2-8-2023

Quadcopter Control Using Single Network Adaptive Critics

Alberto Velazquez
The University of Texas Rio Grande Valley

Lei Xu
Kent State University, lei.xu@utrgv.edu

Tohid Sardarmehni
California State University, Northridge

Recommended Citation

Velazquez, A, Xu, L, & Sardarmehni, T. "Quadcopter Control Using Single Network Adaptive Critics." Proceedings of the ASME 2022 International Mechanical Engineering Congress and Exposition. Volume 5: Dynamics, Vibration, and Control. Columbus, Ohio, USA. October 30–November 3, 2022. V005T07A035. ASME. https://doi.org/10.1115/IMECE2022-96834

Proceedings of the ASME 2022
International Mechanical Engineering Congress and Exposition
IMECE2022
October 30-November 3, 2022, Columbus, Ohio

IMECE2022-96834

QUADCOPTER CONTROL USING SINGLE NETWORK ADAPTIVE CRITICS

Alberto Velazquez
University of Texas Rio Grande Valley,
Edinburg, TX

Lei Xu
Kent State University, Kent, OH

Tohid Sardarmehni*
California State University, Northridge,
Northridge, CA

ABSTRACT

In this paper, optimal tracking control is found for an input-affine nonlinear quadcopter using Single Network Adaptive Critics (SNAC). The quadcopter dynamics consists of twelve states and four controls. The states are defined using two related reference frames: the earth frame, which describes the position and angles, and the body frame, which describes the linear and angular velocities. The quadcopter has six outputs and four controls, so it is an underactuated nonlinear system. The optimal control for the system is derived by solving a discrete-time recursive Hamilton-Jacobi-Bellman equation using a linear in-parameter neural network. The neural network is trained to find a mapping between a target costate vector and the current states. The network's weights are iteratively trained using the least-squares approximation method until the maximum number of iterations or convergence is reached, and training begins at the final time and proceeds backward to the initial time. The trained neural controller applies online optimal feedback control that tracks a trajectory, minimizes control effort, and satisfies the optimality condition. The SNAC method provides a controller that can handle all initial conditions within the domain of training and all times less than the training's final time.

Keywords: Quadcopter, Optimal Control, Adaptive Dynamic Programming, Reinforcement Learning

1. INTRODUCTION

Small unmanned aerial vehicles (UAVs) can be used for a variety of tasks such as surveillance, reconnaissance, search and rescue, forestry, flood and fire tracking, package delivery, and agriculture. [1–4]. Small aerial drones are generally cost-effective and expendable, allowing missions into dangerous areas without endangering lives or otherwise requiring significant cost or human effort.

Several multi-rotor UAV designs exist [5, 6], with quadcopters being the most popular and commercially available. Quadcopter drones have four rotors in a square configuration and are located an equal distance from the center of mass. Despite having relatively simple hardware, quadcopter drones are nonlinear, underactuated, and complex systems with six degrees of freedom and four inputs. Several control strategies have been implemented to control quadcopter drones including: LQR and PID controllers [7, 8], backstepping controls [9, 10], sliding mode controls [10–12], feedback linearization [12], dynamic inversion [13], and reinforcement learning [14].

The control of a quadcopter can be formulated as an optimal control question. The principle of optimality states that for whatever the result of a set of initial states and controls, the remaining controls must be optimal [15]. Dynamic programming is one such way to achieve this. Dynamic programming solves a recursive Hamilton-Jacobi-Bellman equation backward in time to generate optimal controls that minimize a cost function [16]. This closed-loop control method uses discretized states to make a lookup table of optimal controls that can include constraints. However, dynamic programming is limited by the curse of dimensionality. As the number of state space, output space, and action space variables increases, the required data grows exponentially, limiting dynamic programming's practicality [17].

Approximate Dynamic Programming (ADP) is a solution to the curse of dimensionality [17]. ADP achieves this by estimating the value function rather than knowing the exact value function through the backward iterations seen in dynamic programming. This allows ADP to solve a problem forward in time. The solution to ADP comes in the form of various adaptive-critic (AC) algorithms [18]. Dual Heuristic Programming (DHP) is an AC algorithm that uses a dual-network structure where the actor-network maps the state and control variables while the critic-network maps between the state and costate [19, 20]. Single Network Adaptive Critics (SNAC) performs DHP using a single network [19]. SNAC implementations result in a decrease in training effort, computational resources, and storage memory [20].

