Introduction

The Sedaro Nano app allows users to simulate the orbit of a satellite around a host planet, using the planet's and satellite's initial positions and velocities as the input. Previously, the app suffered from accuracy problems due to a simple physics model and an inaccurate numerical time-integration scheme. The purpose of this report is to describe the work that was done to implement a more accurate physics model and a more accurate time-integration scheme.

To sum it all up, the previous version of the app uses a basic physics model that considers only the force of gravity, and it steps through its simulation using a basic Euler-Cromer time-integration scheme. The weakness of the basic physics model is that it ignores all forces that may induce orbital decay, and the weakness of the Euler-Cromer integration scheme is that it is has only first-order accuracy.

To improve the accuracy of the app, my work involved implementing a tunable atmospheric drag force on the satellite and implementing a fourth-order Runge-Kutta time-integration scheme. Including a nonconservative force like atmospheric drag on the satellite allows for the simulation to predict the orbital decay of the satellite, and implementing the Runge-Kutta scheme allows for the simulation to have increased stability and accuracy across a larger simulation time-step.

Physics Model

For the purpose of this report, the "mathematical model" is the model for the forces affecting the satellite and the planet. This section will summarize the previous physics model, describe the current physics model, and describe how this simulation models the variation of air density with altitude.

Previous Model

In the previous model, the planet is assumed to move with constant velocity and the satellite is subject only to the force of gravity exerted by the planet. While it is technically inaccurate to assume that the planet moves with constant velocity in a straight line, this is actually of relatively small concern. Even a geostationary satellite that has a relatively slow orbit will complete an orbit in just 1 day, in which time the earth's heading will have changed by approximately 1°. Most satellites occupy the LEO band and take at most 2-hours to complete an orbit, in which time it is a valid simplification to assume the planet is either stationary or moving in a straight line.

The largest problem with this physics model is that the satellite is subject only to the conservative force of gravity, leading to simulations that predict infinitely stable orbits (under specific configurations). If Sedaro's customers are interested in solving real-world problems related to the orbital decay of their satellites, then they would find no value in a simulation that predicts infinitely stable orbits. Thus, it is important to include nonconservative forces on the satellite to simulate a realistic orbital decay.

Physics Model

As mentioned in the previous section, the focus of my contribution to the physics model is to include the effects of nonconservative forces on the satellite. In reality, satellites can be exposed to effects of aerodynamic drag, electromagnetic drag, collisions with debris, and solar radiation pressure.

For the purposes of this simulation, I chose to implement only the effects of atmospheric drag. I chose to focus on this effect as my reading indicates that this is the largest contributor for orbital decay for LEO satellites, meaning that it ought to be the first force that should be modeled to make the biggest improvement in the accuracy of Sedaro Nano. The force of drag is shown below in Equation 1:

$$F_D = \frac{1}{2}\rho(R)V^2C_DA = \frac{\pi}{8}\rho(R)V^2C_DD^2 \tag{1}$$

where ρ is air density $[kg \cdot m^{-3}]$, R is the altitude of the satellite $[m]$, V is satellite velocity $[m/s]$, C_D is the unitless coefficient of drag, and D is the diameter of the satellite $[m]$. This model assumes the satellite to be a sphere with diameter D, and it assumes that the coefficient of drag is constant.

It is also important to mention that this physics model considers that the force exerted *by* the satellite *on* the planet is negligible. This is valid in cases when the mass of the planet vastly exceeds the mass of the satellite, which is satisfied in nearly all cases of man-made satellites orbiting planetary bodies.

Atmospheric Model

The purpose of the atmospheric model in this simulation is to model the variation in air density at different altitudes. In reality, atmospheric models can be quite complicated. Air density is determined by the temperature and pressure of air in the atmosphere, and these properties themselves have complex variations with altitude. For example, temperature decreases with altitude in some regimes and increases in others, while air pressure exponentially decays at different rates at different regimes.

