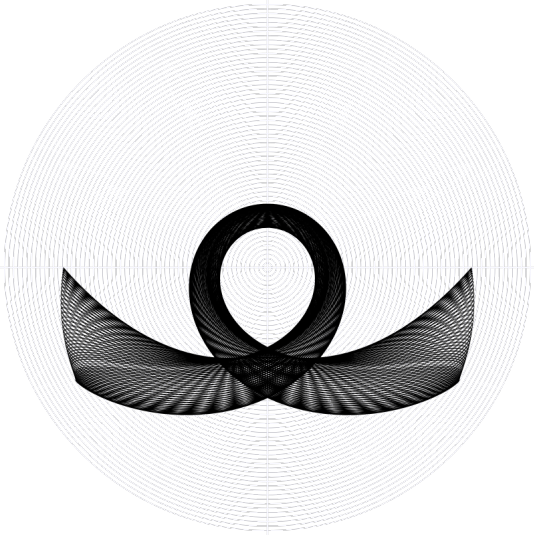


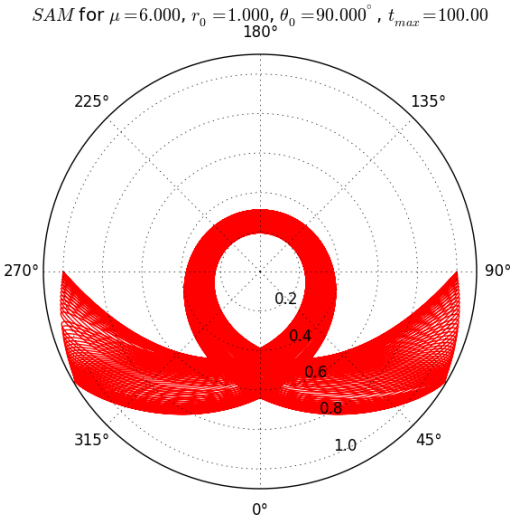


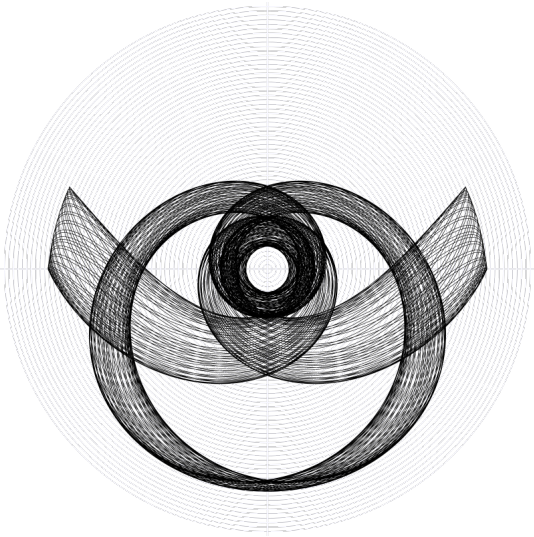


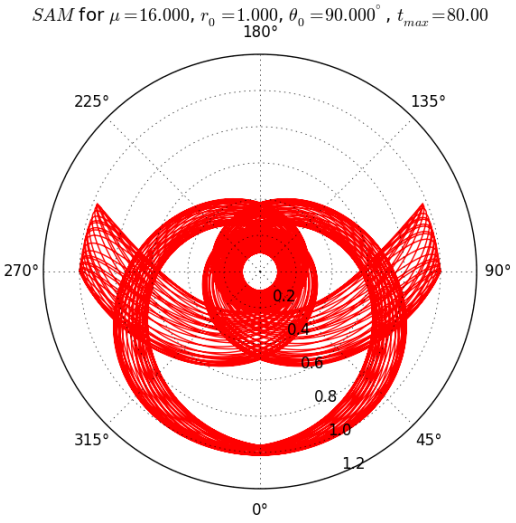


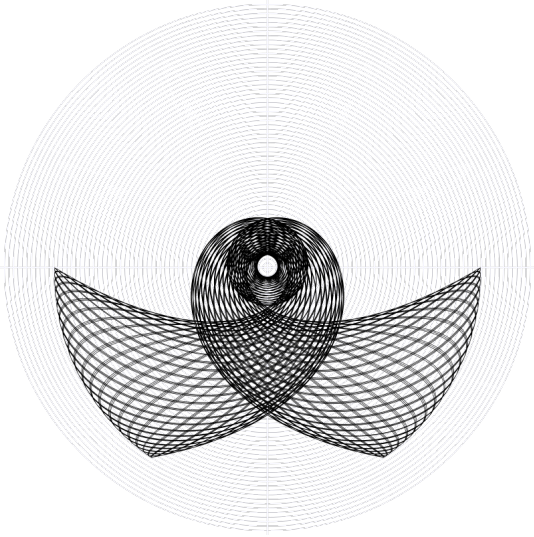

SAM for μ=5.000, r_0 =1.000, θ_0 =90.000°, t_{max}=80.00
180°
225°
135°
270°
90°
315°
45°
0°
0.2
0.4
0.6
0.8
1.0
1.2
1.4