In this paper, Finite-SNAC Algorithm 1 developed in Ref. [20] is used to control the quadcopter drone model derived in Ref. [21]. The quadcopter's state-space dynamics are divided

*Corresponding author: t.sardarmehni@gmail.com

into kinematic and kinetic models. Finite-SNAC is applied to the kinematic model with the linear and angular velocities as controls. To improve SNAC's performance, the states, the controls, and the reference trajectory are nondimensionalized and scaled. A helical trajectory is provided as the position reference signal, and the attitude reference signal is then derived by using small-angle approximations and defining the quadcopter's yaw. This results in a reference signal for all six degrees of freedom. Finite-SNAC uses a linear in-parameter neural network to find the mapping between the costate and the states. The trained neural network is used to find the next set of optimal controls given the current states. These controls (the linear and angular velocities) are used to algebraically derive the input controls to the overall quadcopter model. The resulting controller is then applied online for optimal feedback control that tracks the helical trajectory.

The rest of the paper is organized as follows: Section 2 describes the quadcopter dynamics and implementation, the Finite-SNAC algorithm is discussed in Section 3, the results and simulations are presented in Section 4, and Section 5 provides the conclusion and future work.

2. QUADCOPTER MODEL

Two reference systems need to be related to describe the mathematical model of the quadcopter: the fixed earth frame and the mobile aircraft body frame. The fixed earth frame uses the North-East-Down (O_{NED}) coordinate system while the mobile aircraft body frame describes the Aircraft-Body-Center (O_{ABC}) coordinate system. Linear and angular position are defined in the earth frame as the following vector: $[x\ y\ z\ \phi\ \theta\ \psi]^T$. Euler angles are used to describe the orientating of the quadcopter: ϕ describes the roll, θ describes the pitch, and ψ describes the yaw. In the aircraft body frame, linear and angular velocities are defined as: $[u\ v\ w\ p\ q\ r]^T$.

To relate the mobile aircraft body reference frame to the fixed earth reference frame, a combination of the following rotational matrices is used:

$$R_x(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\phi) & -sin(\phi) \\ 0 & sin(\phi) & cos(\phi) \end{bmatrix} \tag{1}$$

$$R_y(\theta) = \begin{bmatrix} cos(\theta) & 0 & sin(\theta) \\ 0 & 1 & 0 \\ -sin(\theta) & 0 & cos(\theta) \end{bmatrix} \tag{2}$$

$$R_z(\psi) = \begin{bmatrix} cos(\psi) & -sin(\psi) & 0 \\ sin(\psi) & cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3}$$

Different combinations of rotational matrices can result in greatly simplified dynamics that omit the yaw ψ, such as $R_{xyz}(\phi,\theta,\psi)$. The $R_{zyx}(\phi,\theta,\psi)$ combination is used in this paper.

$$R_{zyx}(\phi,\theta,\psi) = \tag{4}$$

$$\begin{bmatrix} c(\theta)c(\psi) & s(\phi)s(\theta)c(\psi) - c(\phi)s(\psi) & c(\phi)s(\theta)c(\psi) + s(\phi)s(\psi) \\ c(\theta)s(\psi) & s(\phi)s(\theta)s(\psi) + c(\phi)c(\psi) & c(\phi)s(\theta)s(\psi) - s(\phi)c(\psi) \\ -s(\theta) & s(\phi)c(\theta) & c(\phi)c(\theta) \end{bmatrix}$$

Here $c() = \cos()$ and $s() = \sin()$.

The rotational matrix will be used to relate the derivative of the linear position and the linear velocities between the two reference frames. The following angular transformation matrix can be used to relate the derivative of the angular positions to the angular velocities in a similar manner:

$$T(\phi,\theta) = \begin{bmatrix} 1 & sin(\phi)tan(\theta) & cos(\phi)tan(\theta) \\ 0 & cos(\phi) & -sin(\phi) \\ 0 & \frac{sin(\phi)}{cos(\theta)} & \frac{cos(\phi)}{cos(\theta)} \end{bmatrix} \tag{5}$$

Relating the two reference frames results in the kinematics of the quadcopter:

$$\begin{aligned} \dot{x} &= u[c(\psi)c(\theta)] \\ &\quad - v[c(\phi)s(\psi) - c(\psi)s(\phi)s(\theta)] \\ &\quad + w[s(\phi)s(\psi) + c(\phi)c(\psi)s(\theta)] \\ \dot{y} &= u[c(\theta)s(\psi)] \\ &\quad + v[c(\phi)c(\psi) + s(\phi)s(\psi)s(\theta)] \\ &\quad - w[c(\psi)s(\phi) - c(\phi)s(\psi)s(\theta)] \\ \dot{z} &= -u[s(\theta)] + v[c(\theta)s(\phi)] + w[c(\phi)c(\theta)] \\ \dot{\phi} &= p + q[s(\phi)t(\theta)] + r[c(\phi)t(\theta)] \\ \dot{\theta} &= q[c(\phi)] - r[s(\phi)] \\ \dot{\psi} &= q\frac{s(\phi)}{c(\theta)} + r\frac{c(\phi)}{c(\theta)} \end{aligned} \tag{6}$$