For the purposes of this simulation, I assume a very simple exponential model for the variation of air density with altitude. In this model, I assume that the air density is equal to the air density at sea level ρ_0 when the orbital radius R is equal to zero, and I assume that the air density is equal to one-millionth of ρ_0 at the initial orbital radius R_0. Further, I assume an exponential relationship between these two points.

$$\rho(0) = \rho_0 \;\; ; \;\; \rho(R_0) = \rho_0 * 10^{-6}$$

$$\rho(R) = \rho_0 \exp\left(\ln\big(\rho(R_0)\big) \cdot \frac{R}{R_0}\right) = \rho_0 \exp\left(-11.5 \cdot \frac{R}{R_0}\right) \tag{2}$$

This exponential relationship induces a positive feedback loop for the atmospheric drag force. As the satellite experiences drag, its orbit will decay to a lower altitude. As the satellite's altitude decreases, the air becomes denser, and the effect of atmospheric drag increases. Increased drag leads to accelerated decay, et cetera.

Including this in the drag force equation results in a new drag equation:

$$F_D = \frac{\pi}{8}\rho_0 \exp\left(-11.5 \cdot \frac{R}{R_0}\right) V^2 C_D D^2 \tag{3}$$

In the code for the drag force, the air density at R_0 also becomes a tunable parameter to the system. Turning this "knob" allows a designer to specify the air density conditions of the satellite's starting point.

Mathematical Model

The purpose of the mathematical model is to translate the physics model into a form suitable to being solved by the numerical time-integration scheme. The major thrust of this effort is to separate the forces into their x- and y-components and to nondimensionalize the equations of motion to make them suitable for numerical simulation.

X- and Y-components of gravity and drag

Splitting the drag and gravity forces into their x- and y-components requires the diagram in Figure 1. Gravity always acts in the direction of the host planet and drag always acts opposite the direction of motion.

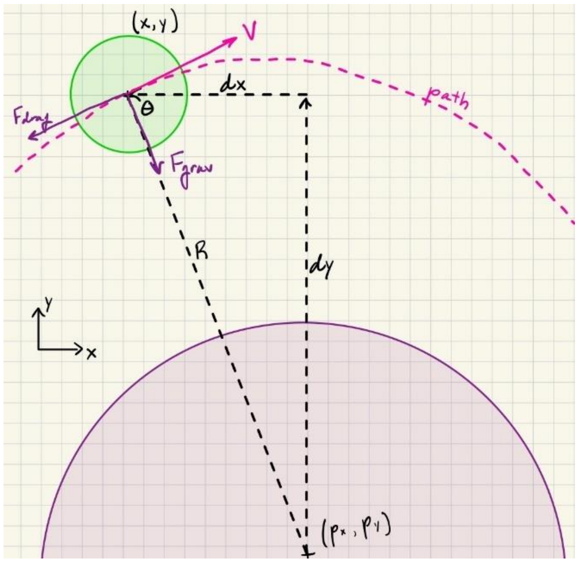

Figure 1 – Diagram for the planet and orbiting satellite. The purple circle is the planet, and the green circle is the satellite.

Using the angle θ between the planet and the satellite, we can project the forces onto the coordinate system:

$$\begin{aligned}\Sigma F_x &= F_g \cdot \cos\theta + F_D \cdot \sin\theta \\ &= F_g \cdot \frac{dx}{R} + F_D \cdot \frac{dy}{R}\end{aligned}$$

$$\begin{aligned}\Sigma F_y &= F_g \cdot \sin\theta - F_D \cdot \cos\theta \\ &= F_g \cdot \frac{dy}{R} - F_D \cdot \frac{dx}{R}\end{aligned}$$

Non-dimensionalizing the equations of motion

Nondimensionalization is a crucial step for implementing a simulation. First, nondimensionalizing the equations and collecting terms into non-dimensional numbers (the most famous example being the Reynolds number from fluid dynamics) produces a very convenient “knob” that designers can turn when carrying out a simulation. Second, a well-thought nondimensionalization re-scales parameters of interest down to a smaller scale closer to $O(1)$. The value of this rescaling is that computers “like” smaller numbers because the floating-point representation used by computers actually loses precision as the magnitude of a number becomes very large. By rescaling simulation parameters / variables down closer to $O(1)$, we are conducting the simulation in the regime with the highest numerical precision and stability.

