Model Predictive Control for Quadrotor Tracking of Moving Landing Platform

Will Cohen

I. Motivation

In the past, package delivery has relied on hub-to-hub air deliveries with package delivery performed by delivery vans and human delivery agents. With the advent of better battery technology, quadrotor unmanned aerial vehicles (UAVs), and modern control, it is now feasible to remove the human from the delivery loop. This new delivery paradigm is driven by developments in model predictive control (MPC), wherein a drone is capable of tracking a delivery vehicle for return, landing, and charging. Whereas the basic tracking can be performed with traditional PID controllers, obstacle avoidance, and the landing process are enabled through MPC. The latter will be the focus of this paper.

II. Problem Statement

This section will introduce a miniaturized system so that future tests on the feasibility of this controller can be performed in a laboratory environment. The drone chosen is the Crazyflie 2.0, a micro aerial vehicle (MAV) with well researched system specifications [2]. The landing platform is assumed to be a 10 cm wide and 20cm long flat rectangle, with a GPS beacon at its center.

A. Crazyflie System Specifications

The Crazyflie 2.0 moments of inertia, thrust characteristics, and rotor torques of the MAV are derived in the paper by Förster [2], which are implemented in the controller later. Cross terms are excluded in the moment of inertia to simplify the physics. These characteristics can be found below.

TABLE I: Crazyflie 2.0 Characteristics

Variable	Value	Unit
I_{xx}	$16.82 * 10^{-6}$	kg m^2
I_{yy}	$16.89 * 10^{-6}$	kg m^2
I_{zz}	$29.81 * 10^{-6}$	kg m^2
k_1	0.0915	
k_2	0.0677	
k_3	$5.49 * 10^{-4}$	
q_1	$5.96 * 10^{-3}$	
q_2	$1.56 * 10^{-5}$	

The multipliers for the force, k_i, and the torques, q_i, are unitless multipliers used to convert the signal input into a force value. This signal value, x, has been normalized for inputs $x \in [0, 1]$ These equations are shown below.

$$f_i(x) = k_1 x^2 + k_2 x + k_3 \tag{1}$$

$$\tau_i(x) = q_1 x + q_2 \tag{2}$$

Each of these values in generated in the body frame of the quadrotor for an individual motor, $i \in \{1, 2, 3, 4\}$.

B. Physics

For the physics of the quadrotor system, a simplified 3D model with 12 states is used:

$$\Big[x, y, z, \dot{x}, \dot{y}, \dot{z}, \phi, \theta, \psi, \dot{\phi}, \dot{\theta}, \dot{\psi}\Big] \tag{3}$$

An image of this configuration can be found below:

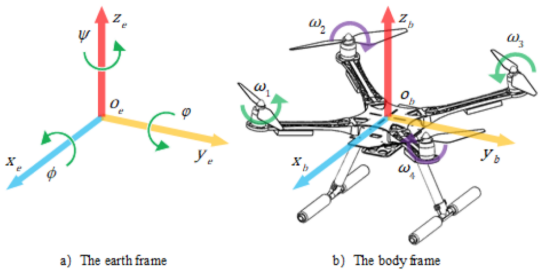


Fig. 1: Quadrotor Coordinate Configuration [3]

These states are further referenced as the following aggregations: $p = [x, y, z]$, $\eta = [\phi, \theta, \psi]$, $\dot{p} = [\dot{x}, \dot{y}, \dot{z}]$, $\dot{\eta} = \left[\dot{\phi}, \dot{\theta}, \dot{\psi}\right]$.

The following assumptions are made regarding the physical model [6]:

1) the body frame of the quadcopter is rigid;
2) the center of the body frame coincides with the center of gravity;
3) the aerodynamics effects are neglected.

The nonlinear state transition equations are generated following these assumptions. The transition for p and η will be dictated by the below equations:

$$\frac{\partial p}{\partial t} = \dot{p}, \ \frac{\partial \eta}{\partial t} = \dot{\eta} \tag{4}$$

In the above equations $\dot{p}$ and $\dot{\eta}$ are the linear and angular velocities from the state in Equation 3.

