Evolutionary Nonlinear Model Predictive Control on a Fixed-Wing and Multirotor

Jaron Ellingson[1], Mathew Haskell[2]

***Abstract*— Real-time model predictive control (MPC) is limited to short time horizons and linear systems because the optimization complexity is too large with long time horizons and nonlinear systems. For this reason, MPC is typically accomplished using linearized models and convex optimization solvers. We seek to explore evolutionary algorithms allowing for nonlinear models and constraints, non-convex costs, and extended time horizons.**

Our contributions include extending nonlinear evolutionary MPC to flight vehicles, fixed-wing and multirotor UAVs, as well as enhancing the evolutionary algorithm. We also intend to parameterize the design space of the optimization to reduce solve times. These contributions validate the robust and effective nature of the algorithm.

I. Introduction

In [1], linear MPC was performed in real-time on a robotic arm using an evolutionary optimization algorithm rather than a typical convex solver. To make the algorithm real-time, each state propagation in the evolutionary algorithm during a single generation occurred in parallel on a graphics processing unit (GPU). GPU's are known to be fast at matrix multiplications, which the linearized model provided.

The previous work using an evolutionary algorithm was continued in [2] where nonlinear dynamics were used as the model for real-time MPC. The key change in this work from [1] is that the nonlinear dynamics were approximated using a neural network. The neural network was able to learn the nonlinear dynamics in a way that still utilized matrix multiplications, allowing for parallelization of the genetic algorithm on a GPU.

In [3], the evolutionary MPC algorithm is compared to a MPC using a QP solver both with a parameterized optimization design space to reduce the number of design variables. Both algorithms are capable of real-time control of nonlinear robotic arms. With MPC, the optimization design variable is the future trajectory of control inputs that should be applied to the system over a finite time horizon. This work used a piece-wise linear function to parameterize the trajectory of future inputs, usually with only 2 lines (3 points). This allowed for a significant reduction in the search space of the optimization and thus, faster solve times. With the parameterization, the solve times of both algorithms are capable of running control at over 100 Hz. MPC using the QP solver was still faster than the parallelized NEMPC, but the evolutionary algorithm allows for a nonlinear model.

[1] Jaron Ellingson is an MS candidate in the Department of Mechanical Engineering, Brigham Young University jaronce@byu.edu

[2] Mathew Haskell is a PhD candidate in the Department of Mechanical Engineering, Brigham Young University mathew.haskell@byu.edu

II. Model Predictive Control

The MPC problem is formulated by

$$\begin{aligned} \text{minimizing} \quad & J = \sum_{k=0}^{T-1} (f(\mathbf{x}_k, \mathbf{u}_k) - \mathbf{x}_d)^\top Q (f(\mathbf{x}_k, \mathbf{u}_k) - \mathbf{x}_d) \\ \text{with respect to} \quad & \mathbf{u}_k \\ \text{subject to} \quad & \mathbf{L}_b \leq \mathbf{u}_k \leq \mathbf{U}_b, \end{aligned} \tag{1}$$

where $f(\mathbf{x}_k, \mathbf{u}_k)$ represents the nonlinear dynamics of either the quadrotor or fixed-wing aircraft applied with Runge-Kutta 4th order approximation. $\mathbf{x}_k$ and $\mathbf{x}_d$ are our calculated state and desired state, $\mathbf{u}_k$ represents our inputs, $\mathbf{L}_b$ and $\mathbf{U}_b$ are our lower and upper bounds on the inputs, and Q is a positive semi-definite diagonal matrix which weights the cost of error associated with each state. In sections IV and V we will highlight the differences between the quadrotor and fixed-wing states, desired state, inputs, and cost matrix.

MCP optimizes over a finite time window or horizon to predict what would be an optimal control trajectory. The first control input from the optimal trajectory is applied for one time step and then resolved at the next time step. This pattern continues, essentially predicting future states and then comparing these states to desired states. Usually this problem is solved using linearized models and is solved efficiently with quadratic programming techniques. This linearized problem works well and is popular with nonlinear systems that behave linearly. MPC even works well for a quadrotor [4] and a fixed-wing [5]. This project explores using the complete nonlinear dynamic equations for solving the optimization for highly nonlinear systems like a quadrotor and fixed-wing aircraft. We believe that using the full nonlinear dynamics will allow the vehicles to fly more aggressively and efficiently compared to using only the linearized dynamics.

III. Evolutionary Algorithm

Most of the work which has been accomplished on our evolutionary algorithm developement was inspired by or implemented from [6].

A. Initialization

To initialize our first population, we choose to implement a Latin-Hypercube sampling algorithm to optimize starting coverage of the design space and allow for input saturation to be enforced. We also decided to insert 1 sample at equilibrium because we know that optimal solutions often lay

near equilibrium, especially at level or minimal movement flight.

Since it is likely that the current optimal command trajectory will be close to the optimum solved for at the previous time step, the population from the previous solve is used to initialize the population for the current solve. This is a way to warm start the optimization after the first solve. After the warm start feature is active, only 1 generation needs to be used because the population is already close to the optimum. Only running 1 generation of the evolutionary algorithm significantly decreases the solve time to facilitate real-time control.