The previous version of Sedaro Nano used a nondimensional representation of the gravity force, whether explicitly or implicitly. As we will see in this section, the planet’s mass and the universal gravitational constant can be rolled up into what I call the “gravity number” Gr, and the previous version of Sedaro Nano sets Gr equal to 1. My addition is to implement a nondimensionalized drag force governed by a “drag number” Dr that governs the contribution due to drag.

First, we begin with the dimensional equations of motion:

$$m\ddot{x} = \Sigma F_x = \frac{GMm}{R^2} \cdot \frac{dx}{R} + \frac{\pi}{8}\rho(R) \cdot (\dot{x} + \dot{y})^2 C_D D^2 \cdot \frac{dy}{R}$$

$$m\ddot{y} = \Sigma F_y = \frac{GMm}{R^2} \cdot \frac{dy}{R} - \frac{\pi}{8}\rho(R) \cdot (\dot{x} + \dot{y})^2 C_D D^2 \cdot \frac{dx}{R}$$

Next, we define new nondimensional variables based on some characteristic length and time scales denoted by $\cdot^*$:

$$x^* = x/L \;\therefore\; x = x^* L$$

$$y^* = y/L \;\therefore\; y = y^* L$$

$$t^* = t/T \;\therefore\; t = t^*T$$

$$R = \sqrt{x^2 + y^2} = \sqrt{(x^*L)^2 + (y^*L)^2} = \sqrt{L^2 \cdot \left({x^*}^2 + {y^*}^2\right)} = L \cdot \sqrt{{x^*}^2 + {y^*}^2} = L * R^*$$

Based on these nondimensional variables, we can also define nondimensional rates of change of these variables:

$$\dot{x} = \frac{dx}{dt} = \frac{d(x^*L)}{d(t^*T)} = \frac{L}{T} \cdot \frac{dx^*}{dt^*} = \frac{L}{T} \cdot \dot{x}^*$$

$$\dot{y} = \frac{dy}{dt} = \frac{d(y^*L)}{d(t^*T)} = \frac{L}{T} \cdot \frac{dy^*}{dt^*} = \frac{L}{T} \cdot \dot{y}^*$$

$$\ddot{x} = \frac{d^2x}{dt^2} = \frac{d^2(x^*L)}{d(t^*T)^2} = \frac{L}{T^2} \cdot \frac{d^2x^*}{d{t^*}^2} = \frac{L}{T^2} \ddot{x}^*$$

$$\ddot{y} = \frac{d^2y}{dt^2} = \frac{d^2(y^*L)}{d(t^*T)^2} = \frac{L}{T^2} \cdot \frac{d^2y^*}{d{t^*}^2} = \frac{L}{T^2} \ddot{y}^*$$

Next, sub these into the equations of motion:

$$\frac{mL}{T^2} \cdot \ddot{x}^* = \frac{GMm}{L^2 \cdot {R^*}^2} \cdot \frac{L \cdot dx^*}{L \cdot R^*} + \frac{\pi}{8} \rho(R) \left(\left(\frac{L}{T} \dot{x}^*\right)^2 + \left(\frac{L}{T} y^*\right)^2 \right) C_D D^2 \cdot \frac{L \cdot dy^*}{L \cdot R}$$

$$\rightarrow\; \ddot{x} = \frac{GMT^2}{L^3} \cdot \frac{dx^*}{{R^*}^3} + \frac{\pi \rho(R) C_D L D^2}{8m} \cdot \left({\dot{x}^*}^2 + {\dot{y}^*}^2\right) \cdot \frac{dy^*}{R^*}$$

$$\rightarrow\; \ddot{x} = \frac{GMT^2}{L^3} \cdot \frac{dx^*}{{R^*}^3} + \frac{\pi \rho_0 C_D L D^2}{8m} \cdot \exp(-11.5 \cdot R^*) \cdot \left({\dot{x}^*}^2 + {\dot{y}^*}^2\right) \cdot \frac{dy^*}{R^*}$$