In order to calculate the forces acting on the quadrotor in the inertial frame, the forces must be rotated out of the body frame. This is done with a 3-2-1 Euler rotation with ϕ, θ, and ψ. This rotation matrix can be found below, where c is cosine and s is sine.

$$\mathbf{R} = \begin{bmatrix} c\theta c\psi & c\psi s\theta s\phi - s\phi c\psi & c\psi s\theta c\phi + s\psi s\phi \\ c\theta s\psi & s\psi s\theta s\phi + c\psi c\phi & s\psi s\theta c\phi - c\psi s\phi \\ -s\theta & c\theta s\phi & c\theta c\phi \end{bmatrix} \tag{5}$$

This rotation is applied to the body frame forces to rotate them into the inertial frame, as shown below:

$$m\ddot{p} = \mathbf{R} \begin{bmatrix} 0 \\ 0 \\ \text{sum}(f_i) \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix} \tag{6}$$

In the above equation, f_i is the force generated by each of the 4 rotors, and m is the mass of the quadrotor. This force can be divided by mass on both sides and will result in the corresponding accelerations for the state transition matrix. These can be seen below.

$$\begin{aligned} \ddot{x} &= (\cos(\psi)\sin(\theta)\cos(\phi) + \sin(\psi)\sin(\phi)) \sum_{i=1}^{4}(f_i) \\ \ddot{y} &= (\sin(\psi)\sin(\theta)\cos(\phi) - \cos(\psi)\sin(\phi)) \sum_{i=1}^{4}(f_i) \\ \ddot{z} &= \cos(\theta)\cos(\phi) \sum_{i=1}^{4}(f_i) - g \end{aligned} \tag{7}$$

The rotational accelerations are derived from the Newton-Euler method [1]:

$$I\ddot{\eta} + \dot{\eta} \times I\dot{\eta} = \begin{bmatrix} \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{bmatrix} \tag{8}$$

In the above, the three torque quantities are defined as below, with the motor number corresponding to Figure 1.

$$\begin{bmatrix} \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{bmatrix} = \begin{bmatrix} r((f_4 + f_1) - (f_2 - f_3)) \\ r((f_4 + f_3) - (f_1 - f_2)) \\ \sum_{i=1}^{4}(-1)^i \tau_i \end{bmatrix} \tag{9}$$

Equation 8 is solved for $\ddot{\eta}$ to resolve the angular accelerations shown below.

$$\begin{aligned} \ddot{\phi} &= I_{xx}^{-1}(\tau_\phi + (I_{zz} - I_{yy})\dot{\phi}) \\ \ddot{\theta} &= I_{yy}^{-1}(\tau_\theta + (I_{xx} - I_{zz})\dot{\theta}) \\ \ddot{\psi} &= I_{zz}^{-1}(\tau_\psi + (I_{yy} - I_{xx})\dot{\psi}) \end{aligned} \tag{10}$$

This produces the full system of equations, which can be seen in block form in Appendix B.

III. System Characteristics

A. Linearization

In order to make the problem more tractable, the highly non-linear physics from the prior section are linearized. MATLAB was used symbolically compute the Jacobian of the continuous time nonlinear

state transition equation to arrive at the continuous time linear state and control matrices.

$$\frac{\partial \vec{f}}{\partial \vec{x}} = A_c, \ \frac{\partial \vec{f}}{\partial \vec{u}} = B_c$$

This linearization occurs at the hovering equilibrium and makes use of a small angle approximation for η. The equilibrium state chosen for the linearization is at $[p, \dot{p}, \eta, \dot{\eta}] = \mathbf{0}$. The hovering input command is determined through a solution of the linear system of equations that results from the aforementioned equilibrium, and is chosen to be $u_{\text{lin}} = [0.5837, 0.5532, 0.5837, 0.5532]$.

B. Discretization

In order to implement the controller, the state transition matrix is manipulated from the form $\dot{x} = A_c x + B_c u$ to the discrete-time format $x_{k+1} = A_d x_k + B_d u_k$ for some sampling time T_s. If A_c is invertible, this can be done numerically with the below method.

$$A_d = e^{A_c T s}, \ B_d = A_c^{-1} \left(e^{A_c T s} - I \right)$$

However, A_c is not invertible and other methods must be used. Instead, MATLAB's `c2d.m` was used to convert the continuous time system into discrete time with a sampling time of 0.01 seconds. This produces A_d and B_d matrices shown in appendix A, which will be used in the simulations.

C. Stability

The stability of the linear time invariant (LTI) system is inspected by examining the spectral radius of the A_d matrix created in the prior section. On this analysis provides the below distinct eigenvalues:

$$\text{spec}(A_d) \ = \ 1$$

For a stable discrete time linear system, $\text{spec}(A_d) < 1$, so this system is unstable [4].

D. Controllability

To check the controllability of the system, the controllability rank condition is used from the course notes [4]. These conditions state that the matrix pair (A_d, B_d) is controllable if they satisfy the below condition, where n is the number of states:

$$\text{rank} \begin{bmatrix} B & AB & \cdots & A^{n-1}B \end{bmatrix} = n$$

For the control matrices found in section III-B, the controllability matrix is full rank after 3 steps and that this system is controllable. Since the system is unstable but controllable, a controller is implemented that will drive this quadrotor towards the reference point, the vehicle landing zone.