Using Newton's law for the total force and Euler's equation for the total torque, the following dynamic model in the body frame can be found:

$$\begin{aligned} f_x &= m(\dot{u} + qw - rv) \\ f_y &= m(\dot{v} - pw + ru) \\ f_z &= m(\dot{w} + pv - qu) \\ m_x &= \dot{p}I_x - qrI_y + qrI_z \\ m_y &= \dot{q}I_y + prI_x - prI_z \\ m_z &= \dot{r}I_z - pqI_x + pqI_y \end{aligned} \tag{7}$$

When solving for the external forces and moments, gyroscopic moments due to the motors, the ground effect, and the effects of environmental factors such as wind are ignored. The

resulting values are shown below:

$$\begin{aligned} f_x &= -mg[s(\theta)] \\ f_y &= mg[c(\theta)s(\phi)] \\ f_z &= mg[c(\theta)c(\phi)] \\ m_x &= \tau_x \\ m_y &= \tau_y \\ m_z &= \tau_z \end{aligned} \tag{8}$$

From Eq. (7), Eq. (8), and the kinematics of Eq. (6), the state-space model for the quadcopter can be found [21]:

$$\begin{aligned} \dot{x} &= u[c(\psi)c(\theta)] \\ &\quad - v[c(\phi)s(\psi) - c(\psi)s(\phi)s(\theta)] \\ &\quad + w[s(\phi)s(\psi) + c(\phi)c(\psi)s(\theta)] \\ \dot{y} &= u[c(\theta)s(\psi)] \\ &\quad + v[c(\phi)c(\psi) + s(\phi)s(\psi)s(\theta)] \\ &\quad - w[c(\psi)s(\phi) - c(\phi)s(\psi)s(\theta)] \\ \dot{z} &= -u[s(\theta)] + v[c(\theta)s(\phi)] + w[c(\phi)c(\theta)] \\ \dot{\phi} &= p + r[c(\phi)t(\theta)] + q[s(\phi)t(\theta)] \\ \dot{\theta} &= q[c(\phi)] - r[s(\phi)] \\ \dot{\psi} &= q\frac{s(\phi)}{c(\theta)} + r\frac{c(\phi)}{c(\theta)} \\ \dot{u} &= rv - qw - g[s(\theta)] \\ \dot{v} &= pw - ru + g[s(\phi)c(\theta)] \\ \dot{w} &= qu - pv + g[c(\theta)c(\phi)] - \frac{f_t}{m} \\ \dot{p} &= \frac{I_y - I_z}{I_x}rq + \frac{\tau_x}{I_x} \\ \dot{q} &= \frac{I_z - I_x}{I_y}pr + \frac{\tau_y}{I_y} \\ \dot{r} &= \frac{I_x - I_y}{I_z}pq + \frac{\tau_z}{I_z} \end{aligned} \tag{9}$$

The quadcopter model is defined as an input-affine system: $\dot{x} = f(x) + g(x)u$ where $x = [x\ y\ z\ \phi\ \theta\ \psi\ u\ v\ w\ p\ q\ r]^T$ represents the state vector and $u = [f_t\ \tau_x\ \tau_y\ \tau_z]^T$ is the control vector.

2.1 Kinematic and Algebraic Controls

Equation (9) describes a complex, nonlinear, and underactuated system. Separating the system into the kinematics of Eq. (6) and into the dynamics of Eq. (7) simplifies solutions to the system. Treating the linear and angular velocities, $[u\ v\ w\ p\ q\ r]^T$, as controls in Eq. (6) lets the actual control, $u = [f_t\ \tau_x\ \tau_y\ \tau_z]^T$, be found algebraically using the following:

$$\begin{aligned} f_t &= m(-\dot{w} + qu - pv + g[c(\theta)c(\phi)]) \\ \tau_x &= \dot{p}I_x - (I_y - I_z)rq \\ \tau_y &= \dot{q}I_y - (I_z - I_x)pr \\ \tau_z &= \dot{r}I_z - (I_x - I_y)pq \end{aligned} \tag{10}$$

2.2 Reference Signal

Provided with a reference signal for the position, a reference signal for the angles can be generated based on small-angle approximations [22]. Setting the yaw, ψ, to zero simplifies the relations between position and angles; however, it prevents the quadcopter from rotating, limiting applications where a fixed component, like a camera, is used. The reference signal approximations are shown below.

$$\begin{aligned} \ddot{x} &= -\theta\frac{f_t}{m} \\ \ddot{y} &= \phi\frac{f_t}{m} \\ \ddot{z} &= g - \frac{f_t}{m} \end{aligned} \tag{11}$$