Similarly:

$$\rightarrow\; \ddot{y} = \frac{GMT^2}{L^3} \cdot \frac{dy^*}{{R^*}^3} - \frac{\pi \rho_0 C_D L D^2}{8m} \cdot \exp(-11.5 \cdot R^*) \cdot \left({\dot{x}^*}^2 + {\dot{y}^*}^2\right) \cdot \frac{dx^*}{R^*}$$

Let:

$$Gr = \frac{GMT^2}{L^3}$$

$$Dr = \frac{\pi \rho_0 C_D L D^2}{8m}$$

Finally, substitute the nondimensional numbers into the nondimensionalized equations of motion:

$$\ddot{x} = Gr\left[\frac{dx^*}{R^{*3}}\right] + Dr\left[\exp(-11.5 \cdot R^*) \cdot \left(\dot{x}^{*2} + \dot{y}^{*2}\right) \cdot \frac{dy^*}{R^*}\right]$$

$$\ddot{y} = Gr\left[\frac{dy^*}{R^{*3}}\right] - Dr\left[\exp(-11.5 \cdot R^*) \cdot \left(\dot{x}^{*2} + \dot{y}^{*2}\right) \cdot \frac{dx^*}{R^*}\right]$$

The interpretation for the exact "meaning" of Gr and Dr is where nondimensional analysis becomes more of an art than a science. For example, what exactly should we consider the characteristic length L? What is the characteristic time scale T? Or is it more appropriate to consider a characteristic frequency f and set $T = 1/f$? The answers to these questions generally have to do with re-scaling the simulation variables down towards $O(1)$ while also picking "sensible" answers for the length and time/frequency scales. Further, there is not always just a single answer, which is apparent to fluid dynamicists in the various flavors of Reynolds numbers (Re_D, Re_L, Re_μ, etc.).

For our purposes, we are just happy to have a "knob" to turn to simulate higher or lower forces from gravity and drag. Observe that setting Dr equal to zero returns the model to the initial state of Sedaro Nano's basic physics model. The initial version of Sedaro Nano simply sets Gr equal to unity (and it's what I use for the simulations in this report), but the above equations give us a more accurate way to determine an appropriate value for Gr.

Numerical Solver

The numerical solver is the process by which we solve for the mathematical model at discrete timesteps. This is a crucial step because implementing more complex models can result in greater accuracy and stability. Although a more complex models may require more computation per timestep, they often allow the simulation to run faster by taking larger timesteps.

Previous Solver

The initial version of Sedaro Nano implements the Euler-Cromer time-integration scheme. This is a semi-implicit scheme that uses the current acceleration a_n to solve for the future velocity v_{n+1}, and then uses the future velocity v_{n+1} to solve for the future position x_{n+1}.

$$a_n = f\left(\vec{x}, \dot{\vec{x}}, \vec{p}, \dot{\vec{p}}, t\right)$$

$$v_{n+1} = v_n + a_n \cdot \Delta t$$

$$x_{n+1} = x_n + v_{n+1} \cdot \Delta t$$

This is a slight (but powerful) modification to the basic Euler approximation that uses the current velocity v_n to solve for the future position x_{n+1}. This solver has been shown to conserve energy for oscillatory problems, unlike the basic Euler approximation, which is useful for an oscillatory system like planetary orbit. However, this method has the downside that it has only "first-order accuracy", meaning that the error term of the approximation decreases only linearly with the step size. The major focus of the improvements in the numerical solver will be to implement a scheme with higher-order accuracy.