IV. Controller

The model predictive controller designed for this projects makes use of the linearized system for tracking the impact of changes on the system as a whole. In addition to the primary 12 states, 6 error states and 6 measurement states are introduced that are used for the error calculation. Finally, four states are added for measuring the current motor command. The output command is converted to a change in the commanded state. The controller state can be seen below.

$$\left[\Delta p, \Delta \dot{p}, \Delta \eta, \Delta \dot{\eta}, e_p, e_{\dot{p}}, p, \dot{p}, u\right], \ \Delta u$$

This state is equivalent to the reference tracking design 4 discussed in the course notes [5], and implements a rate-based model into the controller. In this formulation an individual state, x_k, is calculated as shown in the model below.

$$\Delta x_k = x_k - x_{k-1}$$

Additionally, for a single time step, the controller designed in section 3 remains valid. This can be seen below.

$$\begin{aligned} \Delta x_{k+1} &= A x_k + B u_k - A x_{k-1} - B u_{k-1} \\ &= A \Delta x_k + B \Delta u_k \end{aligned}$$

The errors in this state are calculated as the difference between the current state x_k and some reference value r. In this simulation this will be the position and velocity of the landing platform moving in the inertial frame.

A. State Penalties and Q Parameter Tuning

In order to properly implement an MPC controller, penalties are required for the cost function the optimization problem. The controller should minimize the error quantities, e_p and $e_{\dot{p}}$, and drive the quadrotor towards the reference trajectory and velocity. Therefore, a Q matrix is chosen that produces this result, corresponding to the simplified state vector x below.

$$Q = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & Q_1 & 0 & 0 & 0 \\ 0 & 0 & Q_2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \; x = \begin{bmatrix} \Delta \\ e_p \\ e_{\dot{p}} \\ p \\ u \end{bmatrix}$$

In order to choose the coefficients Q_1 and Q_2, a simulation is run over the set $Q_1 \in [1, 250]$ and $Q_2 \in [0, 2]$. Two key characteristics are examined in this parameter tuning method. The first is the sum of the mean squared value of the position and velocity errors, which is shown below in Figure 2.

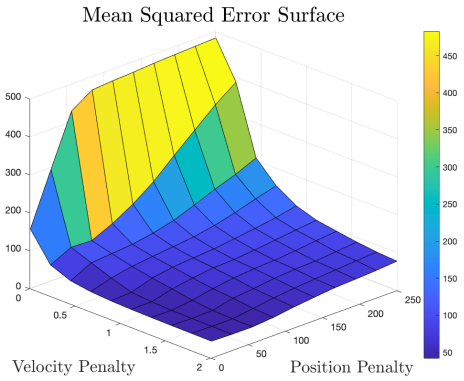


Fig. 2: Mean Squared Error Parameter Tuning

The second value is the sum of the final distance and velocity differences, shown in Figure 3. This error is used to approximate the convergence rate of each simulation, which was run over 100 time steps, or 1 second. A weight of 1 and 0.01 were applied to the final position and velocity, respectively, as the primary objective is to track the position of the beacon. The results are below.

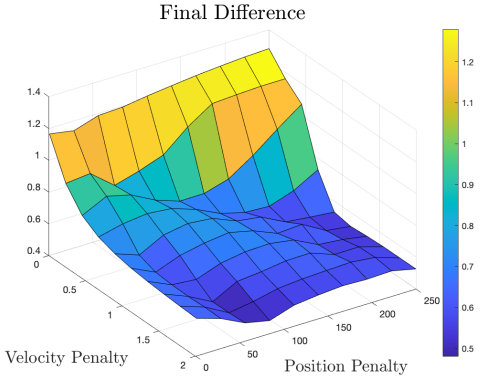


Fig. 3: Final Distance Error Parameter Tuning

From this parameter tuning, it was found that final distance could be minimized near values of $Q_1 = 200$ and $Q_2 = 1$, without extreme values in the mean squared error. Ultimately, these were the Q values chosen for the simulation. An R matrix with diagonals of 0.01 was chosen arbitrarily to drive the changes in the controller output to zero once the position and velocity errors had reached negligible values.

B. Terminal Conditions

The augmented system is not stabilizable and has no unique solution to the discrete algebraic Riccati equation (DARE). However, stability is introduced with an augmented DARE solution. Two implementations are chosen and compared in the Results section. The first is a terminal penalty constructed from only the DARE solution for the unmodified state shown in Equation 9. This solution is appended with zeros for all other augmented states. The second implementation uses the summation of the aforementioned DARE solution with the tuned Q matrix, thereby applying additional terminal penalties to the error terms.