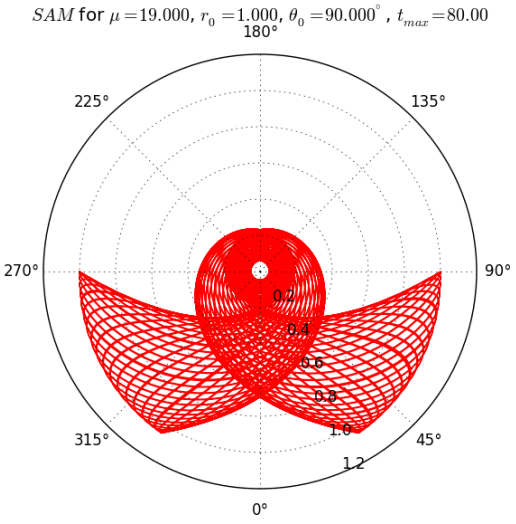


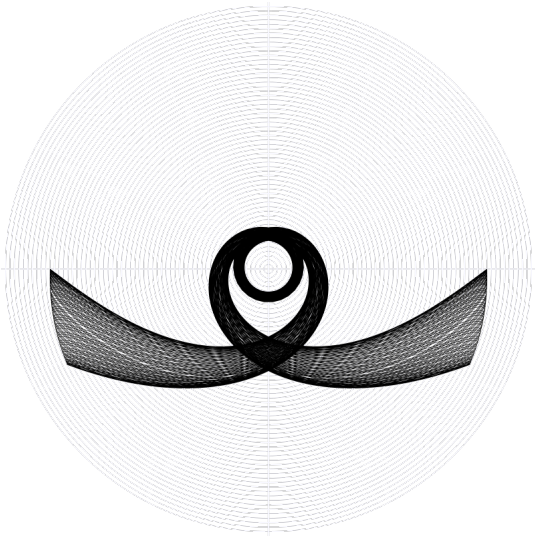

SAM for μ=6.000, r_0 =1.000, θ_0 =90.000°, t_{max}=100.00
180°
225°
135°
270°
90°
315°
45°
0°
0.2
0.4
0.6
0.8
1.0

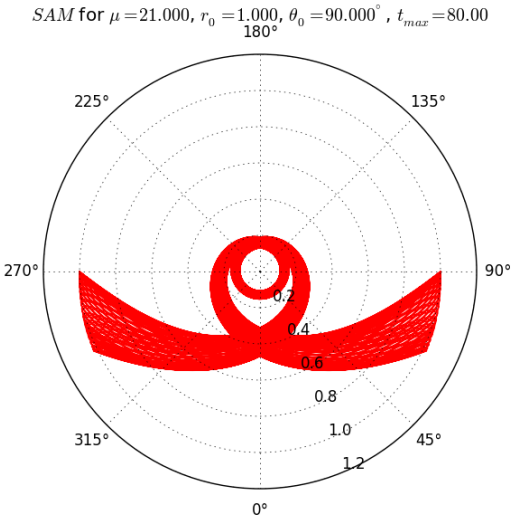


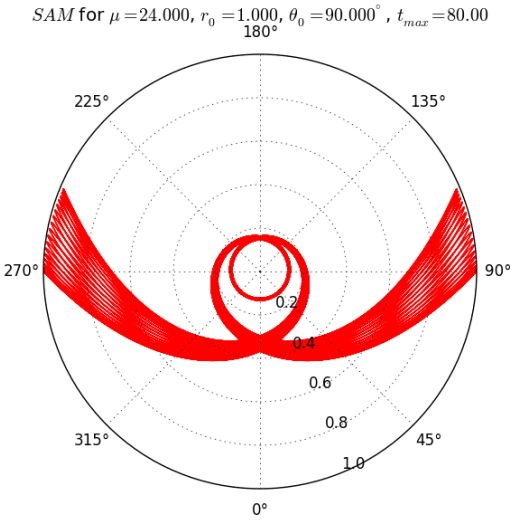

SAM for $\mu = 16.000$, $r_0 = 1.000$, $\theta_0 = 90.000^\circ$, $t_{max} = 80.00$
180°
225°
135°
270°
90°
315°
45°
0°
0.2
0.4
0.6
0.8
1.0
1.2