3. FINITE-SNAC

To utilize Finite-SNAC on the the quadcopter dynamics derived in Eq. (9), the dynamics need to be described as a continuous-time input-affine system: $\dot{x}(t) = f(x(t)) + g(x(t))u(t)$. Where $x \in \mathbb{R}^n$ is the state vector, $u \in \mathbb{R}^m$ is the control, $f : \mathbb{R}^n \to \mathbb{R}^n$ is the drift dynamics of the system, and $g : \mathbb{R}^n \to \mathbb{R}^{n\times m}$ is the control dynamics of the system. The integers n and m are the number of states and the number of controls, respectively. This system is discritized using a small sample time, Δt, to yield the following equation:

$$x_{k+1} = F(x_k) + G(x_k)u_k, k \in [0, 1, ..., N-1] \tag{12}$$

where k represents the time step, $N = t_f/\Delta t$, and $F(x_k) = x_k + \Delta t f(x_k)$ and $G(x_k) = \Delta t g(x_k)$ are derived using Euler integration. In Finite-SNAC with trajectory tracking, the following cost function is minimized:

$$\begin{aligned} J = &\frac{1}{2}(x_N - r_N)^T S(x_N - r_N) \\ &+ \frac{1}{2}\sum_{k=0}^{N-1}((x_k - r_k)^T Q(x_k - r_k) + u_k^T R u_k) \end{aligned} \tag{13}$$

where $S \in \mathbb{R}^{n\times n}$ and $Q \in \mathbb{R}^{n\times n}$ are positive semi-definite matrices, and $R \in \mathbb{R}^{m\times m}$ is a positive definite matrix for penalizing the final states, states, and controls respectively. The reference signal is denoted as r. Minimizing the difference between the reference signal and the states minimizes the cost function and tracks the trajectory.

A discrete-time Hamilton-Jacobi-Bellmen (HJB) equation can be derived from Eq. (13). It shows that the optimal cost-to-go, denoted with *, is the minimization of the cost at step k plus the cost from k+1 to N-1.

$$\begin{aligned} J^*(x_k, k) = \min_{u_k}(&\frac{1}{2}((x_k - r_k)^T Q(x_k - r_k) + u_k^T R u_k) \\ &+ J(x_{k+1}, k+1)), k \in [0, 1, ..., N-1] \end{aligned} \tag{14}$$

To find a closed-form optimal control, the optimality condition: $\frac{\partial J(x_k,k)}{\partial u_k} = 0$ must be satisfied. The optimal control is shown below:

$$u_k^* = -R^{-1}G(x_k)^T\lambda_{k+1}(x_{k+1}^{u_k^*}), k \in [0, 1, ..., N-1] \tag{15}$$

where $x_{k+1}^{u_k^*}$ is the state at time k+1 along policy u_k^*, and λ_{k+1} is the costate vector and can be defined based on the following

equation:

$$\lambda_k(x_k) = \delta J(x_k, k)/\delta x_k \tag{16}$$

Finite-SNAC uses a neural network (NN) that outputs the costate vector λ_{k+1} given the current state vector x_k. The form of the NN is shown below:

$$\lambda_{k+1} = W_k^T \phi(x_k), k \in [0, 1, ..., N-1] \tag{17}$$

where W_k is the time-dependent weight matrix and $\phi(x_k)$ is a vector of smooth linearly-independent scalar basis functions. To train the NN, target costate vectors need to be found. In the final time step, N, the target costate vector, denoted with ^{t}, is the following:

$$\lambda_N^t = S(x_N - r_N) \tag{18}$$

The target costate vector at k+1 can be defined as

$$\lambda_{k+1}^t = Q(x_{k+1} - r_{k+1}) + A_{k+1}^T \lambda_{k+2} \tag{19}$$

where A_{k+1} is equal to

$$A_{k+1} = \frac{\delta x_{k+2}}{\delta x_{k+1}} = \frac{\delta(F(x_{k+1}) + G(x_{k+1})u_{k+1})}{\delta x_{k+1}} \tag{20}$$

Substituting Eq. (12), Eq. (15), and Eq. (17) into Eq. (19) results in the following:

$$\begin{aligned}\lambda_{k+1}^t = {} & Q(F(x_k) - G(x_k)R^{-1}G(x_k)^T W_k^T \phi(x_k) - r_{k+1}) \\ & + A_{k+1}^T W_{k+1}^T \phi(F(x_k) - G(x_k)R^{-1}G(x_k)^T W_k^T \phi(x_k))\end{aligned} \tag{21}$$

Finite-SNAC is trained recursively starting at k = N-1 and continuing backward to the initial time. Therefore, the value W_{k+1} is known at step k. As the target costate vector in Eq. (21) is a function of the current weights, the NN must be trained iteratively with the weights initialized to a random set of values. The training error can be defined as the difference between the current costate output by the NN and the target costate:

$$e_k = \lambda_{k+1} - \lambda_{k+1}^t = W_k^T \phi(x_k) - \lambda_{k+1}^t \tag{22}$$

Once a sufficiently low error is reached, or a maximum number of iterations have passed, the training for step k is concluded and step k-1 begins.