C. Constraints

Making use of MPC's ability to integrate constraints into its design is a key driver behind using it for the landing approach. With constraints, the controller is guaranteed to satisfy bounds on the thrust signal and position such as $e_z > 0$. The state constraints are as below.

$$\begin{aligned} |\Delta\dot{p}| &\leq 1 \\ e_z &\geq 0 \\ 0 \leq u_i &\leq 1 \end{aligned} \tag{11}$$

Additional constraints are imposed on the controller output Δu.

$$|\Delta u_i| < 0.05 \tag{12}$$

D. Quadratic Program Design

A quadratic program must be designed based on the characteristics of the above problem to produce the MPC outputs. In particular, for a set of prediction steps and control steps, the controller should produce the optimal control output for the problem so as to minimize the position and velocity errors as provided above. A secondary consideration for the problem was the calculation time, as ideally this optimization would run in-the-loop of the quadrotor that is tracking the vehicle. With this in mind, prediction and control steps of $N = 5$ were chosen for the controller. For each step N, the constraints and controls are stacked such that they form the below matrices.

$$X = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_5 \end{bmatrix}, \; U = \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_4 \end{bmatrix} \tag{13}$$

Matrices are derived for the quadratic program per prior lectures [5]. These equations are designed to satisfy the state transition equation $x_k = A^k x_0 + \sum_{i=0}^{k-1} A^{k-1-i} B u_i$, which can be rewritten as $X = SU + Mx_0$.

$$S = \begin{bmatrix} B & 0 & \cdots & 0 \\ AB & B & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ A^4B & A^3B & \cdots & B \end{bmatrix}, \; M = \begin{bmatrix} A \\ A^2 \\ \vdots \\ A^5 \end{bmatrix} \tag{14}$$

With these matrices, constraints are specified on the system as below, using the constraints in Equation 11, which are stacked in the same way as Equation 13.

$$\begin{aligned} X_{min} \leq X = SU + Mx_0 \leq X_{max} \\ \begin{bmatrix} S \\ -S \end{bmatrix} U \leq \begin{bmatrix} X_{max} - Mx_0 \\ -X_{min} + Mx_0 \end{bmatrix} \end{aligned} \tag{15}$$

A similar treatment can be provided to the control constraints.

$$\begin{bmatrix} I \\ I \end{bmatrix} U \leq \begin{bmatrix} U_{max} \\ -U_{min} \end{bmatrix} \tag{16}$$

The optimization function that the MPC controller minimizes take the below form.

$$J_5 = \sum_{k=0}^{4} x_k^T Q + u_k^T R u_k + x_5^T P x_5 \tag{17}$$

In Equation 17, Q, R, and P are the state penalty, control penalty, and terminal penalty, respectively. To take advantage of the stacked shape of the X and U values in Equation 13, the Q, R, and P matrices are converted into block matrix form, as shown below.

$$\bar{Q} = \begin{bmatrix} Q & 0 & \cdots & 0 & 0 \\ 0 & Q & 0 & \cdots & 0 \\ \vdots & 0 & \ddots & & 0 \\ 0 & \vdots & & Q & 0 \\ 0 & 0 & \cdots & 0 & P \end{bmatrix}, \bar{R} = \begin{bmatrix} R & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & R \end{bmatrix} \tag{18}$$

This conversion changes Equation 17 to the below. The full derivation can be found in [5].

$$\begin{aligned} J_5 &= X^T \bar{Q} X + U^T \bar{R} U + x_0^T Q x_0 \\ &= U^T H U + 2q^T U + c \end{aligned} \tag{19}$$

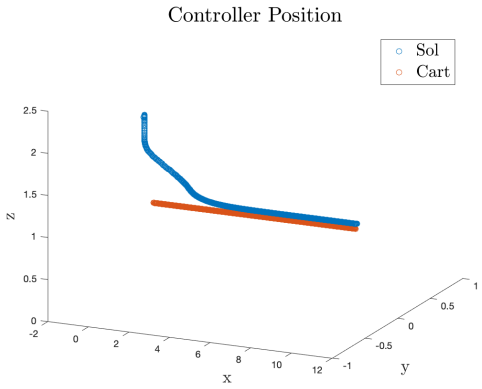
The cost is now a quadratic program in U and can therefore be solved by numerous methods, such as dual projected gradient, or through MATLAB's `quadprog.m`. The latter has been chosen for these simulations for its robustness.

