Efficient Approximation of Optimal High-Order Kinematic Trajectories

John G. Mooney*and Eric N. Johnson †

UAV Research Facility, Georgia Institute of Technology, Atlanta, GA, 30332

A method for efficiently planning one-dimensional pop-limited trajectories is presented, along with a direct method for synchronizing trajectories across multiple dimensions. This heuristic is designed for a double integrator utilizing acceleration commands passed through a 4th-order cascaded filter, the model for which is presented along with the system solution for an arbitrary time step and derivative limits. Examples for trajectories generated in both one and two dimensions are shown, with comparison to an iterative solver which searches for the exact optimal solution. The presented algorithm shows drastically lower computational requirements than the iterative solver, with very little cost in accuracy. Benefits and limitations of this approach are discussed.

I. Introduction

Unmanned Aerial Vehicles (UAVs) are rapidly becoming more important and relevant to military and civil aviation operations. However, there are a number of tasks which, thus far, still require the skill, adaptability, and experience of a human pilot. One task in particular that requires an especially skilled pilot is helicopter slung-load operations.

The addition of an external load changes the stability and handling qualities of the aircraft-load system, typically making it more difficult to fly. As a result, most pilots flying with slung-loads limit their speed and aggressiveness to keep the load and the aircraft well within the flight envelope.

Despite the difficulties of flying with a slung-load, aggressive maneuvers are possible. Christmas tree farms in the US Pacific Northwest have used helicopters with slings to load their trucks for transport for at least 30 years (Figure 1). The pilots have learned how to employ very aggressive maneuvers to pick up and drop the trees at very precise locations in a minimum of time. It appears the key to this capability is to anticipate the load's motion and plan inputs to drive the load to a precise position.

Figure 1. Helicopter loading Christmas trees for transport.

*Graduate Research Assistant, School of Aerospace Engineering, Georgia Institute of Technology, Atlanta, GA 30332, AIAA Student Member.

†Associate Professor, School of Aerospace Engineering, Georgia Institute of Technology, Atlanta, GA 30332, AIAA Member.

Motion planning has long been studied for a wide range of robotics applications, and could enable unmanned slung-load operations as well as augment human pilots in this difficult task. Recent work has been done to apply Rapidly Exploring Random Trees (RRTs) to this problem,[1] but some of the best extensions to RRT (for example, RRT*)[2] require the ability to efficiently find a trajectory between two arbitrary points in the state space, ideally an optimal one. For some simple systems, this "steer function" is not difficult to find; however for a complex system, such as a slung-load encumbered helicopter, this boundary value problem may be computationally intractable if a solution even exists.

I.A. Problem Statement

One approach to motion planning for slung loads is to generate a reference trajectory for the load to track. If the helicopter-load system is considered as a simple pendulum, the acceleration of the load is generated by tension and orientation of the line; and the tension on the line is determined (roughly speaking) by the aircraft's position and acceleration. This implies that the load cannot be subjected to instantaneous changes in acceleration. By same reasoning, jerk on the load is related to aircraft velocity relative to the load. Defining the 4th, 5th, and 6th derivatives of position as snap, crackle, and pop,[3] snap on the load is related to the aircraft's acceleration, crackle on the load to the aircraft's jerk, and pop on the load to the aircraft's snap. By limiting the pop of the load, we consequently ensure the snap of the aircraft is bounded, as well as jerk and acceleration. Further limits on the aircraft's motion derivatives are probably overkill.[3]

With these considerations in mind, the problem to be solved is to find a kinematic trajectory from one arbitrary position, velocity, and acceleration, to another arbitary position, velocity, and acceleration in minimum (or near-minimum) time with limits on the 1st through 6th time derivatives of position, in a way which is computationally efficient. For planning in more than one degree of freedom, we assume that it is sufficient to consider each degree of freedom separately, then synchronize the motion plans so that the shorter duration plans are extended to the length of the plan for the limiting degree of freedom.

This paper describes a proposed solution which uses a simplified model (a double-integrator with a fourth-order filter on acceleration) and a set of heuristics to find an approximation of the time-optimal trajectory.

I.B. Related Work

The minimum-time trajectory of a one-dimensional double integrator is a classic problem in optimal control. However, closed form solutions exist only for systems with acceleration and jerk constraints. Further constraints on state derivatives require numerical solutions.[3] Thompson showed a comparison between the numerically-solved minimum-time pop-limited trajectory, and an acceleration-limited trajectory, which was then passed through at 6th order cascaded "jerk filter" to impose bounds on the 3rd through 6th derivatives on position. This system works well for trajectories with stationary endpoints, such as the example telescope slewing problem described in the paper. However, it more difficult to apply to problems with non-stationary endpoints due to the challenge of correctly initializing the filter.