The procedure for training the neural network's weights is described in Algorithm 1.

Algorithm 1 FINITE-SNAC ALGORITHM - REF. [20]

procedure NEURAL NETWORK TRAINING
  Select an initial guess on $W_k^0, k = [0, 1, ..., N-1]$
  while $e_{N-1}(x_{N-1}) < tolerance$ **do**
    Randomly select x_{N-1} within the domain of interest
    Calculate λ_N^t through the process in Eq. (18)
    Train weights W_{N-1} on input-target pair $[x_{N-1}, \lambda_N^t]$
    Calculate training error $e_{N-1}(x_{N-1})$ using Eq. (22)
  end while
  for K = N-2 to 0 **do**
    while $e_k(x_k) < tolerance$ **do**
      Randomly select state vector x_k
      Calculate λ_{k+1}^t through the process in Eq. (21)
      Train weights W_k on input-target pair $[x_k, \lambda_{k+1}^t]$
      Calculate training error $e_k(x_k)$ Eq. (22)
    end while
  end for
  return Trained weights: W
end procedure

4. RESULTS AND DISCUSSION

The Finite-SNAC algorithm is applied to the kinematics of Eq. (6). Using unaltered states to train the neural network with the least-squares method results in singular or badly scales values. This is due to the potential similarities between the smooth linearly-independent scalar basis functions of $\phi(x_k)$, and the issue is exacerbated by the difference in magnitude between the linear positions and velocities and the angular positions and velocities. Nondimensionalizaton and scaling improves the performance of the Finite-SNAC algorithm. An example of an Euler integrated, nondimensionalized state-space equation is shown below:

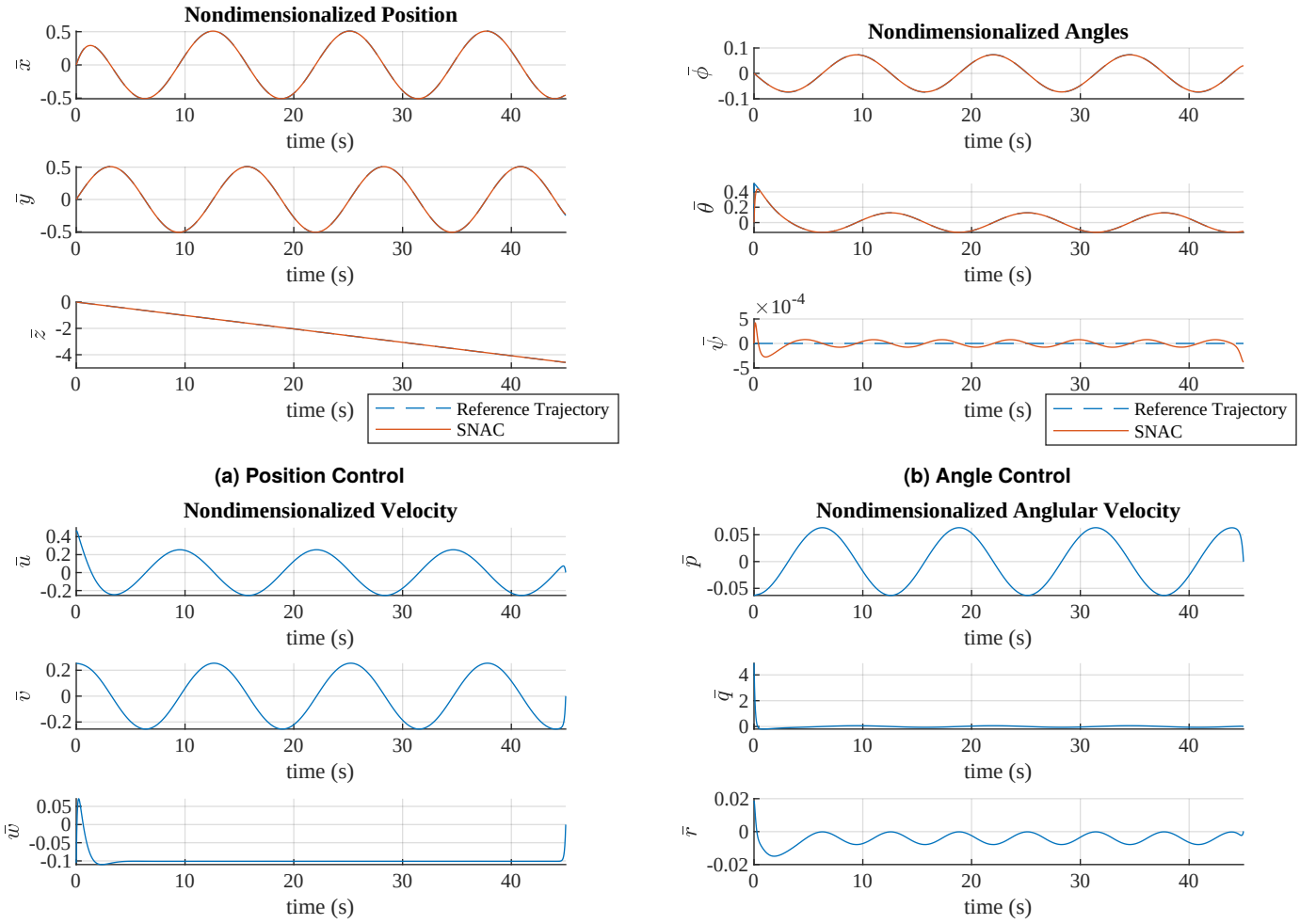
$$\bar{\theta}_{k+1} = \bar{\theta}_k + \frac{\Delta t}{\Theta}(\bar{q}_k Q[c(\bar{\phi}_k \Phi)] - \bar{r}_k R[s(\bar{\phi}_k \Phi)]) \tag{23}$$