V. Results

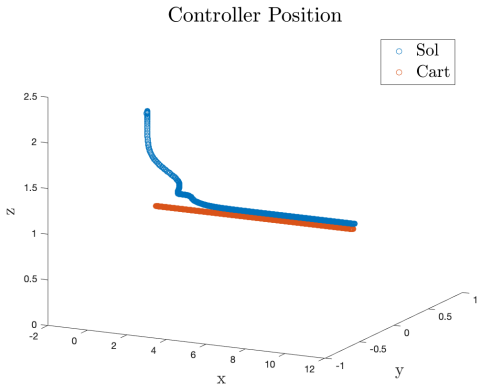
This section will discuss the results of two trials. The first is the pursuit of and landing on a platform moving linearly at 1 m/s, comparing controllers generated from the two aforementioned terminal conditions from Section IV-B. The second analysis will be on the quadrotor tracking a platform moving in a nonlinear fashion.

A. Linear Motion of Landing Platform

The controller shows good performance for tracking in straight line configurations. A landing platform moving 1 m/s in a straight line sees convergence in approximately 4 seconds from a position 1.1 meters away, seen in Figure 4.



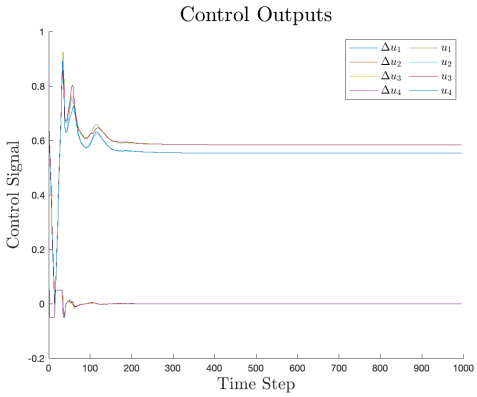
(a) DARE Solution



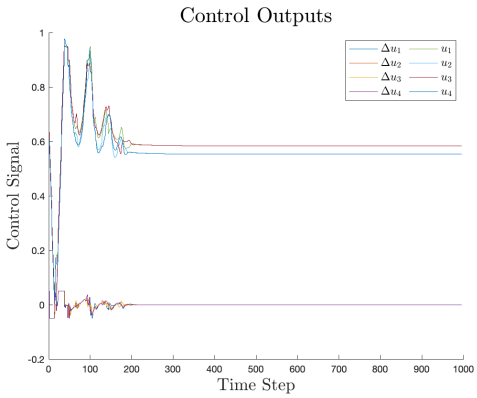
(b) DARE Solution + Tuned Q Matrix

Fig. 4: Quadrotor Tracking Paths

In the above paths, it is clear that the modified DARE solution induces extra noise into the controller. The path that this controller takes has a much larger variation in it, which could introduce instabilities in flight.



(a) DARE Solution



(b) DARE Solution + Tuned Q Matrix

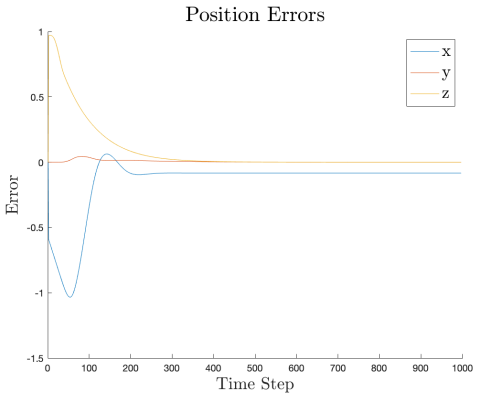
Fig. 5: Quadrotor Control Action

It is clearer to see the noise added by the modified DARE controller through the control actions. In the above Figure 5, the DARE-only solution has smooth control actions that accelerate the quadrotor and then decelerate it. In the modified DARE solution, there are non-smooth portions, noticeable in the Δu_i values on the lower part of the chart that cause the more chaotic behavior. The preferred solution would appear to be the DARE-only terminal con-

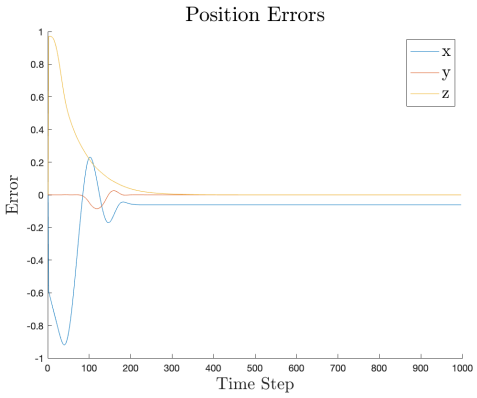
dition for its smoothness. In addition to the above, there is larger total control action in the modified DARE solution. This would imply that there would be a higher energy use in this controller, which is an important consideration when weighting controller schemes in energy-limited drone platforms. Further analysis could be done using economic MPC in future work.

1) Error Convergence

This section will cover the convergence of the position and velocity error for the DARE and modified DARE terminal conditions.