Tsourdos, White, and Shanmugavel describe how to create trajectories using Pythagorean hodographs. These paths are intended for constant-speed fixed-wing applications, having bounds on the derivatives of curvature.[4] Kumar used iterative methods on piecewise polynomial functions of time to create minimum-snap trajectories.[5] More recently, Bouef, et al.[6] developed a method to rapidly find snap-limited trajectories with non-zero velocity and acceleration at the endpoints. They broke the trajectory into phases of positive, negative, and zero snap, and assumed that the dynamics would be fast enough that the system would reach maximum jerk, acceleration, and velocity across the trajectory. This assumption means that the entire trajectory can be defined by the time at maximum velocity and therefore extended as necessary. The primary shortcoming of this method is that finding the optimum for shorter moves (where maximum velocity would not be reached) requires an iterative dichotomous search.

Finally, an Israeli research team has found a general method by which the optimum trajectory can be found with independent and asymmetric limits on each of the higher order derivatives.[7] This method uses a recursive iterative algorithm, and its solution very closely matches the true optimum. Ezair also extends the algorithm to enable synchronization of multiple degrees of freedom using the same basic method as a single degree of freedom.

II. General Approach

The approach described in this paper takes inspiration from Thompson,[3] but rather than passing a position command through a 6th-order cascaded filter, we pass an acceleration command through at 4th-order cascaded filter (Figure 2). However, it is not clear what the unfiltered acceleration command should be in order to reach the goal state in minimum time. The classic optimum bang-bang solution offers a clue—a switching curve can be drawn in phase-space, on one side of which the input should be positive, and the other negative. These curves are represent the maximum acceleration trajectories that intersect the desired final position and velocity. The system trajectory follows these curves down to the desired state.

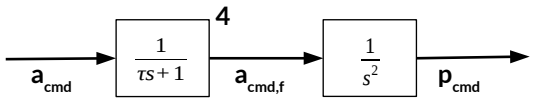
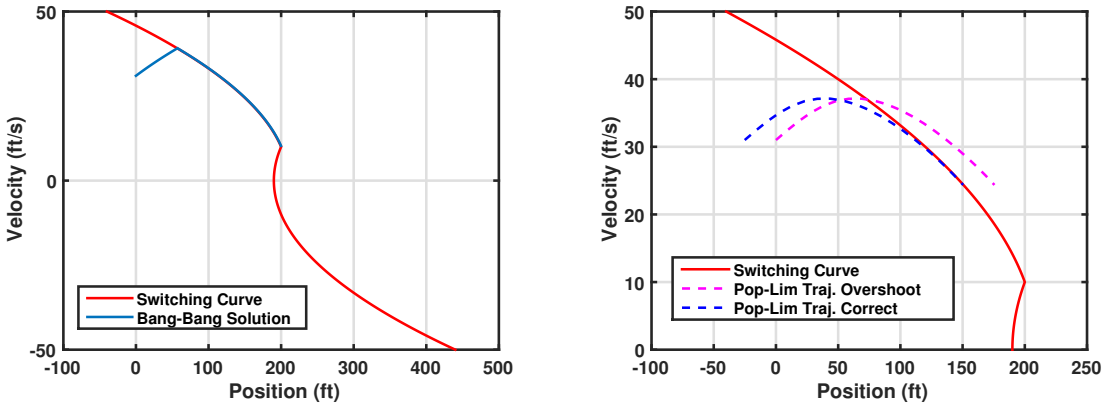


Figure 2. Block diagram for the pop-limited double integrator.

The cusp in the resultant trajectory is a point of instantaneous change of acceleration. However, with limits on jerk and the other sub-derivatives of acceleration, the trajectory in phase space must be smoothly curving. In Figure 3(a), the bang-bang trajectory is shown along with the associated switching curves. Figure 3(b) shows in magenta the resultant trajectory with the same initial conditions as 3(a), but with the maximum deceleration limited by the pop constraint. Clearly, the pop-limited system cannot (in this case) reach the switching curves; but if the point where the system crosses the switching curve can be anticipated, the unfiltered command can be selected so that the pop-limited trajectory settles on the switching curve and thus can follow the curve to the goal state (Figure 3(b) in blue).



(a) (b)

Figure 3. Comparison in the phase plane of ordinary optimal double integrator trajectory with pop-limited trajectory. The trajectories on the right demonstrate the necessity of projecting the system forward in time to predict crossings of a switching surface.

Two insights can be gained from this observation. First, the optimal unfiltered acceleration command to the system will take the general form shown in Figure 4—that is, the unfiltered command will be of bang-bang form with six phases, some of which will be of zero duration (depending on which side of the switching curves the initial state of the system lies).

Second, there needs to be some efficient way of predicting the future state of the system to anticipate crossings of the switching curve. Fortunately, since this is a linear system, we have an analytical solution available.

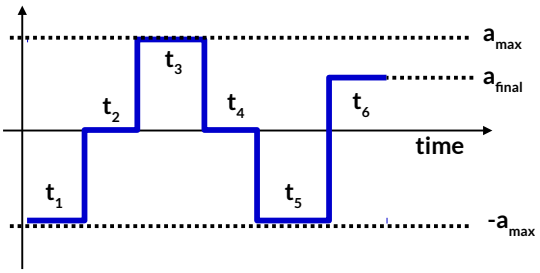


Figure 4. General form of the optimal unfiltered acceleration command.

