Spline-Based Flight Path Planning and Following for Aerial Navigation

Nathan D. Kim*, Sanchir Erdenebat†, Kyle Law‡
Propulsive Landers at the Georgia Institute of Technology, Atlanta, Georgia, 30332

Optimizing flight paths for fuel efficiency in the context of airborne systems while maintaining strict control over endpoint velocity is uniquely challenging. Linear paths fail because of the strict angle constraints they present. Instead, we leverage the endpoint behavior control of Hermite splines and the intuitive design of Bézier curves to create a smooth flight path for our single-engine test vehicle that gives us precise control over our endpoint state and fuel consumption. To follow the path, we use an intuitive iterative check algorithm to validate which points were the most efficient to target while still staying on the path to reach our desired endpoint. The computational and physical efficiency of our algorithms as well as their adaptability and robustness were validated in MATLAB and Python.

I. Introduction

The shortest direct path between any two points in space is a straight line connecting them. However, real-world constraints often make this shortest path impractical or suboptimal. In rocketry and aerial navigation, constraints like angular limitations and unpredictable external forces often cause straight-line paths to be inefficient or even impossible. Therefore, splines are often used to interpolate between endpoints, resulting in a curve defined by a set of polynomials. Splines are defined by fixed endpoints and intermediate control points that define the curve's structure. Two common spline families are Hermite and Bézier curves, which are both third-degree splines, but they differ in their defining properties. Hermite curves are uniquely characterized by their use of end-point derivatives, allowing users to control the curve's beginning and ending properties. In contrast, Bézier curves are defined by their efficient point calculations along the curve, offering a different approach to shaping and manipulating curves. When applied to flight path optimization, combining the properties of these curves gives one strong control over the endpoints of the flight path while minimizing the computational requirements.

Rockets designed for vertical take-off and landing (VTOL) are faced with the challenge of having to be vertically oriented throughout the entirety of their flight. If a VTOL rocket were to exceed its maximum angle of tilt relative to the vertical axis, then it would lead to loss of control. Designing a path-planning and following algorithm to operate within these tight constraints while accounting for unexpected events (turbulence, etc) can be challenging. In a study by Boris Lau et al., Bézier splines were used to generate a set of continuous segments, which formed a path that allowed their robot to replan their trajectories in the event that they faced unmapped obstacles [1]. They were able to execute precise path corrections that maintained path continuity in real time.

This paper presents a path planning algorithm for VTOL rocket flight that expands upon a Bézier spline's smooth and continuous interpolation by using Hermite splines, which allow for real-time flight data to be accounted for in path creation. We will convert the inputs of the Hermite equation into inputs that can be used in the more efficient Bézier equation so that it outputs an identical path. Then, a target point along the path is found by an iterative algorithm that checks for a point along the path that is both reachable and closest to our rocket.

II. Path Planning

A. Path-Planning and Following Constraints

Unique constraints to VTOL rockets include resource efficiency (fuel and energy), longevity, and reusability. Additionally, our VTOL rocket will require precise navigation because of our goal of hovering in the air. This added precision puts extra strain on the resources used. Therefore, the primary figures of merit in flight path optimization are

*Guidance, Navigation, and Controls (GNC) Vice-Lead, Georgia Tech College of Computing, AIAA Undergraduate Student Member, 1810782
†GNC member, Georgia Tech Woodruff School of Mechanical Engineering, AIAA Undergraduate Student Member, 1810360
‡GNC member, Georgia Tech College of Computing, AIAA Undergraduate Student Member, 1810786

precision and resource management to minimize operational costs. The rocket's velocity, at the time of creating a new path, limits the shape of the beginning of the curve. This is taken into account with Hermite splines, so they are able to output an ideal path that is followable, smooth, and continuous.