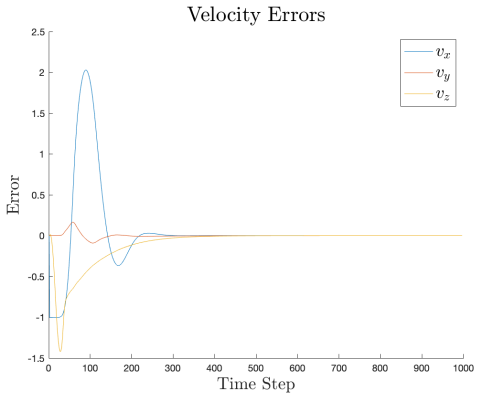
(a) DARE Solution



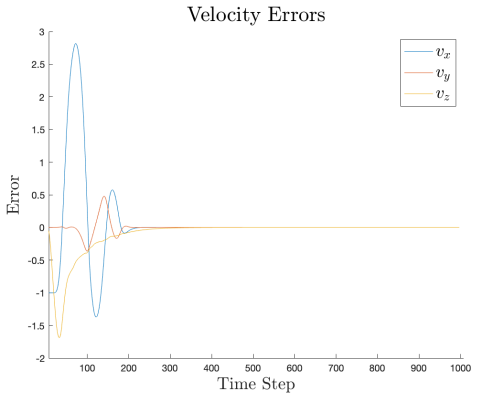
(b) DARE Solution + Tuned Q Matrix

Fig. 6: Position Error Measurements

In Figure 6, the DARE and modified DARE solutions show similar convergence along the z measurement. However, the modified DARE shows larger deviations in the x and y coordinate, which can be confirmed by the path in Figure 4b.



(a) DARE Solution



(b) DARE Solution + Tuned Q Matrix

Fig. 7: Velocity Error Measurements

In Figure 7 larger total variances in velocity are also seen in the modified DARE solution when compared to the DARE-only solution. However, these variances are not necessarily indicative of an inferior controller as they could produce faster convergence to the landing platform. There is marginally faster convergence, but this also introduces more perturbations in the velocity. Given that these could induce instabilities in non-disturbance-free environments, the DARE solution is the optimal choice for terminal conditions for the controller.

2) *Computation Time*

As mentioned earlier, a secondary consideration for this controller was its ability to compute control actions in-the-loop. This would be a vital requirement if it were to fly in the real world instead of along precomputed paths. For the current iteration of the linearized controller, good but not perfect performance is seen.

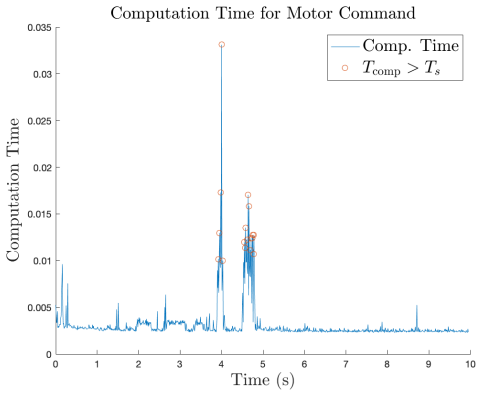


Fig. 8: Computation Time of Control Action

In Figure 8 there are 16 computations that exceed 0.01 seconds out of the 995 time steps that are run. Over 10 runs, the mean number of computations that exceed the sampling time is 7.1, accounting for 0.71% of all runs. With some controller optimization, a faster processor, or more efficient code, it is realistic to think that this controller could run in-the-loop at 100 Hz given any of these adjustments.

3) *Tracking Noisy Measurements*

One of the benefits of the integrator controller is that it can guarantee offset-free reference tracking in the presence of constant errors, and shows good convergence for random errors. This was tested using the linear landing platform track. The results are discussed below.

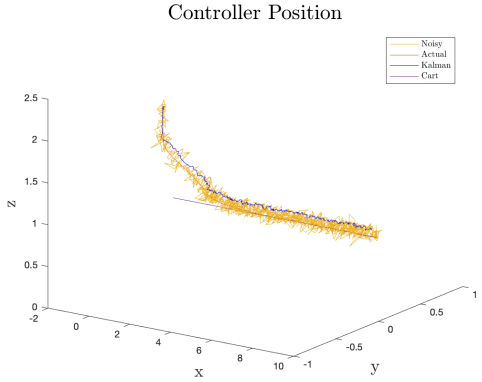
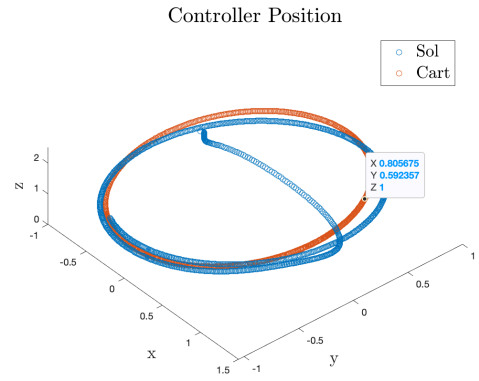


