SPACESHOT LOOP DYNAMICS

CHANDLER WATSON (@UWU)

The Spaceshot Loop is ~~the bane of our collective existence~~ a phenomenon repeatedly observed in the Spaceshot spin-launch system: the fin-less rocket when launched at a high spin rate often does a single loop in the air before proceeding upwards. Every time this has been observed, the rocket proceeds (generally) upward afterward. Several hypotheses have been put forward on why this happens, but no data or simulation exists yet supporting one hypothesis or another.

1. The Big Idea

Many reasons proposed for why the Spaceshot Loop happens involve aerodynamics. If we can rule out aerodynamic effects as a cause for the Spaceshot Loop, we can save ourselves a lot of work in the long run. This document describes how to make a simplified simulation of a spinning rocket that ignores aerodynamics. If we can recreate the Spaceshot Loop with it, we'll be able to narrow down why the Spaceshot Loop happens.

If we can't, we'll add new features (like some basic aerodynamic terms) until we see the Loop.

2. How to Math: Dynamics 2, Electric Bugaloo

We'll be modeling the rocket as a spinning cylinder with thrust: force applied to the bottom of the cylinder, no matter what way it's facing. Imagine spinning a pencil on your finger (insanely fast) and pushing it by the eraser.

Now, how do we simulate this? Many students in physics classes start by drawing what's called a "free body diagram." They determine all the forces on the object they care about, draw which way they act, and add them together. While this works well for most problems, it can become very difficult when dealing with spinning objects.

For this reason, we introduce **Lagrangian dynamics**, which are actually *way easier* than their name makes them sound. It's effectively a physics superpower that lets you find the equations of motion for a system–say, a rocket–given basic information on it.

What is a Lagrangian? Here, have one! (We write $\dot{x}$ for the time derivative of x-position, i.e. velocity.)

$$L = \frac{1}{2}m\dot{x}^2.$$

It's nothing more than a function. A function of time, since x is a function of time. This Lagrangian describes a free mass m moving along a 1-dimensional line. You might recognize the $\frac{1}{2}mv^2$ as kinetic energy–more on that soon. How do we use the Lagrangian for a system? We solve the Euler-Lagrange equation:

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{x}}\right) - \left(\frac{\partial L}{\partial x}\right) = 0.$$

Any choice of position $x(t)$ determines L. There is a unique[1] $x(t)$ such that the above equation is satisfied, and that describes what our free mass actually does.

Without further ado, let's actually solve this thing:

$$\frac{\partial L}{\partial \dot{x}} = \frac{\partial}{\partial \dot{x}}\left(\frac{1}{2}m\dot{x}^2\right) = m\dot{x}.$$

Date: February 22, 2019.

[1]Well, usually unique. There's some slight shortcomings to the Lagrangian formulation, but those are few and far between.

and

$$\frac{\partial L}{\partial x} = \frac{\partial}{\partial x}\left(\frac{1}{2}m\dot{x}^2\right) = 0$$

where the second derivative is zero because x alone (different from $\dot{x}$) does not appear in the expression–it's a constant as far as looking for xs is concerned. Note that we treat x and $\dot{x}$ as completely separate variables when taking partial derivatives: when we take a partial derivative with respect to one, we *completely ignore* the other.

The above becomes

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{x}}\right) - \left(\frac{\partial L}{\partial x}\right) = \frac{d}{dt}(m\dot{x}) - 0 = m\ddot{x},$$

i.e. the Euler-Lagrange equation says

$$m\ddot{x} = 0.$$

Dividing out by m, we have $\ddot{x} = 0$, i.e. Euler-Lagrange tells us our free particle has zero acceleration–it's just moving along at a constant speed. Which is correct–this is just Newton's first law, i.e. an object in motion will stay in motion! What this speed is depends on how fast the object was moving when we started at $t = 0$ (our "initial conditions"). Thus, with this mysterious L I conjured up, and the mysterious Euler-Lagrange equation, we have *derived* what happens to a particle in 1-dimension!

Now, let's get rid of the mystery. The Lagrangian looked a lot like kinetic energy, right? That's because for a *single particle*, in general, we have

$$L = T - V$$

where L is our Lagrangian, T is our kinetic energy, and V is our potential energy. Since a free particle has no potential energy, our Lagrangian *was* just the kinetic energy. How about the Euler-Lagrange equations?

If you don't really care, skip this paragraph. But if you're curious, read on, and ping me if you're curious about anything. The solutions of the Euler-Lagrange equations are exactly the equations of motion that minimize what's called the *action*:

$$S = \int_{t_1}^{t_2} L\,\mathrm{d}t$$

i.e. our particle "instinctively" takes the route that minimizes action. How does the particle know to do that? Well, the particle doesn't–for different cases (i.e. rigid bodies, single particles, etc.) we end up choosing a *different form for the Lagrangian* so that minimizing the action gives you the same physics that any other method would. This is just the kind of thing where Ye Old Physicists saw the same thing over and over, and eventually made it a rule: for most systems, you can choose a Lagrangian such that minimizing the action just works.[2] If you're exceptionally curious, there are derivations online that show how solving the Euler-Lagrange equations minimizes action, but I leave those out for "brevity" (my way of saying "I don't like deriving things").