II.A. Model

This system is a double integrator fed by a filtered acceleration command:

$$\frac{d}{dt}\begin{Bmatrix} p \\ v \\ a_{f4} \\ a_{f3} \\ a_{f2} \\ a_{f1} \end{Bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & -\frac{1}{\tau} & \frac{1}{\tau} & 0 & 0 \\ 0 & 0 & 0 & -\frac{1}{\tau} & \frac{1}{\tau} & 0 \\ 0 & 0 & 0 & 0 & -\frac{1}{\tau} & \frac{1}{\tau} \\ 0 & 0 & 0 & 0 & 0 & -\frac{1}{\tau} \end{bmatrix} \begin{Bmatrix} p \\ v \\ a_{f4} \\ a_{f3} \\ a_{f2} \\ a_{f1} \end{Bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \frac{1}{\tau} \end{bmatrix} a_c \tag{1}$$

The analytical solution for this system over a time interval T with a constant a_c is computed relatively easily:

$$x_T = e^{AT}x_0 + \left(\int_0^T e^{As}Bds\right) a_c \tag{2}$$

where A is the system matrix and B is the control matrix from (1). Define

$$\lambda = e^{-\frac{T}{\tau}} \tag{3}$$

and the discrete system matrix is then

$$e^{AT} = A_d(T,\tau) = \begin{bmatrix} 1 & T & A_{1,3} & A_{1,4} & A_{1,5} & A_{1,6} \\ 0 & 1 & \tau - \tau\lambda & A_{2,4} & A_{2,5} & A_{2,6} \\ 0 & 0 & \lambda & \frac{T\lambda}{\tau} & \frac{T^2\lambda}{2\tau^2} & \frac{T^3\lambda}{6\tau^3} \\ 0 & 0 & 0 & \lambda & \frac{T\lambda}{\tau} & \frac{T^2\lambda}{2\tau^2} \\ 0 & 0 & 0 & 0 & \lambda & \frac{T\lambda}{\tau} \\ 0 & 0 & 0 & 0 & 0 & \lambda \end{bmatrix} \tag{4}$$

$$A_{1,3} = \tau^2\lambda + T\tau - \tau^2 \tag{5}$$

$$A_{1,4} = 2\tau^2\lambda + T\tau - 2\tau^2 + T\tau\lambda \tag{6}$$

$$A_{1,5} = \frac{T^2\lambda}{2} + 3\tau^2\lambda + T\tau - 3\tau^2 + 2T\tau\lambda \tag{7}$$

$$A_{1,6} = \frac{T^3\lambda + 24\tau^3\lambda + 6T\tau^2 - 24\tau^3 + 18T\tau^2\lambda + 6T^2\tau\lambda}{6\tau} \tag{8}$$

$$A_{2,4} = \tau - T\lambda - \tau\lambda \tag{9}$$

$$A_{2,5} = -\frac{T^2\lambda + 2\tau^2\lambda - 2\tau^2 + 2T\tau\lambda}{2\tau} \tag{10}$$

$$A_{2,6} = -\frac{T^3\lambda + 6\tau^3\lambda - 6\tau^3 + 6T\tau^2\lambda + 3T^2\tau\lambda}{6\tau^2} \tag{11}$$

$$\left(\int_0^T e^{As}Bds\right) = B_d(T,\tau) = \begin{bmatrix} T^2/2 - 4T\tau - 10\tau^2\lambda + 10\tau^2 - (3T^2\lambda)/2 - 6T\tau\lambda - (T^3\lambda)/(6\tau) \\ T - 4\tau + 3T\lambda + 4\tau\lambda + (T^2\lambda)/\tau + (T^3\lambda)/(6\tau^2) \\ 1 - (\lambda(T^3 + 3T^2\tau + 6T\tau^2 + 6\tau^3))/(6\tau^3) \\ 1 - (\lambda(T^2 + 2T\tau + 2\tau^2))/(2\tau^2) \\ 1 - (\lambda(T+\tau))/\tau \\ 1 - \lambda \end{bmatrix} \tag{12}$$

The time constant of the acceleration filter is $\tau_{af} = \sqrt{\tau^2 + \tau^2 + \tau^2 + \tau^2}$, so the 99.5% rise time (t_r) is about 10.5τ. In practice, $t_r = 11\tau_{af}$ produced qualitatively better results.