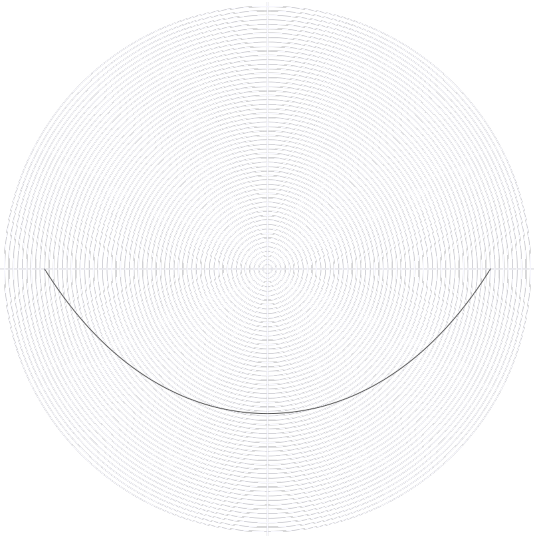


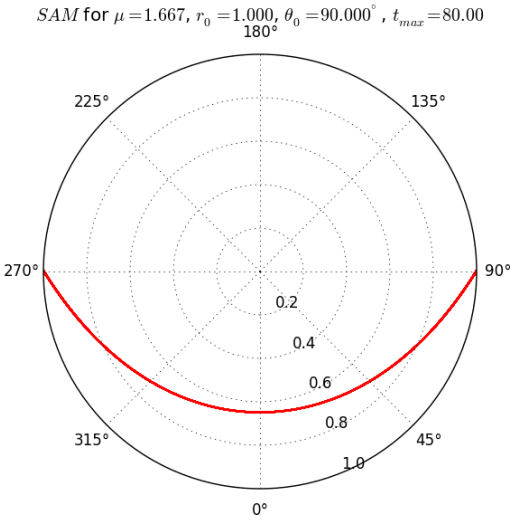

SAM for $\mu = 19.000$, $r_0 = 1.000$, $\theta_0 = 90.000^\circ$, $t_{max} = 80.00$
180°
225°
135°
270°
90°
315°
45°
0°
0.2
0.4
0.6
0.8
1.0
1.2

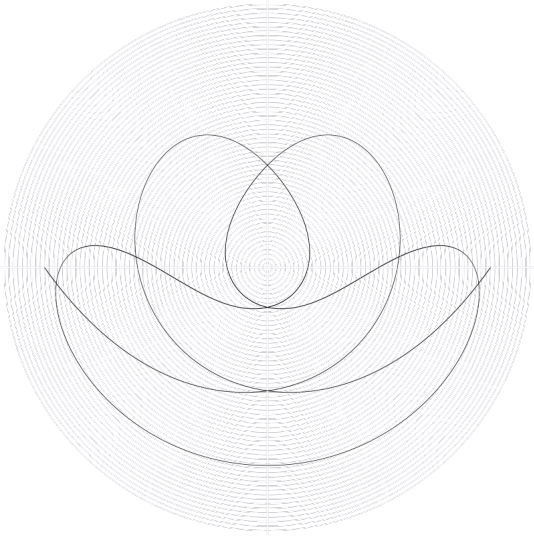


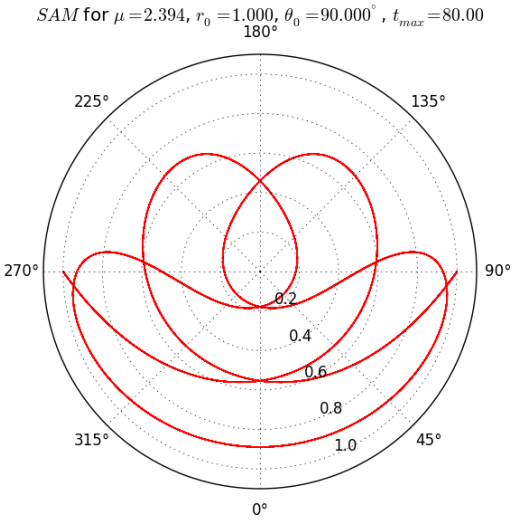

SAM for μ=21.000, r_0 =1.000, θ_0 =90.000°, t_{max}=80.00
180°
225°
135°
270°
90°
315°
45°
0°
0.2
0.4
0.6
0.8
1.0
1.2

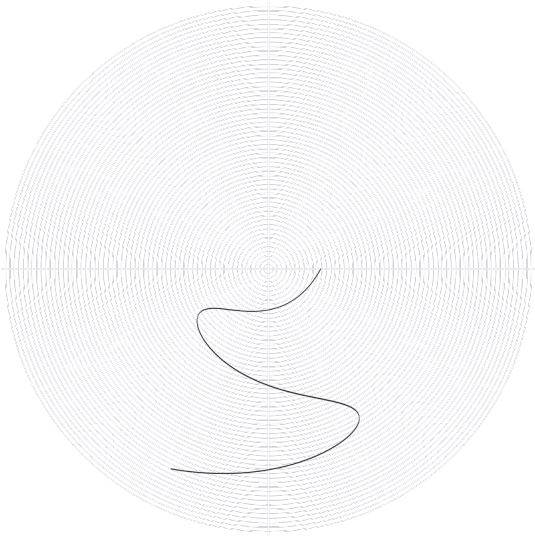

SAM for μ=24.000, r_0 =1.000, θ_0 =90.000°, t_{max}=80.00
180°
225°
135°
270°
90°
315°
45°
0°
0.2
0.4
0.6
0.8
1.0