Fourth-order Runge-Kutta

For this project, I chose to implement the fourth-order Runge-Kutta (RK4) time-integration scheme. This scheme has the benefit that it has fourth-order accuracy, meaning that the error decreases proportionally with Δt^4. Therefore, reducing the error by one-half requires reducing the timestep to only 84% of its original value, whereas the Euler-Cromer algorithm would require reducing the timestep to 50% of its original value. Another valuable quality of the RK4 scheme is that it is “self-starting”, meaning that it requires only information about the current timestep to compute an estimate for the future timestep. It requires 4 computations per simulation variable per timestep, resulting in 16 computations per timestep (plus 2 to account for the motion of the planet).

I have decided not to include the RK4 equations for the sake of keeping this document at least somewhat concise.

Results

In general, the result of this implementation is to make the solver more accurate and stable.

Regarding stability, the initial version of Sedaro Nano predicts that the orbit of the satellite actually increases in magnitude when exposed only to the effect of gravity, as seen in Figure 2 below. This is unlikely to be accurate, and it indicates that the numerical solver is increasing the numerical energy in the system. Over time, the planet’s orbit will increasingly depart from the planet until reaching some sort of escape velocity. After implementing the RK4 solver, the simulator predicts that the satellite will be held in a stable orbit, as shown in Figure 3.

Regarding physical accuracy, the new solver actually includes a tunable atmospheric drag force in addition to the tunable gravitational force. Figure 4 used a simulation with an exaggerated drag force, and it can be observed that the satellite plummets down toward the host planet when drag is very large. Alternatively, Figure 5 used a simulation with a smaller drag force, and it can be observed that the satellite is relatively unaffected by drag until the satellite undergoes many orbits. Due to the nature of drag’s positive feedback loop, it will not take many more cycles for the satellite to crash into the planet.

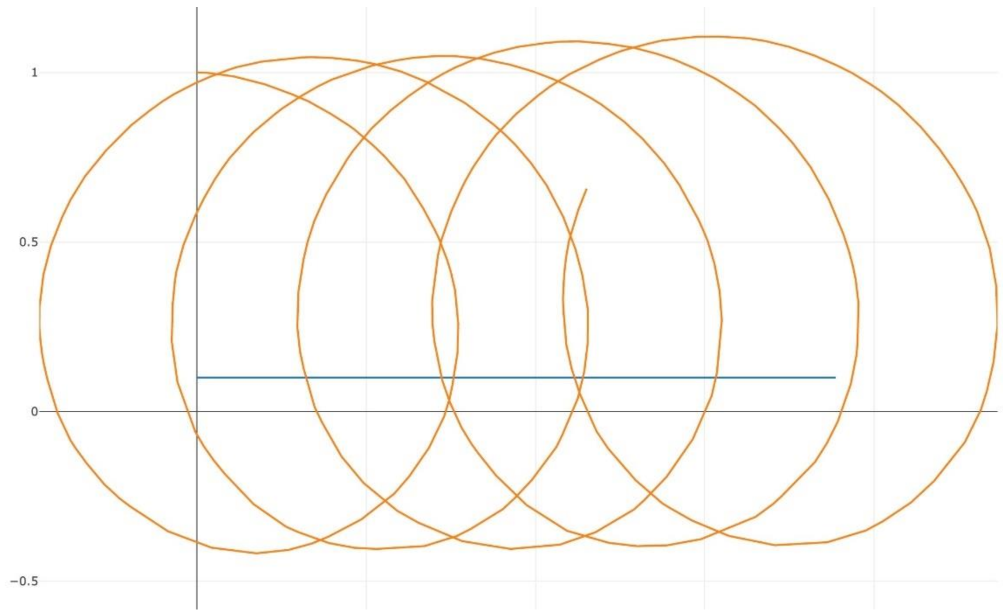

Figure 2 – Simulation from original version of Sedaro Nano. Observe that the orbit generally drifts away from the planet despite only being exposed to the conservative gravitational force.