Further, the maximum amount of jerk, snap, crackle, and pop produced by these trajectories is directly related to both the maximum acceleration and the time constant used in the filter. In equation (1), the expression for the derivative of the "real" acceleration command after being filtered is

$$\dot{a}_{f4} = J = -\frac{1}{\tau}a_{f4} + \frac{1}{\tau}a_{f3} \tag{13}$$

which is also equivalent to the system jerk. The successive derivatives of this expression give the snap, crackle, and pop as well:

$$\ddot{a}_{f4} = S = \frac{1}{\tau^2}a_{f4} - \frac{2}{\tau^2}a_{f3} + \frac{1}{\tau^2}a_{f2} \tag{14}$$

$$\dddot{a}_{f4} = C = -\frac{1}{\tau^3}a_{f4} + \frac{3}{\tau^3}a_{f3} - \frac{3}{\tau^3}a_{f2} + \frac{1}{\tau^3}a_{f1} \tag{15}$$

$$\ddddot{a}_{f4} = P = \frac{1}{\tau^4}a_{f4} - \frac{4}{\tau^4}a_{f3} + \frac{6}{\tau^4}a_{f2} - \frac{4}{\tau^4}a_{f1} + \frac{1}{\tau^4}a_c \tag{16}$$

Since the magnitude of the filtered value of acceleration can never exceed the magnitude of the unfiltered acceleration command, this means that jerk is bounded by $2\frac{a_{max}}{\tau}$. Likewise, snap is bounded by $4\frac{a_{max}}{\tau^2}$, crackle by $8\frac{a_{max}}{\tau^3}$, and pop by $16\frac{a_{max}}{\tau^4}$. These bounds are, of course, worst-case and represent situations where, for example $a_{f4} = a_{f2} = a_c = a_{max}$ and $a_{f3} = a_{f1} = -a_{max}$. However, this should never happen in practice since the filtered accelerations will all be initialized at the same value, and should not differ by that much. In fact, the practical maxima were estimated empirically:

$$J_{max} = 0.2239\frac{a_{max}}{\tau} \tag{17}$$

$$S_{max} = 0.13\frac{a_{max}}{\tau^2} \tag{18}$$

$$C_{max} = 0.16\frac{a_{max}}{\tau^3} \tag{19}$$

$$P_{max} = 0.23\frac{a_{max}}{\tau^4} \tag{20}$$

An important feature of this method is that additional derivatives (for example, the first and second derivatives of pop) can be added for very little cost—merely add an additional first order filter to the cascade for every additional subderivative. The discrete system matrix and control matrix can easily be found using any number of off-the-shelf symbolic solvers.

II.B. Heuristic

With the model specified above, a heuristic is defined which behaves similarly to a switching surface. When instantaneous changes in acceleration are allowed, and the goal state is a stationary point, the switching surface viewed in the position-velocity phase plane is composed piecewise of a pair of parabolas representing positive and negative maximum acceleration, and horizontal lines representing positive and negative maximum velocity. Below/to the left of the curve the command is $a_c = +a_{max}$, or $a_c = 0$ when at maximum velocity, and the opposite command on the other side of the curve. This logic produces the solution to the classic minimum-time problem.[3] If the goal state is not stationary, the problem changes very little—the switching curves incorporate more of one parabola and less of the other (Figure 3), and the optimal solution is still a bang-zero-bang control.

If the problem instead has limits on derivatives of acceleration (in this case, down to pop), and the final state has non-zero acceleration, a similar switching surface would also have to include segments representing the transition from $a_c = \pm a_{max}$ to $a_c = a_{final}$, and the maximum acceleration curves are offset from the abscissa. There are also segments representing the transition from $a_c = 0$, $v_c = \pm v_{max}$ to $a_c = \pm a_{max}$ (see Figure 5). Finally, several other curves of maximum acceleration are found to assist in estimating the time (or location in phase plane) to transition from one command to another.

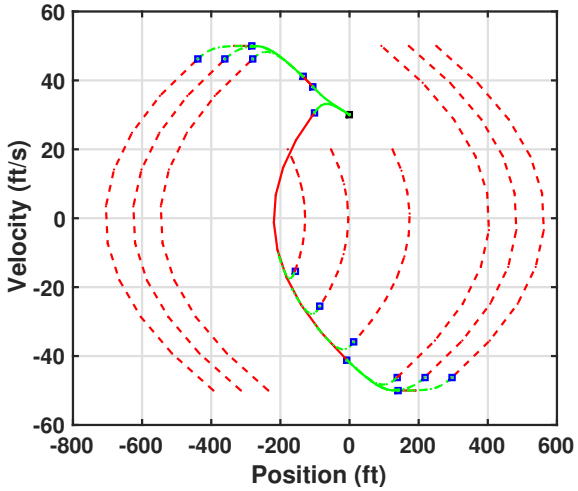


Figure 5. Switching curve and transition points in the phase plane for an example case. Curves of maximum acceleration are shown in red, with the primary switching curve in solid, supplementary curves dashed. Pop-limited transitions from one curve of acceleration to another are shown in green. Transition points are marked in blue.