Fig. 9: Path with Noisy Measurements

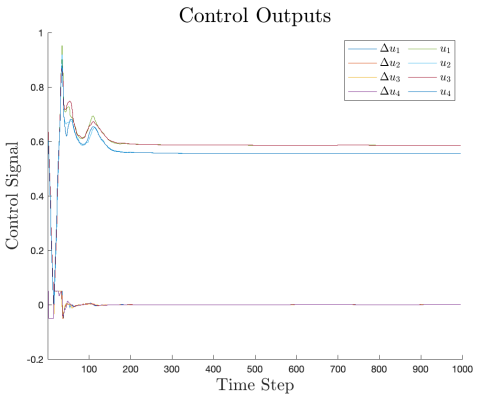
In Figure 9 the quadrotor is provided with noisy position and velocity measurements on the order of $\mathcal{N}_p(0, 0.05)$ and $\mathcal{N}_{\dot{p}}(0, 0.01)$. These measurements are shown in gold and the filtered position is shown in blue. This controller shows good, but not offset-free, tracking in the presence of these disturbances.

B. *Nonlinear Landing Platform Motion*

Prior sections have discussed the quadrotor's behavior when tracking a landing platform that is moving linearly. However, this is not a realistic case in many environments as the landing platform will have to maneuver in urban or suburban environments. To test the controller in a more realistic setting, a nonlinear path is generated for testing the quadrotor's ability to track and land on the platform. Similar to the last section, the landing platform is moving at 1 m/s, but now follows an elliptical path with a major axis of 1 m and a minor axis of 0.5 m. The results of this simulation are discussed below.



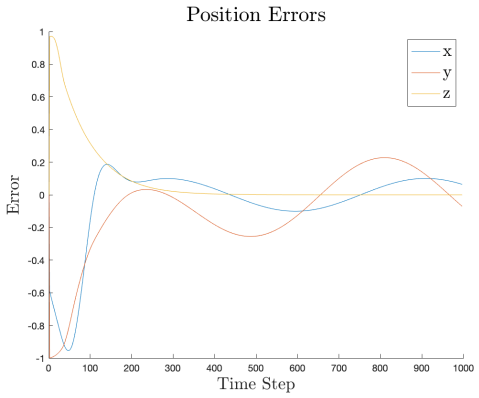
(a) Quadrotor Path



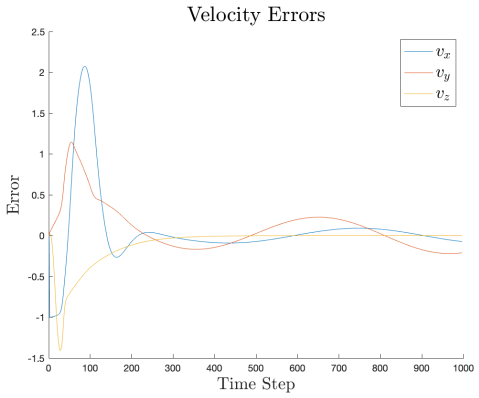
(b) Control Action

Fig. 10: Path and Control Action

Figure 10 shows that the tracking for the turning formulation is not as good as the tracking of the linear landing platform. In the 10 second run of the simulator, the minimum position tracking error occurs at 6.63 seconds at a distance of 8 cm. This is similar to the convergence of the linear controller, albeit at twice the convergence time. However, the magnitude of the velocity error at this point is 0.23 m/s with a vector form of $\begin{bmatrix}0.06 & 0.22 & 0\end{bmatrix}$, meaning that the quadrotor will overshoot the cart in the y direction on this approach, as seen in Figure 11. One modification that could be made to the control to account for this error in future work would be the inclusion of a Kalman estimator for the landing platform's position. Given past position and velocity measurements, this estimator could be used as part of the MPC model to minimize tracking error through time.



(a) Position Errors



(b) Velocity Errors

Fig. 11: Errors for Turning Landing Platform

VI. Conclusions and Future Work

This paper demonstrated the efficacy of a linearized model predictive controller using a modified state of Δ values and error terms. This controller builds integrator control directly into the solution, allowing for offset-free reference tracking in the presence of constant errors. The controller also shows good convergence for noisy position and velocity measurements, though it is not offset-free and maintains some error. These results also demonstrate the controller is able to track a landing plat-

form moving in a non-linear path, in this experiment an ellipse, though it does not show convergence to zero error in the simulation as the linear tracking did.