A brief recap: the *Lagrangian is essentially the DNA of our system.* It's a very short description of our system that, when paired with the Euler-Lagrange equation, unravels into our equations of motion. Rather than trying to take a difficult system and keep track of all the forces in a weird frame of reference, just use the Lagrangian for the situation, solve Euler-Lagrange, and call it a day.

[2]A couple things. First, it's not really minima we're after: it's actually critical points (so maxima too, and all points at which the derivative of the action is zero) that we care about. Also, for background on why the heck we use the Principle of Least Action, see here: `https://physics.stackexchange.com/questions/15899/why-the-principle-of-least-action`.

3. Spin Things Already, Sheesh

Okay, okay, calm down. You're busy, I get that. Here, have another Lagrangian:

$$L = \frac{1}{2}m(\dot{x}^2 + \dot{y}^2 + \dot{z}^2) + \frac{1}{2}I_1(\dot{\phi}\sin\theta + \dot{\theta}) + \frac{1}{2}I_3(\dot{\phi}\cos\theta + \dot{\psi})^2.$$

Hah. Gotcha. You thought there was gonna be a ton more math. That's actually the Lagrangian that we use in the sims. You made it here already, and we're pretty much almost done. Some symbols to introduce:

- I_1: moment of inertia perpendicular to axis of rocket
- I_3: moment of inertia along axis of rocket
- ϕ, θ, ψ: tip, tilt, and turn of the rocket (for those who like technical words, these are the Euler angles).

It turns out this isn't any special, application-specific equation. It's just the Lagrangian for a rigid body that's symmetric along one axis. You can find it in most textbooks on Lagrangian mechanics.[3]

So how do we use this guy? Easy! We use the Euler-Lagrange equation again. Since there's more dimensions, though, the Euler-Lagrange equation becomes the Euler-Lagrange *equations*–one for each coordinate. Additionally, if you have external forces (namely thrust and gravity!) you can add them to the right hand side of the Euler-Lagrange equations before you solve. So we have:

$$\begin{aligned}
\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{x}}\right) - \left(\frac{\partial L}{\partial x}\right) &= T_x \\
\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{y}}\right) - \left(\frac{\partial L}{\partial y}\right) &= T_y \\
\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{z}}\right) - \left(\frac{\partial L}{\partial z}\right) &= T_z + F_g \\
\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\phi}}\right) - \left(\frac{\partial L}{\partial \phi}\right) &= 0 \\
\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta}}\right) - \left(\frac{\partial L}{\partial \theta}\right) &= 0 \\
\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\psi}}\right) - \left(\frac{\partial L}{\partial \psi}\right) &= 0
\end{aligned}$$

where T_x, T_y, T_z are the components of thrust in the x, y, z directions, and F_g is the force due to gravity. This seems like a lot to solve, right? Six equations? Once we put our massive Lagrangian in there, this is going to get messy.

The good news is that we don't care! Many software libraries (such as Mathematica's `EulerEquations` which I used in the original conception of the code, or the Python library SymPy's `LagrangesMethod`) allow you to simply toss in your Lagrangian and get out the Euler-Lagrange equations, no effort needed. After doing that, we can simply use a standard differential equation solver (in the original code, I use Mathematica's `NDSolve`) to solve the system. Friends don't let friends solve systems of differential equations by hand.

Some last details: first off, what are the components of thrust in each direction? I could derive them here, or–

$$\begin{aligned}
T_x &= T\sin\theta\sin\phi \\
T_y &= T\cos\phi\sin\theta \\
T_z &= T\cos\theta
\end{aligned}$$

Ohp, too late. Looks like I just gave them to you. T here is the magnitude of thrust, and in future versions of the code we can have it vary with time to match the thrust curve of various motors. F_g from above is just $-mg$, as you might expect.

[3] I yoinked this one from my personal favorite, Landau and Lifshitz. It also happens to be the textbook for Physics 120, which is an outstanding (if not a little tricky) class.

4. What Next?

So you've persevered all the way through three pages of my writing. Clearly you've got some willpower to you. Where do we go from here? Well, first we need to implement this all (either in Python with SymPy, or Mathematica, or anything–I'm okay with whatever, though Python might be easiest) and get the very basic model here working as described. Fingers crossed, we should have a spinning rocket that behaves exactly like any other rocket, because, well, if we shoot it straight up and down, it really doesn't matter if it's spinning.

Then we'll add some perturbations. Make the thrust misaligned (i.e. skew it so it doesn't go directly out the back of the rocket, but maybe at a slight angle). Play with things and make the rocket unstable. Then try to see if there's a fixed speed, or some conditions under which our very simplified rocket model is stable or not. For kicks (and also science) we should probably compare what we find to conditions for stability listed in books like *Modern Exterior Ballistics*.

From there, we start hunting for the infamous Spaceshot Loop. See if there's conditions that allow us to create what looks like that infuriating little corkscrew we all know and love. If so, we very well have cracked the code, and can start doubling down by (ideally) working toward launches with spin speeds that would confirm our findings. If not, we start exploring the jungle of possible other things to account for in a model for the Spaceshot Loop. Do we add basic aerodynamic terms like you might find in OpenRocket? Do we explore unusual things like skin friction or the Magnus effect (which I should probably Google and understand at some point)? Do we need to switch to a completely different simulation that can do aerodynamics at a fluid dynamics level?

There are lots of big and exciting questions to ask. We're paving some exciting ground, even just building this first-stage simulation. I'm excited. I hope y'all are too.

Let's go to space, folks. ■