As in the classic bang-zero-bang solution, the optimal solution to this trajectory will consist of a segment to get the system on to the switching surface as quickly as possible, then following the switching surface to the desired state. The general approach to this is to propagate the system forward in time by t_r with $a_c = \pm a_{max}$, positive if left of the curve, negative if right of curve. If the propagated state is still on the same side of the curve, then a_c is maintained until the pre-solved transition time, and the remainder of the commands are found based on the command profile of the switching curve. If the propagated state crosses the switching curve, then a binary search is conducted on the switching time until the propagated state lies within a specified distance of the switching curve, and again the remainder of the commands can be found from the switching curve. If the initial state is to the left of the switching curve and is near the "final approach funnel" then it may not be possible to get on the switching curve, but a series of binary searches can find a trajectory solution which attempts to minimize the trajectory time. The logic is written assuming that the final velocity with be zero or positive without loss of generality, as the signs on the initial and final states can be changed, then the signs of the resultant trajectory changed back.

II.C. Synchronization

Since the above methodology can only apply to a single degree of at a time, with a non-fixed final time, independent planning of two or three (or more) degrees of freedom will result in different final times. The

shorter duration degrees of freedom will obviously require some way to extend their duration to match the end time of the longest one. Fortunately, this is a straightforward problem to solve considering that an analytical solution for the system is available. Consider the discrete system given in Equations (2), (4), and (12). This system takes the form of a difference equation:

$$x_{k+1} = A_d x_k + B_d u_k \tag{21}$$

where

$$x_k = x(t_0 + kT) \tag{22}$$
$$u_k = a_c(t_0 + kT) \tag{23}$$
$$A_d = A_d(T, \tau) \tag{24}$$
$$B_d = B_d(T, \tau) \tag{25}$$

Using recursion, we can find an expression for the state at any time interval:

$$x_{k+1} = A_d x_k + B_d u_k \tag{26}$$
$$x_{k+2} = A_d x_{k+1} + B_d u_{k+1} \tag{27}$$
$$= A_d(A_d x_k + B_d u_k) + B_d u_{k+1} \tag{28}$$
$$= A_d^2 x_k + A_d B_d u_k + B_d u_{k+1} \tag{29}$$
$$\cdots \tag{30}$$
$$x_{k+N} = A_d^N x_k + \left[A_d^{N-1} B_d, A_d^{N-2} B_d, \cdots, A_d B_d, B_d\right] U \tag{31}$$
$$= A_d^N x_k + \Theta U \tag{32}$$

where

$$U = [u_k, u_{k+1}, \cdots, u_{k+N-1}]^T \tag{33}$$

Since the final time for the system is known, Equation (32) can be solved for a sequence of control inputs, U, assuming that $N \geq 6$ (or whatever the system order). It can be shown that the matrix Θ has full row rank. The minimum norm solution takes the form

$$U = \Theta^T(\Theta\Theta^T)^{-1}(x_{k+N} - A_d^N x_k) \tag{34}$$

This leaves only the question of the ideal selection for N, and consequently for the discrete time step T. By inspection, it is clear that the matrix Θ contains many instances of $e^{-T/\tau}$. Numerical experimentation has revealed that the condition number of Θ is very dependent on the ratio between T and τ, with the best conditioned matrices occurring when the two are the same order of magnitude, and $T < \tau$. Since $T = t_f/N$, it is clear that N must be selected to get a desirable time step T. Fortunately, though large N increases the computational cost somewhat, the matrix inverse required in (34) will always be 6-by-6.

A concern with using Equation (34) is that U is unconstrained. That said, this solution represents about the same control effort as the optimal one, but spread out over a larger time period; and (34) represents the minimum-norm solution. Further analysis may reveal bounds on the individual elements of the solution vector, but again numerical experimentation indicates the acceleration constraint (and by extension, constraints on the subderivatives) is normally satisfied.

An example case is shown in Figure 6, where an optimal trajectory is stretched by 50%.

III. Results

III.A. One Dimensional Trajectories

Two example one-dimensional plans are presented (Figures 7 and 8). For comparison purposes, these results are presented with the solution provided by the algorithm described by Ezair, et al.[7] These plans were

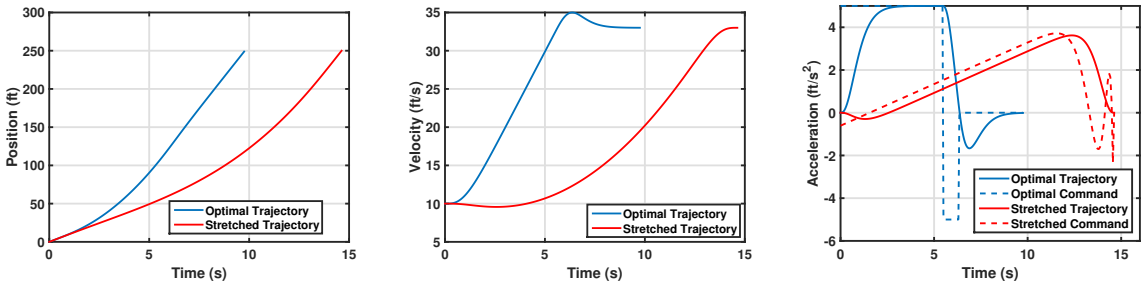


Figure 6. Example position, velocity, and acceleration of a trajectory solution from the heuristic, and one "stretched" by another 50%.

computed sequentially on the same machine using Matlab scripts. In addition, each comparison was run twice successively to take advantage of Matlab's Just-In-Time (JIT) accelerator for both methods.