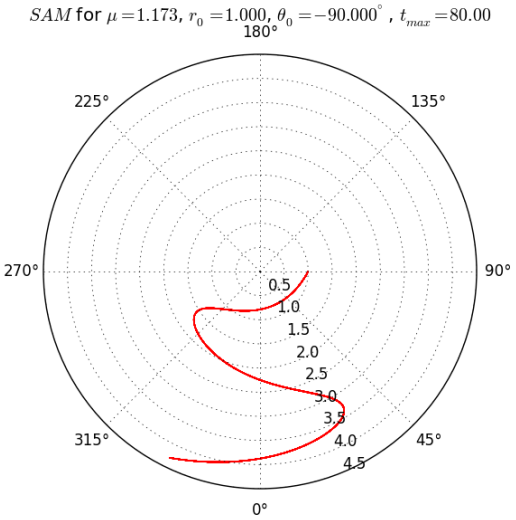


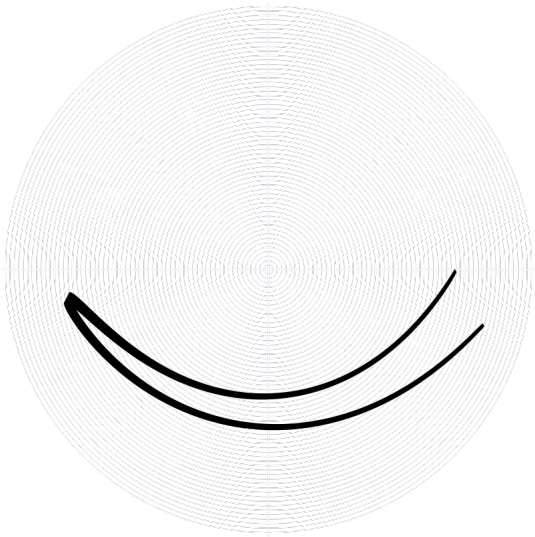

SAM for $\mu = 1.667$, $r_0 = 1.000$, $\theta_0 = 90.000^\circ$, $t_{max} = 80.00$
180°
225°
135°
270°
90°
315°
45°
0°
0.2
0.4
0.6
0.8
1.0

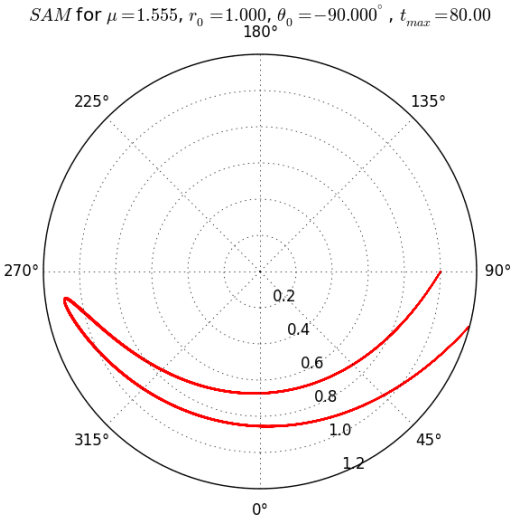


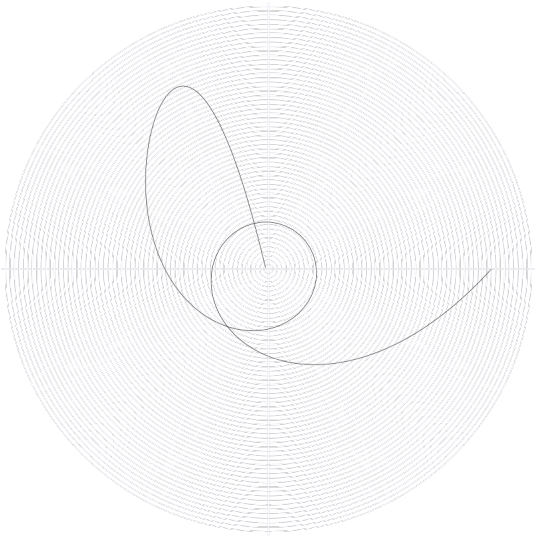

SAM for $\mu = 2.394$, $r_0 = 1.000$, $\theta_0 = 90.000^\circ$, $t_{max} = 80.00$
180°
225°
135°
270°
90°
315°
45°
0°
0.2
0.4
0.6
0.8
1.0