where $\bar{\theta}$, $\bar{\phi}$, $\bar{q}$, and $\bar{r}$ represent the respective nondimensionalized states. Θ, Φ, Q, and R represent the nondimensionalizing and scaling values for their respective states. The same approach was applied to the remainder of the state-space equations in Eq. (9). The values of the nondimensionalization terms were arbitrarily chosen to ensure better scaling of the states. Nondimensionalization has a significant effect on reducing training divergence and improving network performance.

The reference signal is defined as a helix that begins and circles about the origin while increasing in height at a constant velocity. Equation (11) was used to find the reference signal for the six degrees of freedom of the quadcopter's kinematics. Figure 1a and Fig. 1b show the Finite-SNAC controller tracking the nondimensionalized position and angles effectively. Figure 1c and Fig. 1d show the optimal, nondimensionalized linear and angular velocities that are required to track the helical trajectory as seen in Fig. 2.

Using the values of Fig. 1c and Fig. 1d, the control $u = [f_t\ \tau_x\ \tau_y\ \tau_z]^T$ can be found using Eq (10). The nondimensionalized results are seen in Fig. 3a. The controls are lightly filtered to reduce the overall noise of the signals. Applying the controls to the state-space system in Eq. (9) results in the nondimensionalized trajectory shown in Fig. 3b. The results show the quadcopter trajectory deviating from the reference trajectory over time. This occurs regardless of filtering. The deviation is likely due to instabilities during the initial moments of simulations.

The results in Fig. 3b do not use online optimal feedback control. The desired optimal linear and angular velocities of Figure 1c and Fig. 1d were calculated until the final time. The desired velocities were then used to find the controls, $u = [f_t\ \tau_x\ \tau_y\ \tau_z]^T$, using Eq. (10), and they were then propagated to the system