Both example cases show clear similarity between the true optimum solved by Ezair, and by the approximation described here—in most cases tested, the approximated trajectories are 1% to 5% longer in duration than the optimum. The inaccuracy of the heuristic in finding the true optimum is offset by the savings in computation: with JIT-accelerated code, the presented method is a minimum of 100 times faster, and commonly 130 to 150 times faster.

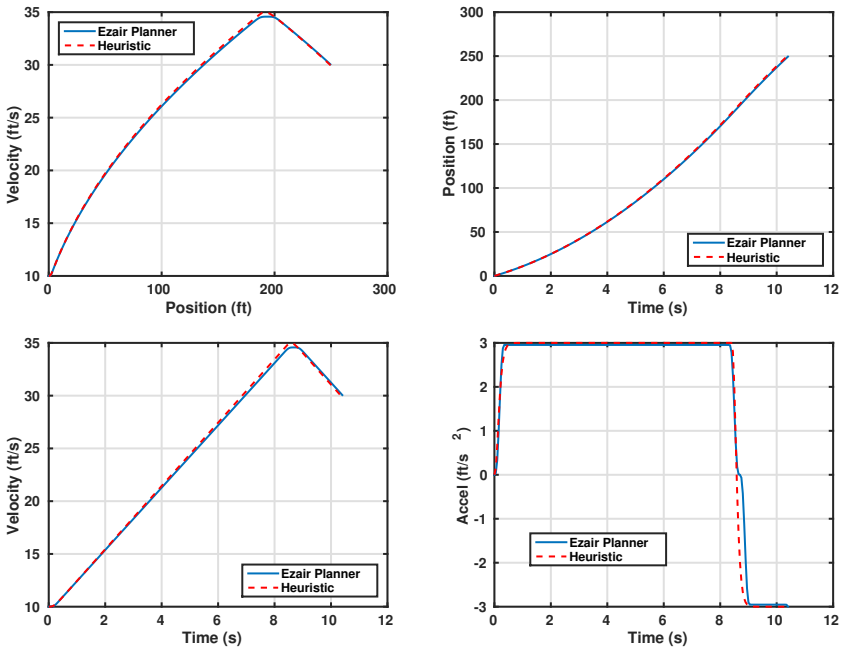


Figure 7. Example one dimensional trajectory, with the corresponding solution using the Ezair method.[7] Phase plot is in the upper left, position history upper right, velocity history lower left, and acceleration history lower right.

III.B. Two Dimensional Trajectories

An example scenario is shown using the presented algorithm (Figures 9 through 12), and again Ezair[7] is used as a comparison in each of the individual degrees of freedom. Like the one-dimensional cases, the presented planner finds trajectories similar to the optimum, at least in the limiting degree of freedom (for this example, the 2nd degree of freedom). In the degree of freedom requiring synchronization, the presented planner finds trajectories that are notably different, especially at the level of acceleration. However, these differences still meet the desired constraints, and result in a trajectory which is once again only 1% to 5% longer in duration than the optimal.

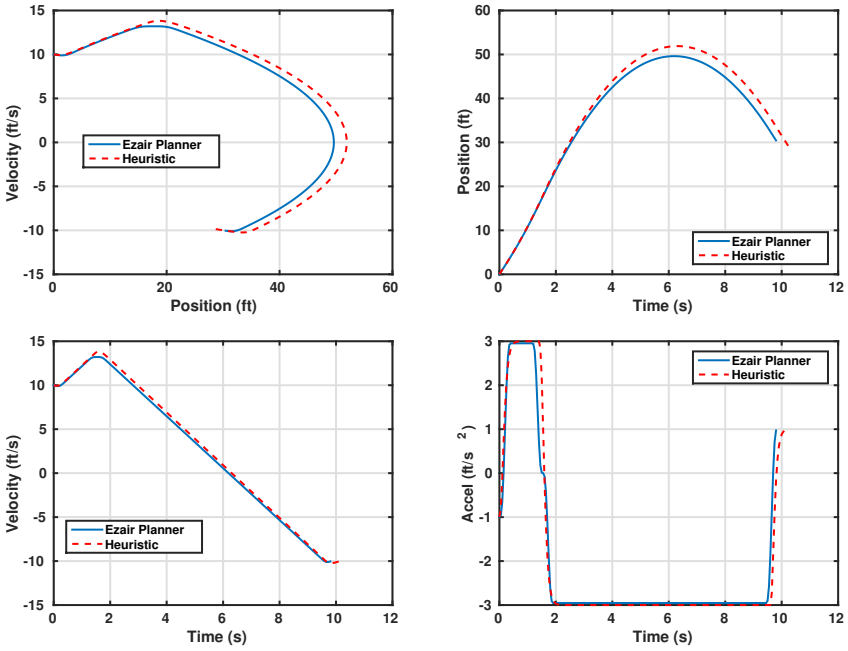


Figure 8. Example one dimensional trajectory, with the corresponding solution using the Ezair method.[7] Phase plot is in the upper left, position history upper right, velocity history lower left, and acceleration history lower right.