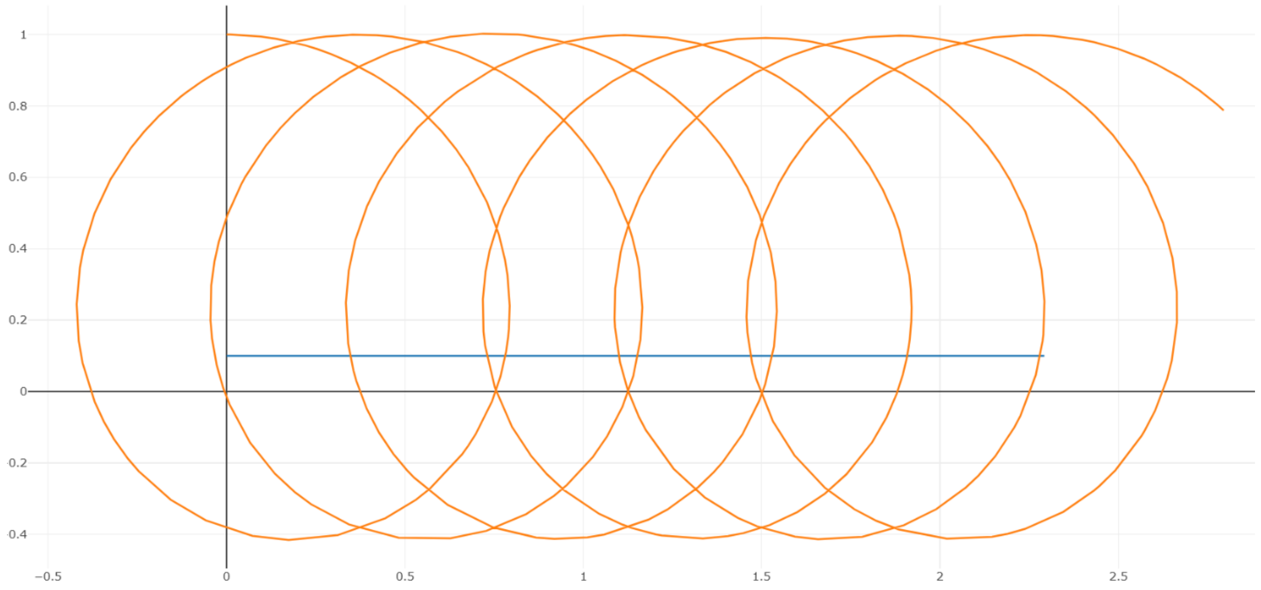

Figure 3 – Simulation from the updated version of Sedaro Nano, $Gr = 1, Dr = 0$. The timestep size is exactly the same, and the new version predicts a stable orbit.