(a) Position Control

(b) Angle Control

(c) Linear Velocity Output

(d) Angular Velocity Output

FIGURE 1: FINITE-SNAC CONTROLLING A QUADCOPTER'S KINEMATICS

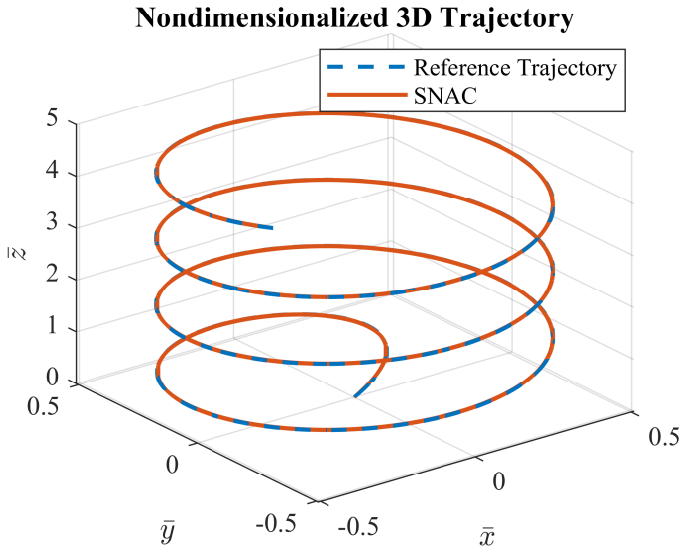
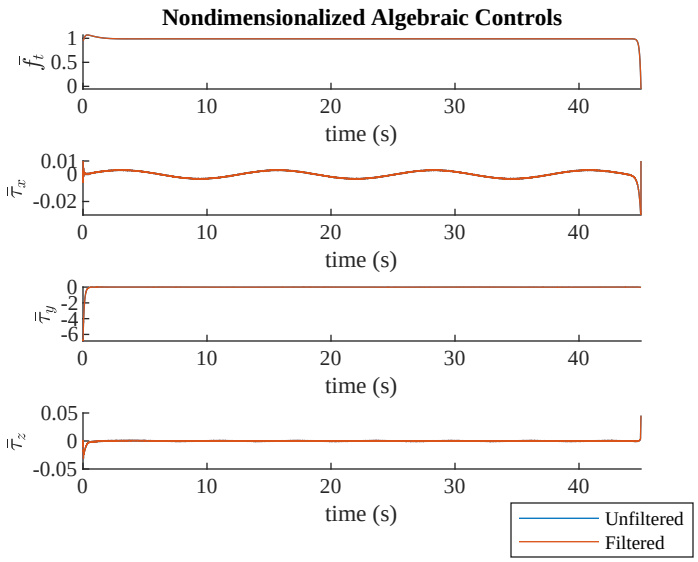
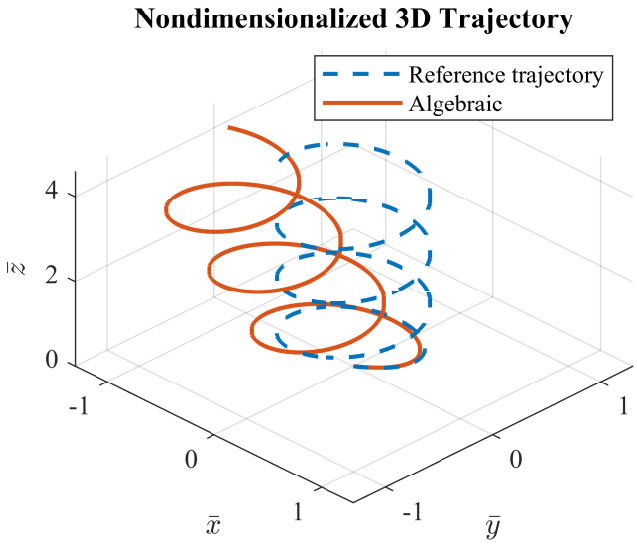


FIGURE 2: FINITE-SNAC CONTROLLER TRACKING A HELICAL TRAJECTORY



(a) Algebraic Controls



(b) Algebraic Control on Quadcopter Model

FIGURE 3: ALGEBRAIC CONTROLS AND RESULTING 3D TRAJECTORY

independently.

Implementing online optimal feedback control, where the optimal linear and angular velocities found with Eq. (15) are passed to Eq. (10) at each time step, resulted in divergence. Further research into divergence in the Finite-SNAC algorithm, particularly the linear in-parameter neural network and the domain of training, is needed.

5. CONCLUSION

The Finite-SNAC algorithm was used on a quadcopter's kinematics to find the optimal linear and angular velocities that track a certain trajectory. The result was used to algebraically find the optimal controls for the quadcopter's state-space dynamics. However, this solution failed to effectively track the trajectory. Future work into quadcopter control using Finite-SNAC can implement deep neural networks, include environmental perturbations such as wind in the dynamics, and apply online optimal feedback control to make the system more robust.

ACKNOWLEDGMENTS

This work was partially supported by the U.S. Department of Homeland Security under the award number 21STSLA00009-01-00, and by the U.S. National Science Foundation under the award number 2112650.

REFERENCES

[1] Lee, Dongwoo, Kim, Seungkeun and Suk, Jinyoung. "Formation flight of unmanned aerial vehicles using track guidance." *Aerospace Science and Technology* Vol. 76 (2018): pp. 412–420. DOI https://doi.org/10.1016/j.ast.2018.01.026. URL https://www.sciencedirect.com/science/article/pii/S1270963817302006.

[2] Baldazo, David, Parras, Juan and Zazo, Santiago. "Decentralized Multi-Agent Deep Reinforcement Learning in Swarms of Drones for Flood Monitoring." (2019): pp. 1–5DOI 10.23919/EUSIPCO.2019.8903067.

[3] Julian, Kyle D. and Kochenderfer, Mykel J. "Distributed Wildfire Surveillance with Autonomous Aircraft using Deep Reinforcement Learning." (2018). DOI 10.48550/ARXIV.1810.04244. URL https://arxiv.org/abs/1810.04244.