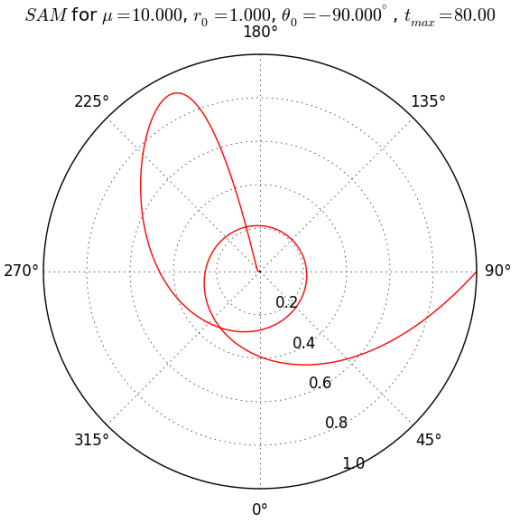


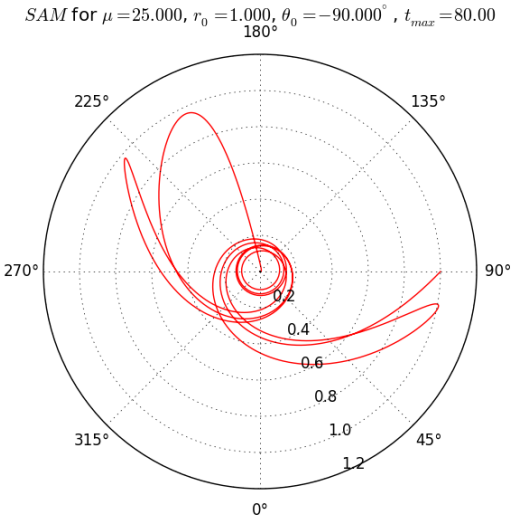

SAM for $\mu = 1.173$, $r_0 = 1.000$, $\theta_0 = -90.000^\circ$, $t_{max} = 80.00$
180°
225°
135°
270°
90°
315°
45°
0°
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5

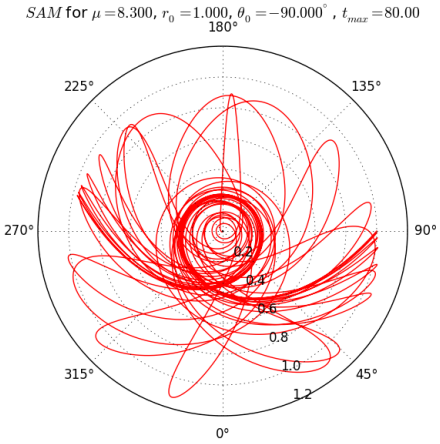


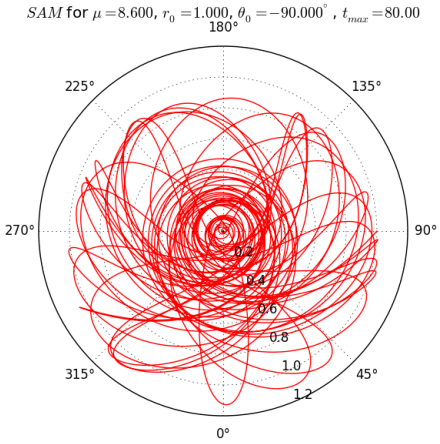

SAM for $\mu = 1.555$, $r_0 = 1.000$, $\theta_0 = -90.000^\circ$, $t_{max} = 80.00$
180°
225°
135°
270°
90°
315°
45°
0°
0.2
0.4
0.6
0.8
1.0
1.2

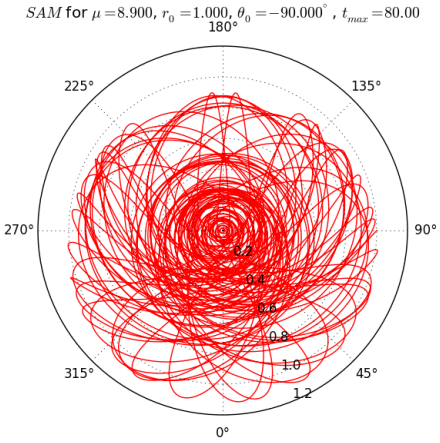


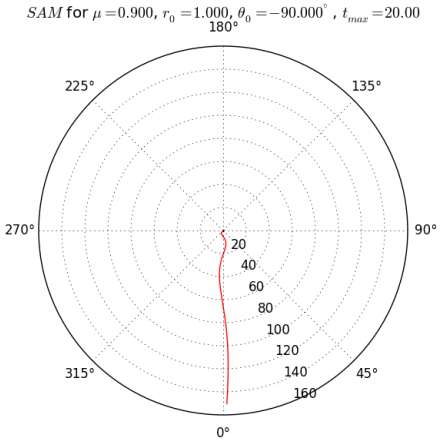

SAM for $\mu = 10.000$, $r_0 = 1.000$, $\theta_0 = -90.000^\circ$, $t_{max} = 80.00$
180°
225°
135°
270°
90°
315°
45°
0°
0.2
0.4
0.6
0.8
1.0


SAM for $\mu = 25.000$, $r_0 = 1.000$, $\theta_0 = -90.000^\circ$, $t_{max} = 80.00$
180°
225°
135°
270°
90°
315°
45°
0°
0.2
0.4
0.6
0.8
1.0
1.2


SAM for $\mu = 8.300$, $r_0 = 1.000$, $\theta_0 = -90.000^\circ$, $t_{max} = 80.00$
180°
225°
135°
270°
90°
315°
45°
0°
0.2
0.4
0.6
0.8
1.0
1.2


SAM for $\mu = 8.600$, $r_0 = 1.000$, $\theta_0 = -90.000^\circ$, $t_{max} = 80.00$
180°
225°
135°
270°
90°
315°
45°
0°
0.2
0.4
0.6
0.8
1.0
1.2


SAM for $\mu = 8.900$, $r_0 = 1.000$, $\theta_0 = -90.000^\circ$, $t_{max} = 80.00$
180°
225°
135°
270°
90°
315°
45°
0°
0.2
0.4
0.6
0.8
1.0
1.2


SAM for $\mu = 0.900$, $r_0 = 1.000$, $\theta_0 = -90.000^\circ$, $t_{max} = 20.00$
180°
225°
135°
270°
90°
315°
45°
0°
20
40
60
80
100
120
140
160

III. Analysis

From these results, we can reasonably conclude that the images representing the *SAM* system on *Wikipedia* are quite reliable and definitely reproducible. The comparison images are almost exactly identical. Any minor dissimilarities can be explained by differences in algorithmic parameters, such as step-sizes and maximum calculation time.