III.C. Discussion

Development and testing of this method has revealed several of its strengths and weaknesses. First, and perhaps most importantly, this method is computationally light and theoretically could be used as a bang-zero-bang control block in real-time rather than as a planner. Second, while this does not produce the true minimum-time pop-limited solution, it does appear to approximate it well. The linear model used in this approach means that it can easily be adapted to a crackle-, snap-, or jerk-limited system, or possibly limits on derivatives higher than pop. Since the algorithm uses a switching surface, and the model implicitly incorporates high-order derivative constraints, the limits on derivatives higher than pop could be added with very little additional computation.

The capability for higher dimensional planning is another strength of this method. The "stretching" or synchronization phase of this method is direct and non-iterative, where each added dimension after the 2nd requires one computation of the approximate optimum trajectory, and one computation of the synchronization routine. Therefore, computation scales linearly with dimension—adding a 100th dimension

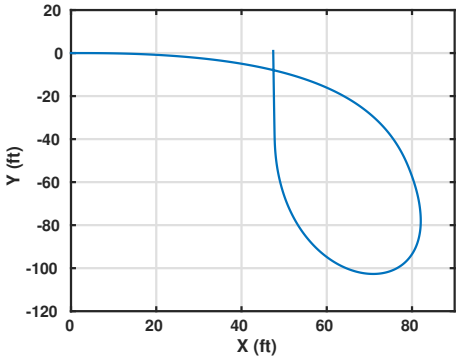


Figure 9. Example 2D trajectory. The particle begins at (0,0) moving in the X direction, and ends at (50,0) moving in the Y direction.

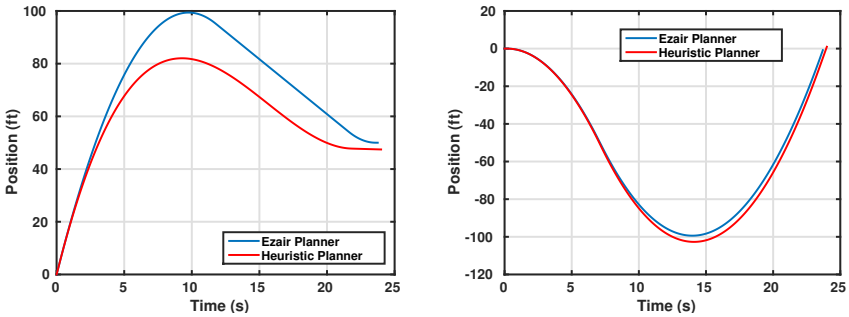


Figure 10. Example 2D trajectory position history. The heuristic result matches the optimal result in the second degree of freedom, but synchronized degree of freedom shows a different result.

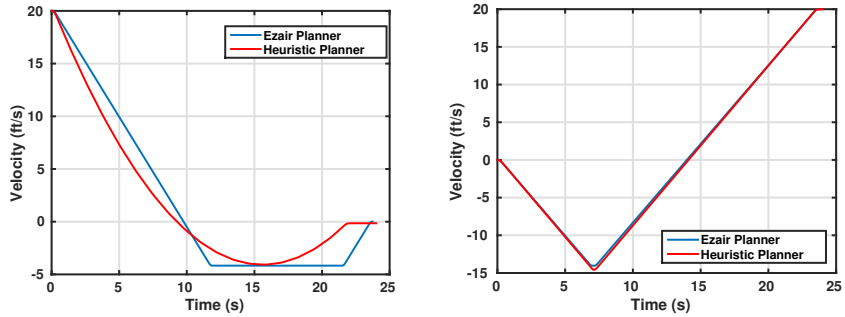


Figure 11. Example 2D trajectory velocity history. The heuristic result matches the optimal result in the second degree of freedom, but synchronized degree of freedom shows a different result.

has the same marginal computing cost as adding the 3rd. This may have applications in attempting to synchronize, for example, the arrival times of a swarm of aircraft.

However, the method is not without limitations. In particular, since each degree of freedom is planned and limited separately, the overall control authority available in the system may not be fully used in systems where the constraint is on total control effort. This problem is more pronounced with each additional dimension planned. A possible solution would be to estimate the required control authority in each direction before solving for the individual trajectories, and set limits accordingly. Another method may be to iteratively solve the problem giving more authority to one degree of freedom and less to the others until the trajectory time in each degree of freedom is the same.