[4] Frachtenberg, Eitan. "Practical Drone Delivery." *Computer* Vol. 52 No. 12 (2019): pp. 53–57. DOI 10.1109/MC.2019.2942290.

[5] Mehmood, Hamza, Nakamura, Takuma and Johnson, Eric N. "A maneuverability analysis of a novel hexarotor UAV concept." *2016 International Conference on Unmanned Aircraft Systems (ICUAS)*: pp. 437–446. 2016. DOI 10.1109/ICUAS.2016.7502576.

[6] Zhu, He, Nie, Hong, Zhang, Limao, Wei, Xiaohui and Zhang, Ming. "Design and assessment of octocopter drones with improved aerodynamic efficiency and performance." *Aerospace Science and Technology* Vol. 106 (2020): p. 106206. DOI https://doi.org/10.1016/j.ast.2020.106206. URL https://www.sciencedirect.com/science/article/pii/S1270963820308889.

[7] Argentim, Lucas M., Rezende, Willian C., Santos, Paulo E. and Aguiar, Renato A. "PID, LQR and LQR-PID on a quadcopter platform." *2013 International Conference on Informatics, Electronics and Vision (ICIEV)*: pp. 1–6. 2013. DOI 10.1109/ICIEV.2013.6572698.

[8] Bouabdallah, S., Noth, A. and Siegwart, R. "PID vs LQ control techniques applied to an indoor micro quadrotor." *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, Vol. 3: pp. 2451–2456 vol.3. 2004. DOI 10.1109/IROS.2004.1389776.

[9] Madani, Tarek and Benallegue, Abdelaziz. "Backstepping Control for a Quadrotor Helicopter." *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*: pp. 3255–3260. 2006. DOI 10.1109/IROS.2006.282433.

[10] Bouabdallah, S. and Siegwart, R. "Backstepping and Sliding-mode Techniques Applied to an Indoor Micro Quadrotor." *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*: pp. 2247–2252. 2005. DOI 10.1109/ROBOT.2005.1570447.

[11] Xu, Rong and Ozguner, Umit. "Sliding Mode Control of a Quadrotor Helicopter." *Proceedings of the 45th IEEE Conference on Decision and Control*: pp. 4957–4962. 2006. DOI 10.1109/CDC.2006.377588.

[12] Lee, Daewon, Kim, H. Jin and Sastry, S. Shankar. "Feedback linearization vs. adaptive sliding mode control for a quadrotor helicopter." *International Journal of Control, Automation and Systems* Vol. 7 (2009): pp. 419–428.

[13] Das, Abhijit, Subbarao, Kamesh and Lewis, Frank L. "Dynamic inversion with zero-dynamics stabilisation for quadrotor control." *Iet Control Theory and Applications* Vol. 3 (2009): pp. 303–314.

[14] Koch, William, Mancuso, Renato, West, Richard and Bestavros, Azer. "Reinforcement learning for UAV attitude control." (2019). URL https://open.bu.edu/handle/2144/40560.

[15] Bellman, Richard and Kalaba, Robert E. "Dynamic Programming and Modern Control Theory." 1966.

[16] Kirk, Donald E. "Optimal control theory : an introduction." 1970.

[17] Powell, Warren B. "Approximate Dynamic Programming: Solving the Curses of Dimensionality." 2011.

[18] Konda, Vijay and Tsitsiklis, John. "Actor-Critic Algorithms." Solla, S., Leen, T. and Müller, K. (eds.). *Advances in Neural Information Processing Systems*, Vol. 12. 1999. MIT Press. URL https://proceedings.neurips.cc/paper/1999/file/6449f44a102fde848669bdd9eb6b76fa-Paper.pdf.

[19] Padhi, Radhakant, Unnikrishnan, Nishant, Wang, Xiaohua and Balakrishnan, S.N. "A single network adaptive critic (SNAC) architecture for optimal control synthesis for a class of nonlinear systems." *Neural Networks* Vol. 19 No. 10 (2006): pp. 1648–1660. DOI https://doi.org/10.1016/j.neunet.2006.08.010. URL https://www.sciencedirect.com/science/article/pii/S0893608006001912.

[20] Heydari, Ali and Balakrishnan, S.N. "Fixed-final-time optimal tracking control of input-affine nonlinear systems." *Neurocomputing* Vol. 129 (2014): pp. 528–539. DOI https://doi.org/10.1016/j.neucom.2013.09.006. URL https://www.sciencedirect.com/science/article/pii/S0925231213009065.

[21] Sabatino, Francesco Di. "Quadrotor control: modeling, nonlinearcontrol design, and simulation." 2015.

[22] Clemente, Salvatore. "Quadrotor control: implementation,cooperation and human-vehicleinteraction." 2015.