Future work for this controller is mentioned throughout the paper. Some pieces that require further investigation or work would be an analysis of using economic MPC to minimize the energy consumption of the control actions as these MAVs are generally highly energy constrained. Additionally, the implementation of a Kalman estimator into the controller, which would likely provide better tracking in the nonlinear case, remains an avenue to minimize controller errors. Additionally, this controller will be implemented onto Crazyflie 2.0 drones in the coming months in order to test the controller's real-world efficacy in landing a drone on a moving platform.

A nonlinear controller was also planned. This nonlinear controller was not working at the time of writing and additional work will be put into finishing this controller at a future date.

REFERENCES

[1] Dennis S. Bernstein and Ankit Goel. *Geometry, Statics, Kinematics, and Dynamics*, chapter 7.9 Euler's Equation for the Rotational Dynamics of a Rigid Body. 2021.

[2] Julian Förster. System identification of the crazyflie 2.0 nano quadrocopter. Master's thesis, ETH Zurich, Zurich, 2015-08.

[3] Dada Hu, Zhongcai Pei, and Zhiyong Tang. Single-parameter-tuned attitude control for quadrotor with unknown disturbance. *Applied Sciences*, 10(16), 2020.

[4] Ilya Kolmanovsky. Lecture on discrete time linear systems theory, module 2, 2021.

[5] Ilya Kolmanovsky. Lecture on discrete time linear systems theory, module 5, 2021.

[6] Pengcheng Wang, Zhihong Man, Zhenwei Cao, Jinchuan Zheng, and Yong Zhao. Dynamics modelling and linear control of quadcopter. In *2016 International Conference on Advanced Mechatronic Systems (ICAMechS)*, pages 498–503, 2016.

APPENDIX A
A_d AND B_d MATRICES

$$A_d(1:6) = \begin{bmatrix} 1.0 & 0 & 0 & 0.01 & 0 & 0 \\ 0 & 1.0 & 0 & 0 & 0.01 & 0 \\ 0 & 0 & 1.0 & 0 & 0 & 0.01 \\ 0 & 0 & 0 & 1.0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1.0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1.0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$A_d(7:12) = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -4.9\text{e}{-4} & 0 & 0 & -1.6\text{e}{-6} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -0.098 & 0 & 0 & -4.9\text{e}{-4} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1.0 & 0 & 0 & 0.01 & 0 & 0 \\ 0 & 1.0 & 0 & 0 & 0.01 & 0 \\ 0 & 0 & 1.0 & 0 & 0 & 0.01 \\ 0 & 0 & 0 & 1.0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1.0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1.0 \end{bmatrix}$$

$$B_d = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 2.79\text{e}{-6} & 2.70\text{e}{-6} & 2.79\text{e}{-6} & -2.70\text{e}{-6} \\ 3.12\text{e}{-4} & 3.02\text{e}{-4} & 3.12\text{e}{-4} & 3.02\text{e}{-4} \\ 0 & 0 & 0 & 0 \\ -1.11\text{e}{-3} & 1.08\text{e}{-3} & 1.11\text{e}{-3} & -1.08\text{e}{-3} \\ 0.062 & 0.060 & 0.062 & 0.060 \\ 0.034 & 0.033 & -0.034 & -0.033 \\ 0.034 & -0.033 & -0.034 & 0.033 \\ -1.78\text{e}{-3} & 1.72\text{e}{-3} & -1.78\text{e}{-3} & 1.72\text{e}{-3} \\ 6.85 & 6.64 & -6.85 & -6.64 \\ 6.82 & -6.60 & -6.82 & 6.60 \\ -0.356 & 0.344 & -0.356 & 0.344 \end{bmatrix}$$

APPENDIX B
BLOCK DYNAMICS

$$\frac{\partial}{\partial t}\begin{bmatrix} x \\ y \\ z \\ \dot{x} \\ \dot{y} \\ \dot{z} \\ \phi \\ \theta \\ \psi \\ \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ (\cos(\psi)\sin(\theta)\cos(\phi)+\sin(\psi)\sin(\phi))\sum_{i=1}^{4}(f_i) \\ \sin(\psi)\sin(\theta)\cos(\phi)-\cos(\psi)\sin(\phi))\sum_{i=1}^{4}(f_i) \\ \cos(\theta)\cos(\phi)\sum_{i=1}^{4}(f_i)-g \\ \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \\ I_{xx}^{-1}(\tau_\phi+(I_{zz}-I_{yy})\dot{\phi}) \\ I_{yy}^{-1}(\tau_\theta+(I_{xx}-I_{zz})\dot{\theta}) \\ I_{zz}^{-1}(\tau_\psi+(I_{yy}-I_{xx})\dot{\psi}) \end{bmatrix}$$

APPENDIX C
MATLAB PRINTOUT