B. Path Creation

A big concern for smaller rockets is the unpredictability of wind patterns. A small, sudden gust of wind is enough to throw a rocket completely off path if it does not have the right correction algorithms. In our case, we define a maximum distance and angular error threshold from our current position to our desired path that, when crossed, triggers the creation of a new path. When creating a new path, we must first consider whether our original endpoint is still physically reachable. The main concern regarding this is the angle between our current position at the time of creating a new path and the endpoint. If this exceeds our maximum angle constraint, then our current endpoint is no longer reachable given our current state. So, we must define a new endpoint that is higher up and able to be reached. Once the endpoint is defined and verified, we move on to creating the actual path.

Hermite splines require 4 inputs: a starting point p_0, starting tangent, or starting velocity, v_0, endpoint p_1, and end tangent v_1. The iterative calculation for v_1 is shown in the subsequent section. All of these values are defined as a point (x, y, z) in the global coordinate system. Using this, we can define a parametric equation using the parameter $0 \leq t \leq 1$, where $t = 0$ represents the starting point and $t = 1$ the ending point. Using this, a Hermite spline is defined as:

$$H(t) = (2t^3 - 3t^2 + 1)p_0 + (t^3 - 2t^2 + t)v_0 + (-2t^3 + 3t^2)p_1 + (t^3 - t^2)v_1 \tag{1}$$

This returns the coordinates of a point along the curve, H, at a given t value. With Hermite splines, the tangents v_0 and v_1 influence the shape of the resulting curve. v_0 determines how the beginning of the path is shaped by "pulling" the path towards itself, while v_1 determines how the path approaches the endpoint.

The Bézier spline is calculated by defining the following in the global coordinate system: endpoints p_0 and p_1 and intermediate control points c_0 and c_1. Using these points, the Bézier equation is defined as

$$B(t) = (1-t)^3 p_0 + 3(1-t)^2 t c_0 + 3(1-t)t^2 c_1 + t^3 p_1 \tag{2}$$

We see that computing Hermite splines requires additional space and time to complete, as it requires more computational steps. In our calculations, computational efficiency is dominated by terms raised to a power. Hermite (eq. 1) calculations require 8 terms raised to a power, while Bézier (eq. 2) calculations only require 4 terms to be raised to a power, making them faster to run.

So, to leverage both equations, one must convert the 2 tangents used in Hermite spline calculations into the 2 interior control points used in the Bézier equation. One can directly calculate c_0 using v_0 and p_0 using the following equation:

$$c_0 = p_0 + \frac{1}{3}v_0{}^* \tag{3}$$

Our VTOL rocket will always hover at the end of the path, resulting in $v_1 = 0$. This translates to the end of the path approaching the endpoint in a non-vertically aligned manner. So, we cannot use an end tangent of $v_1 = 0$ in the c_1 calculation. Instead, we want to find a point that results in a minimally curved path when approaching the end. Therefore, one must find the values of t at which each control point has its highest level of control over the shape of the curve. To do this, we first want to find the value of t that maximizes the coefficient of each point.

For example, in Equation (2), using c_1 we want to maximize its coefficient: $3(1-t)t^2$. By distributing the terms, setting the derivative to 0 to find critical points, and solving for t, one finds:

$$\frac{d}{dt}[3t^2 - 3t^3] = 6t - 9t^2 = 3t(2 - 3t) = 0 \Rightarrow t = 0, \frac{2}{3} \tag{4}$$

With these critical points, we plug them back into our original coefficient equation to find which results in a larger value.

$$\begin{aligned} t = 0 &\Rightarrow 3(0)^2 - 3(0)^3 = 0 - 0 = 0 \\ t = \frac{2}{3} &\Rightarrow 3(\frac{2}{3})^2 - 3(\frac{2}{3})^3 = \frac{4}{3} - \frac{8}{9} = \frac{4}{9} \end{aligned} \tag{5}$$

*The resulting Bézier path using c_0 will be identical to the Hermite path using v_0.