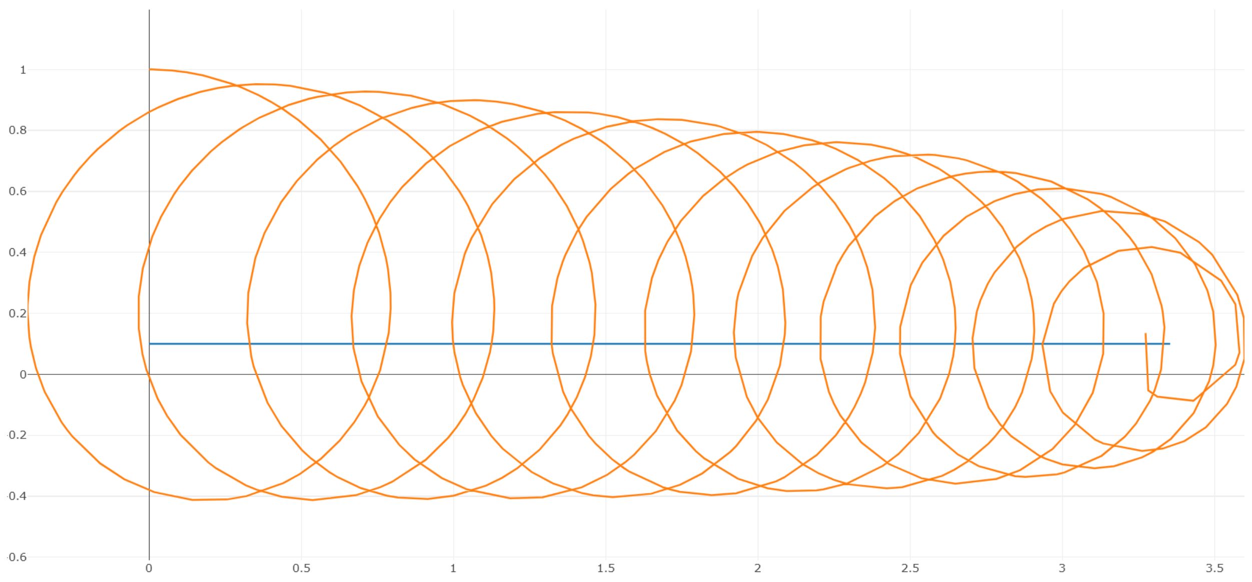

Figure 4 – Simulation from updated Sedaro Nano, $Gr = 1, Dr = 10$. *This simulation has an exaggerated force of drag to show the orbital decay due to atmospheric drag.*

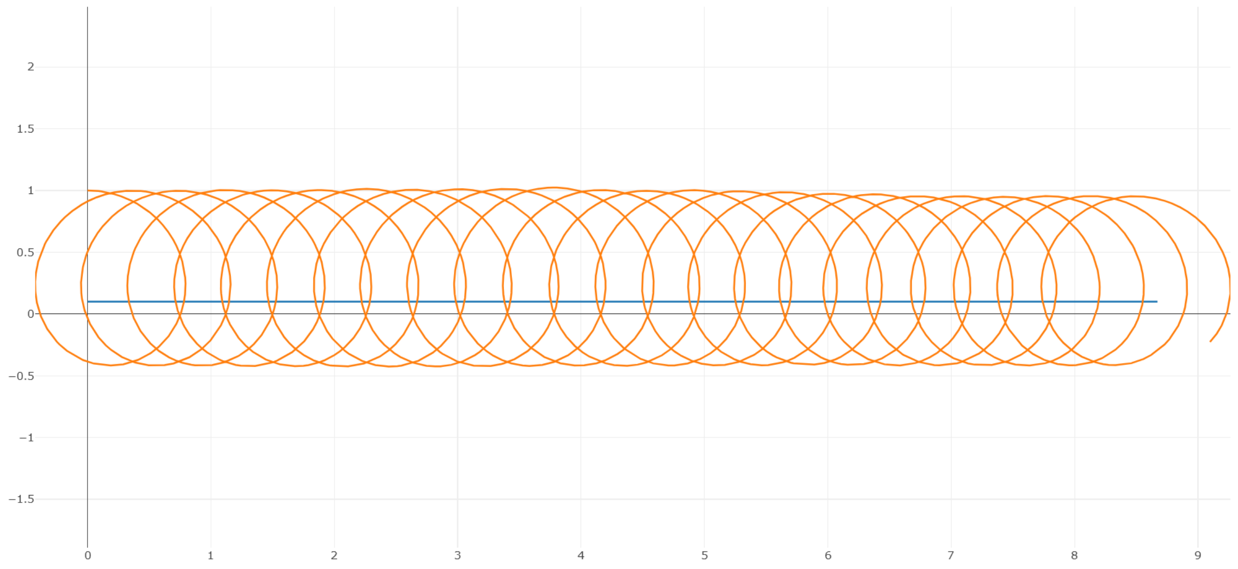

Figure 5 – Simulation from updated Sedaro Nano, $Gr = 1, Dr = 1/4$. *This simulation shows the satellite has a relatively stable orbit, but the effects of drag begin to severely affect the satellite's orbit near the end of this trial.*