As formulated, the planner does not incorporate each derivative limit individually, but picks a single time constant τ which ensures all limits are met. Trying to incorporate each limit individually is possible, though—one would need to modify (1) to have a different time constant for each filter, and re-solve the discrete system matrices. These limits could be found by rewriting (13) through (16), and solving for the required constants based on each limit. Some experimentation may be required to find the practical limits on individual τ's in order to avoid overly conservative results.

Similarly, this solver does not currently consider asymmetric acceleration limits (as one may need, for example, to plan the descent and landing of a reusable rocket.) Again, this would be theoretically possible, as the logic depends only upon detecting where the system would cross the switching surface—different curves could easily be incorporated for positive vs. negative acceleration.

Finally, as mentioned in section II.C, the condition number of the Θ matrix is heavily dependent upon the ratio T/τ. If a system has very high jerk, snap, etc. limits, the τ used in (1) will consequently be very small. To maintain a well-conditioned Θ matrix in this case would require a small T and accordingly a large number (N) of elements in the control sequence U. This would hurt the computational efficiency of the synchronization routine by requiring multiplication of several large matrices. At a certain point, then, it may be worthwhile to ignore a very high snap/crackle/pop limit and model the jerk/snap/crackle as changing instantaneously.

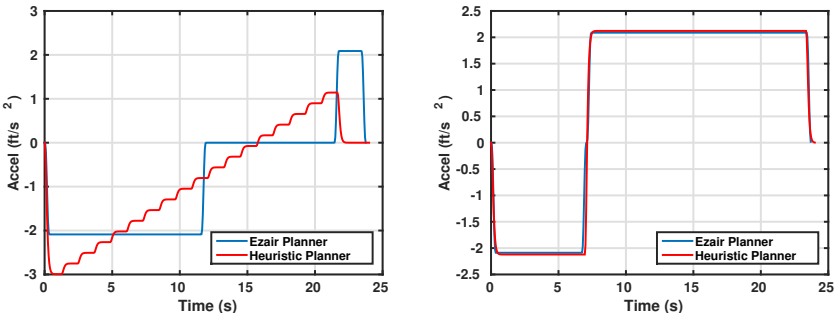


Figure 12. Example 2D trajectory acceleration history. The heuristic result matches the optimal result in the second degree of freedom. In the synchronized degree of freedom, the Ezair planner still uses a variation of bang-zero-bang, but the minimum norm solution presented here uses a more gradual change.

Though the motivating application for this method is planning reference trajectories for slung-loads, this is a useful result for any number of motion planning problems where the both the smoothness of the trajectory and the optimality of the solution are important.

IV. Conclusion

This paper presented an algorithm for approximating optimal one-dimensional pop-limited trajectories, along with a direct technique for synchronizing multiple degrees of freedom. The trajectories provided by this algorithm are an estimated 1% to 5% longer in duration than the optimum, but require $1/100^{th}$ or less of the computation required for a high-accuracy iterative solver searching for the true optimum. The solution described here could be helpful for use as an incremental or local planner for another, higher-level trajectory planner (e.g. RRT*), or for use in real-time applications.

Acknowledgments

This study is funded by the U. S. Army under the Vertical Lift Research Center of Excellence (VLRCOE) program managed by the National Rotorcraft Technology Center, Aviation and Missile Research, Development and Engineering Center under Cooperative Agreement W911 W61120010 between the Georgia Institute of Technology and the U. S. Army Aviation Applied Technology Directorate. The authors would like to acknowledge that this research and development was accomplished with the support and guidance of the NRTC. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Aviation and Missile Research, Development and Engineering Center or the U.S. Government.

References

[1]Johnson, E. N. and Mooney, J. G., "Longitudinal Motion Planning for Slung-Loads Using Simplified Models and Rapidly-Exploring Random Trees," *Sixth AHS International Specialists' Meeting On Unmanned Rotorcraft Systems*, Chandler, Arizona, 2015.

[2]Karaman, S. and Frazzoli, E., "Optimal kinodynamic motion planning using incremental sampling-based methods," *Proceedings of the IEEE Conference on Decision and Control*, 2010.

[3]Thompson, P. M., "Snap, Crackle, and Pop," *AIAA Southern California Aerospace Technology and Systems Conference*, Santa Ana, CA, 2011.

[4]Tsourdos, A., White, B., and Shanmugavel, M., *Cooperative Path Planning of Unmanned Aerial Vehicles*, Aerospace Series, Wiley, 2010.

[5]Mellinger, D. and Kumar, V., "Minimum Snap Trajectory Generation and Control for Quadrotors," *International Conference on Robotics and Automation*, Shanghai, China, 2011, pp. 2520–2525.

[6]Boeuf, A., Cortes, J., Alami, R., and Simeon, T., "Planning agile motions for quadrotors in constrained environments," *International Conference on Intelligent Robots and Systems*, Chicago, Illinois, 2014.

[7]Ezair, B., Tassa, T., and Shiller, Z., "Planning high order trajectories with general initial and final conditions and asymmetric bounds," *The International Journal of Robotics Research*, Vol. 33, No. 6, 2014, pp. 898–916.