Therefore, $t = \frac{2}{3}$ is our maximum. Following the same process for the remaining coefficients:

- p_0 has full control at $t = 0$
- c_0 has the maximum control at $t = \frac{1}{3}$
- c_1 has the maximum control at $t = \frac{2}{3}$
- p_1 has full control at $t = 1$[†]

A point that has full control at a given t means the curve is determined only by that point. So, at their given t value, the curve is directly where the control point is. Maximum control is when a point is the one with the highest level of influence over the shape of the curve; however, all other points still maintain some level of influence over the shape. Each t-step is interpreted as moving from t_a to t_b, where $0 \leq a < b \leq 1$. So, we want to put c_1 in a place that lines up with its t value of maximum control. This minimizes the "pull" that c_1 has over the curve, which reduces the sharpness of the path. To achieve this, we want c_1 to be $\frac{2}{3}$ of the way up along the path, giving us the equation

$$c_1 = (p_{0x}, p_{0y}, p_{0z}) + \frac{2}{3}(p_{1x} - p_{0x}, p_{1y} - p_{0y}, p_{1z} - p_{0z}) \tag{6}$$

However, if p_0 and p_1 are not vertically aligned ($p_{0x} \neq p_{1x}$ or $p_{0y} \neq p_{1y}$), this causes our ending approach to not be vertically aligned either. This isn't optimal because we want our VTOL rocket to be pointed upright at the end of its path so that it can hover properly. To achieve this without having to perform a separate correction, we want to have the control point c_1 vertically aligned with p_1. To do this, we can instead set c_1 as

$$c_1 = (p_{1x}, p_{1y}, p_{0z}) + \tfrac{2}{3}(0, 0, p_{1z} - p_{0z})$$

This vertically aligns c_1 with p_1 while still maintaining even interpolation, which results in a vertical end approach that has minimal curvature. We now have all 4 points required to compute a path using the Bézier curve equation (eq. 2).

III. Path Following

The path is iteratively defined by calling the Bézier equation B(t). We divide the interval t = [0, 1] into 100[‡] equal points and call B(t) on each of these values of t, storing each point successively in a list. This defines our target path. Storing the path as points in this manner increases computational efficiency and allows for a more efficient path-following algorithm as this can be done pre-flight. Additionally, this approach allows us to access each point by accessing elements in an array, as opposed to calculating a new point mid-flight. Although there will be worst-case scenarios of recalculating a path mid-flight, this allows for an amortized constant runtime algorithm. Using this, we can iteratively check for the closest valid point to target efficiently. Unforeseen circumstances, like gusts of wind, can cause deviations from our intended path, making it inefficient or impossible to target the "next in line" point whenever we need a new point. If the angle θ (fig. 1), created from the current position to the next target point and the vertical axis below the target point, exceeds the maximum angular tilt, the next point along the path is prioritized as the initial point would not be physically reachable. We can quickly calculate θ by creating 2 vectors, v_1 from our rocket (r_x, r_y, r_z) to the point (p_x, p_y, p_z) and v_2 from the point going straight down, to get:

$$\begin{aligned} v_1 &= (p_x - r_x, p_y - r_y, p_z - r_z) \\ v_2 &= (0, 0, p_z) \\ \theta &= \frac{v_1 \cdot v_2}{|v_1| \cdot |v_2|} \end{aligned} \tag{7}$$

If there are no more valid points, then we must create a new path. This path is calculated using our maximum angle of tilt $\theta_{\max}$ to create a baseline minimum for the height of our new endpoint. For this new endpoint (fig. 2), we only increase the z position so that the new angle created, θ_n, is within the maximum angle constraint.

As the VTOL rocket hovers at the end of the ascension, the endpoint behavior should ideally be able to indicate when the rocket is approaching the end of the path. As we approach the end, we want to stop attempting to create a new path and instead focus on getting as close to the endpoint with a net 0 velocity (which corresponds to hovering). Therefore, we define a distance from our endpoint at which we begin calling a new end behavior algorithm. For this algorithm,

[†] p_1 was solved by inspection because the only critical point is $t = 0$, which gives p_1 0 influence. By inspection, we found that with $0 \leq t \leq 1$, $t = 1$ gave us the maximum coefficient value.