Moreover, there appear to be some very chaotic trajectories, as evidenced by some of the images at the end. The first set of comparison images exhibit non-singular, bounded and periodic motion. The periodicity is evident by the repetitive motion of the object. In some cases this periodicity is rather complex, while in others it appears as though the object is simply swinging back and forth like a pendulum. For bounded cases we see that the radius is equal to or less than one radius unit. Furthermore, the trajectories are certainly non-singular for the first few cases.

The latter set of comparison images exhibit singular motion. This is evidenced by the path leading to the center of the system (i.e. where $r = 0$), where the program then exits prematurely. The last images (non-comparisons) show that there exist trajectories that are markedly chaotic and the last image is unbounded as well.

IV. Conclusions

With these results, I believe it is appropriate to say with complete confidence that all of the major claims that *Wikipedia* made were factually accurate. To summarize, we first tested the equations of motion that was derived in the article. We recovered the same equations of motion independently, and validated this facet of the article. Secondly, we tested the various trajectories of the system to see if it exhibited characteristics of non-singular, singular, bounded, unbounded, periodic, and chaotic motion. Our results validated these assertions in several different ways. Finally, the *Wikipedia* article contained several images of various trajectories for given values of μ and various initial conditions. We were able to fully reproduce all of these trajectories with an almost perfect match in every case. Small differences can be attributed to differing algorithmic parameters (such as those outlined in *Methods*).

This project could be further expanded to include massive pulleys in the system. Another intriguing option would be to have *two* swinging masses.

Finally, this project was an excellent exercise in validating the oft-acclaimed *Wikipedia* with respect to physics articles. My faith in the website has grown substantially a result of these trials. Perhaps most importantly, ability to apply concepts of programming to the field of physics feels like an enormously powerful tool that I will continue to utilize throughout my education.

V. Appendix:

I. Lagrange's Formulation of Classical Mechanics

The formulation of classical mechanics through the Euler-Lagrange equations is an exercise usually completed by the end of undergraduate studies. Nonetheless, it remains a lightly touched but incredibly powerful tool that many undergraduates underutilize. We develop an introductory formalism to the subject here.

Consider a system where a particle is traveling in free space. We know an initial point and a final point, yielding the values of the velocity and position. Now, instead of considering only a single trajectory from the initial point to the final point, we consider *infinitely many* paths. After all - there are infinitely many ways the particle could get from point A (the initial point) to point B (the final point).

We now must consider a few philosophical points. Newtonian mechanics suggests that we can know every value of the position and momentum to the final point given the initial point, given that we are aware of the forces present in the system. One might conclude then, that there exists some more general mechanics that can *also* determine the position and momentum under the same assumptions. Continuing, we are compelled to believe that given a stationary point in the trajectory between the initial point and final point, we can determine the rest of the trajectory. In other words, the system progresses from one mechanical state to the next by *acting* via physical laws from the stationary point. We now suggest that there exists a dynamic quantity in the system (specific to each distinct trajectory), called the *action*. We also suggest that there exists a function that takes both position and momentum coordinates as arguments, over which this action function (more specifically, a *functional* - taking a function as the argument and returning something else) operates upon. This is analogous to how Newton's kinematic laws of motion take position and velocity coordinates (as well as the forces present) to determine future paths for the object; the only difference is that we're packaging everything into a single function rather than separating seemingly unrelated coordinates (that is, position and velocity). We'll call this enigmatic function the "Lagrangian" function, and it takes position and velocity as arguments. We continue by defining the Lagrangian and the supposed action:

$$\mathcal{L} = \mathcal{L}(q, \dot{q})$$
$$\mathcal{S} = \int_{t_1}^{t_2} \mathcal{L}(q, \dot{q}) dt$$

Note that there are bounds on the integral expressing the action: the action is calculated from the initial point of the given trajectory to the final point of the initial trajectory (i.e. where $t : t_1 \to t_2$. The action will return a different value for each trajectory. A natural question arises: how do we know which trajectory is the "true" trajectory? By "true" trajectory, we specifically mean the trajectory that is most likely to be traversed by the particle according to Newtonian mechanics. Now, it just so happens that nature tends to take the least amount of "action" necessary whilst adhering to physical laws. In other words, the most probable trajectory (i.e. the one that would be given by Newton's mechanics) is the trajectory that yields an extremal value of the action. To retrieve this "true" trajectory mathematically, we enforce the rules of the calculus of variations:

$$\delta \mathcal{S} = 0$$

Utilizing our above equations and continuing the calculus:

$$\begin{aligned}
0 &= \delta \int_{t_1}^{t_2} \mathcal{L}(q,\dot{q})dt \\
&= \int_{t_1}^{t_2} \delta\mathcal{L}(q,\dot{q})dt \\
&= \int_{t_1}^{t_2} \left(\frac{\partial \mathcal{L}}{\partial q}\delta q + \frac{\partial \mathcal{L}}{\partial \dot{q}}\delta\dot{q} \right) dt \\
&= \int_{t_1}^{t_2} \left(\frac{\partial \mathcal{L}}{\partial q}\delta q + \frac{\partial \mathcal{L}}{\partial \dot{q}}\frac{d}{dt}\delta q \right) dt \\
&= \int_{t_1}^{t_2} \left(\frac{\partial \mathcal{L}}{\partial q}\delta q \right) dt + \int_{t_1}^{t_2} \left(\frac{\partial \mathcal{L}}{\partial \dot{q}}\frac{d}{dt}\delta q \right) dt \\
&= \int_{t_1}^{t_2} \left(\frac{\partial \mathcal{L}}{\partial q}\delta q \right) dt + \frac{\partial \mathcal{L}}{\partial \dot{q}}\delta q\Big|_{t_1}^{t_2} - \int_{t_1}^{t_2} \left(\frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \dot{q}}\delta q \right) dt \\
&= \int_{t_1}^{t_2} \left(\frac{\partial \mathcal{L}}{\partial q}\delta q \right) dt - \int_{t_1}^{t_2} \left(\frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \dot{q}}\delta q \right) dt \\
&= \int_{t_1}^{t_2} \left(\frac{\partial \mathcal{L}}{\partial q}\delta q - \frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \dot{q}}\delta q \right) dt \\
&= \int_{t_1}^{t_2} \left(\frac{\partial \mathcal{L}}{\partial q} - \frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \dot{q}} \right) \delta q \quad dt
\end{aligned}$$

Where the second term in the fifth step was evaluated by integration-by-parts. The first term in the evaluation is zero, because the variation in the trajectory at the boundaries is necessarily zero (i.e. $\delta q(t_i) = \delta q(t_f) = 0$). Regarding the last step, the equality can only be true if the quantity in the parentheses must be zero - for if $\delta q = 0$ the entire derivation becomes a triviality. Therefore, we recover the so-called Euler-Lagrange equation utilized in our study of the *SAM* system:

$$\boxed{\frac{\partial \mathcal{L}}{\partial q} - \frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \dot{q}} = 0} \tag{3}$$

Finding the expression $\mathcal{L} = T - U$ (kinetic energy minus potential energy) is left as an exercise to the reader.

II. Derivation of the Equations of Motion for the *SAM* System:

For our purposes here, the pulleys will be assumed to be non-existent and massless, acting as mere tethers for the wire. We begin by establishing the famed *Euler-Lagrange* equation:

$$\frac{\partial \mathcal{L}}{\partial q_i} = \frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \dot{q}_i}, \qquad i \in \mathbb{Z}^+$$

Additionally, we require an explicit expression of the Lagrangian for this system, which incidentally has *two* degrees of freedom:

$$\begin{aligned}\mathcal{L} &= T - U \\ &= T_1 + T_2 - U_1 - U_2\end{aligned}$$

We pause here to define the quantity $\mu \equiv m_2/m_1$, as is customary in *SAM* literature.

$$\begin{aligned}\mathcal{L} &= \left(\frac{1}{2}m_1(\dot{r}^2 + r^2\dot{\theta}^2)\right) + \left(\frac{1}{2}m_2\dot{r}^2\right) \\ &\quad - (m_2 g r - m_1 g r\cos(\theta)) \\ &= \frac{1}{2}(m_1 + m_2)\dot{r}^2 + \frac{1}{2}m_1 r^2\dot{\theta}^2 \\ &\quad - gr\Big(m_2 - m_1\cos(\theta)\Big)\end{aligned}$$

given $(\dot{r}_1)^2 = (-\dot{r}_2)^2 = \dot{r}^2$ in the *KE* term. The above equation needs to be processed through the *E-L* equation to be of any use. It is also of utility to determine the canonical momenta for these respective coordinates:

$$p_i = \frac{d\mathcal{L}}{d\dot{q}_i}$$

We calculate them as follows:

$$\begin{aligned}p_r &= \frac{d\mathcal{L}}{d\dot{r}} \\ &= (m_1 + m_2)\dot{r} \\ p_\theta &= \frac{d\mathcal{L}}{d\dot{\theta}} \\ &= m_1 r^2\dot{\theta}\end{aligned}$$

The canonical momenta are thus:

$$\boxed{p_r = (m_1 + m_2)\frac{dr}{dt}, \qquad p_\theta = m_1 r^2\frac{d\theta}{dt}} \tag{4}$$

II.1 Radial Equation

The radial equation (perhaps obviously) concerns the dynamics of the distance of the pendulum mass from the pulley.