[‡] The number of points to create was mostly an arbitrary choice, as it must later be redefined, through empirical testing, to maximize path precision while minimizing computational strain

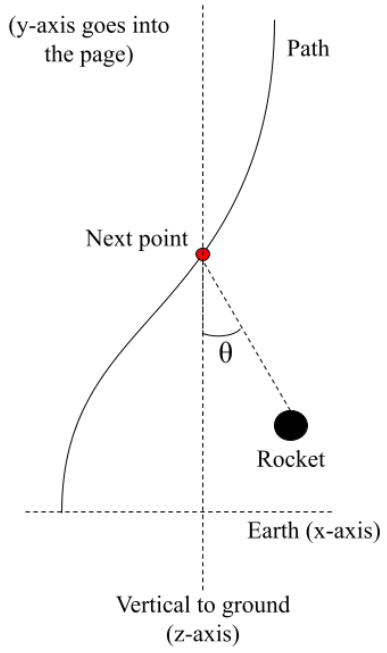

Fig. 1 Angle between the rocket and target point

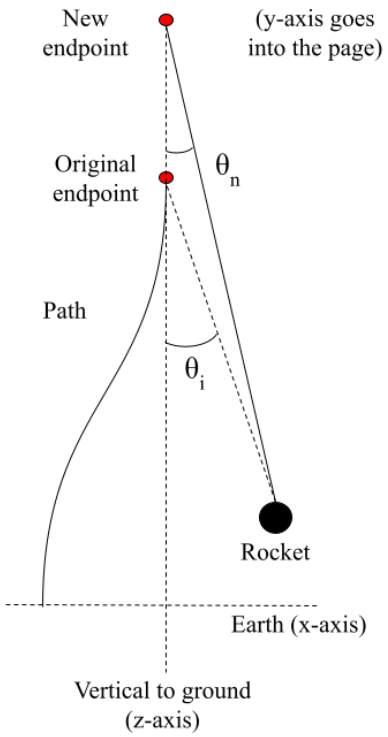

Fig. 2 Angle between the rocket and the original and new endpoints

we target only the endpoint, disregarding all other points. Additionally, now the angular constraint is disregarded (as the priority is hovering as close to the endpoint as possible), a new path is only created if the distance to the endpoint exceeds a predefined limit. Finally, once the target altitude is reached, the endpoint is also disregarded. This gives the largest area circle around the endpoint altitude to hover.

IV. Conclusion

A straight line path, despite being the shortest distance, between 2 points often fails to account for the physical limitations of real-world systems that require intricate path-planning. A path creation algorithm was created to optimize for the resource constraints of a vertical take-off and landing (VTOL) rocket, combining the control over endpoint behavior given by Hermite splines with the efficient computational requirements of Bézier splines. Then, a path-following algorithm was presented to take into consideration external forces that could throw our rocket off the desired path. In the case of our VTOL rocket, we must reach a specified minimum endpoint altitude, so our path-planning algorithm will primarily be used to update the path in cases where our original destination is no longer possible. Our path-following algorithm will be used broadly throughout the duration of our rocket's flight to maintain intermediate locations to target. This will ensure that our rocket follows an optimally smooth path, rather than a rigid one.

Acknowledgments

We would like to thank Anyi X. Lin (Guidance, Navigation, and Controls Team Lead) for his initial findings regarding our research and his continued mentorship throughout. We also would like to express our gratitude to Szu H. Chen (Project-Lead) and Propulsive Landers at Georgia Tech for providing us with the opportunity to conduct research as a team. Their support has been instrumental in furthering our research and development.

References

[1] Lau, B., Sprunk, C., and Burgard, W., "Kinodynamic motion planning for mobile robots using splines," *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2009, pp. 2427–2433. https://doi.org/10.1109/IROS.2009.5354805.