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial r} &= m_1\left(r\dot{\theta}^2 + g\cos(\theta)\right) - m_2 g \\
\frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \dot{r}} &= \frac{d}{dt}\left(m_1\dot{r} + m_2\dot{r}\right) \\
&= (m_1 + m_2)\ddot{r} \\
\Rightarrow (m_1 + m_2)\ddot{r} &= m_1\left(r\dot{\theta}^2 + g\cos(\theta)\right) - m_2 g \\
(1+\mu)\ddot{r} &= r\dot{\theta}^2 + g\cos(\theta) - \mu g
\end{aligned}$$

Recovering the principal radial equation:

$$\boxed{(1+\mu)\frac{d^2 r}{dt^2} = r\left(\frac{d\theta}{dt}\right)^2 + g\cos(\theta) - g\mu} \tag{5}$$

II.2 Angular Equation

The angular equation (perhaps obviously) concerns the dynamics of the angle of the pendulum mass from the pulley.

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial \theta} &= -m_1 g r\sin(\theta) \\
\frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \dot{\theta}} &= \frac{d}{dt}\left(m_1 r^2\dot{\theta}\right) \\
&= 2m_1 r\dot{r}\dot{\theta} + m_1 r^2\ddot{\theta} \\
\Rightarrow -g\sin(\theta) &= 2\dot{r}\dot{\theta} + r\ddot{\theta}
\end{aligned}$$

Recovering the principal angular equation:

$$\boxed{r\frac{d^2\theta}{dt^2} = -g\sin(\theta) - 2\frac{dr}{dt}\frac{d\theta}{dt}} \tag{6}$$

III. Program 1:

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

seterr(all='raise')        #If we encounter overflows, trigger an error
   (that'll be resolved by 'try-catch')

def RKalgorithm():
    global flag
    flag         = False
    variables[0] = array([r0, rdot0, -theta0, -thetadot0])  #Thetas are
        negative to agree with the diagram in the paper that goes along
        with it

    for i in range(indices - 1):
        try:
            if(((theta0 == 0 and thetadot0 == 0) or theta0 == pi) and i
                == 0):
                print('Warning: Unstable or stable equilibrium chosen
                    as initial value. Results may be unintended.')
            if(variables[i][0] <= (threshold * r0)):
                print('The trajectory came within the threshold for
                    identifying a singularity (%.4f%% of r0). The
                    program has finished early (%.2f s) to avoid
                    infinities.' % ((threshold * r0 * 100), (i * step)))
                break

            k1 = step * RKaccel(variables[i], times[i])
            k2 = step * RKaccel(variables[i] + k1 / 2, times[i] + step
                /2)
            k3 = step * RKaccel(variables[i] + k2 / 2, times[i] + step
                /2)
            k4 = step * RKaccel(variables[i] + k3, times[i] + step)

            variables[i + 1] = variables[i] + k1/6 + k2/3 + k3/3 + k4/6

        except FloatingPointError:    #This isn't actually an error,
            but we've told the system to associate OVERFLOWS with errors
            .
            flag = True
            print('A Runtime Warning was triggered, indicating
                infinities as r -> 0. Increase the singularity threshold
                .')
            print('As a result, plotting procedures have been abandoned
                to avoid an erroneous display.')
            break
```

```
def RKaccel(variables, times):
    radius     = variables[0]
    radiusdot  = variables[1]
    theta      = variables[2]
    thetadot   = variables[3]

    radiusdotdot = ((radius / (1 + mu)) * ((thetadot) ** 2)) + (((g *
       cos(theta)) - (g * mu)) / (1 + mu))
    thetadotdot  = - ((g * sin(theta)) / radius) - (2 * ((radiusdot) *
       (thetadot)) / radius)

    return array([radiusdot, radiusdotdot, thetadot, thetadotdot])

#Initialize algorithmic variables.
step      = 0.001
maxtime   = 20
threshold = 0.01   #singularity theshold! read the paper.

indices = int(maxtime / step)
times   = linspace(0, (indices - 1) * step, indices)

#Initialize the initial physical variables.
r0         = 1
rdot0      = 0
theta0     = -pi / 2
thetadot0  = 0
g          = 9.8
mu         = 0.9#4.177 is a nice one.

#Runge-Kutta algorithm variables
variables = zeros([indices, 4], dtype=float)
RKalgorithm()


#Begin plotting
plt.figure()
ax = plt.subplot(111, projection='polar')
ax.set_theta_zero_location("S")
ax.plot(variables[:,2], variables[:,0], color='r', linewidth=1)
plt.title('$SAM$ for $\mu = %.3f$, $r_0 = %.3f$, $\\theta_0 = %.3f^\
   degree$, $t_{max} = %.2f$' % (mu, r0, theta0 * 180 / pi, maxtime), y
    = 1.06)
ax.grid(True)
if(flag == False):
    print('Calculation was successful.')
    plt.show()
```

References

[1] George Atwood, Cambridge (1784). A treatise on the rectilinear motion and rotation of bodies; *Cambridge*, Sect. VII. 292:0-22

[2] Tufillaro, Nicholas B. (1982). Smiles and Teardrops (Senior Thesis) *Reed College*

[3] O. Pujol, J.P. Pérez, J.P. Ramis, et al. (2010). SAM: Experimental and numerical results, and a theoretical study; *Physica D*, 239:1067-1081

[4] `Swinging Atwood's machine: https://en.wikipedia.org/wiki/Swinging_Atwood\%27s_machine`