B. Fitness

The cost function in Eq. (1) was used to evaluate the fitness of each member of the population, where the dynamics for a quadrotor and a fixed-wing are used for $f(\mathbf{x}_k, \mathbf{u}_k)$. Each input trajectory in the population is independent of the others, meaning that propagating a single generation can be computed in parallel. In this work, we did not parallelize generation propagation; however, our framework will allow for this feature to be added in the future.

C. Selection

For the selection process of the evolutionary algorithm, two methods were implemented for comparison. The first method kept a percentage of the most fit members of the population from generation to generation while introducing a certain number of strangers to form the mating pool. This yields a small mating pool were each member was paired enough times to fill up the population size. The second method was tournament style where each member of the population was randomly paired. The most fit of the pairs formed the first half of the mating pool and the process was repeated to form the second half. This created a mating pool of the same size as the population and the most fit member always appears twice in the pool while the least fit is eliminated. Both of these methods worked, but the tournament style ran faster and was more consistent while the first method had times where it didn't work as well.

D. Crossover

Two methods were implemented and compared for the crossover portion of the genetic algorithm. One involved creating a child by randomly choosing gene by gene with equal likelihood which parent the gene came from. This provided much variation mixing the parents. The other method was a linear crossover where two children were created as the following points:

$$\begin{aligned} \mathbf{x}_{c1} &= 0.5\mathbf{x}_{p1} + 0.5\mathbf{x}_{pf}, \\ \mathbf{x}_{c1} &= 2\mathbf{x}_{p1} - 1\mathbf{x}_{pf}, \end{aligned} \tag{2}$$

where $\mathbf{x}_{p1}$ is the more fit parent. The first child is the average of the two parents and the second child extrapolates in the direction of the parent with a lower cost. A couple of the best parents were also randomly inserted after the crossover step to ensure that the lowest cost in the population doesn't go up between generations. Both of the methods above function and provide diversity for the next generation. The simulation data in later sections used the linear crossover method because the code executed faster.

E. Mutation

For all mutations, a small amount of Gaussian noise was added. We tried both mutating entire members of the population and mutating each individual gene of each member of the population individually, each method occurring with low probability. It didn't seem to make a difference which of these two methods were used during the mutation phase.

IV. Quadrotor

This section covers the dynamics and control formulation used for a quadrotor. For this work, inertia was ignored by setting the inputs of the system to be angular velocities rather than torques and assuming the commanded angular velocities are achieved instantaneously. This is a similar assumption made in successive loop closure methods for PID controllers. The state $\mathbf{x}$ and input $\mathbf{u}$ are defined as

$$\mathbf{x} = \begin{bmatrix} \mathbf{p} \\ \boldsymbol{\Theta} \\ \mathbf{v} \end{bmatrix} \text{ and } \mathbf{u} = \begin{bmatrix} \mathrm{s} \\ \boldsymbol{\omega} \end{bmatrix} \tag{3}$$

where, $\mathbf{p}$ is the position of the vehicle in an inertial frame of reference, $\boldsymbol{\Theta}$ represents Euler angles describing the attitude of the vehicle, $\mathbf{v}$ is the velocity of the vehicle in it's own body frame of reference, s is the commanded throttle signal, and $\boldsymbol{\omega}$ is the commanded angular velocities in the body's frame of reference. The state derivative equations are shown in Eqs. 10, 11, and with 12 being substituted for Euler angle kinematics. The force term in Eq. 11 contains forces from the combined thrust of the propellers, gravity, and a linear drag term:

$$\frac{1}{m}\mathbf{f}^b = -g\frac{s}{s_e}\hat{\mathbf{e}} + gR_I^b\hat{\mathbf{e}} - c_d\mathbf{v}_{b/I}^b \tag{4}$$

where g is the gravity constant, s_e is the quadrotor's equilibrium throttle signal, c_d is a drag coefficient, and $\hat{\mathbf{e}}$ is a unit vector along the z-axis. The throttle term is a linear approximation of thrust.

Finally, the cost matrix and bound constraints from Eq. 1 were set as

$$\begin{aligned} &Q = \text{diagonal}(\big[10, 10, 100, 20.5, 20.5, 20, 9.8, 9.8, 10\big]). \\ &\mathbf{L}_b = \big[0, -\tfrac{\pi}{2}, -\tfrac{\pi}{2}, -\tfrac{\pi}{2}\big], \\ &\mathbf{L}_b = \big[1, \tfrac{\pi}{2}, \tfrac{\pi}{2}, \tfrac{\pi}{2}\big]. \end{aligned} \tag{5}$$

The reference command included values for all 3 position states as well as heading and the reference for all other states was set to 0.

V. Fixed-Wing

The fixed-wing evolutionary and control characteristics are described in this section. We start with the dynamics $f(\mathbf{x}_k, \mathbf{u}_k)$ represents the nonlinear dynamics of a fixed wing aircraft applied with Runge-Kutta 4th order approximation (see Appendix I and [7] for more details). $\mathbf{x}_k$ and $\mathbf{x}_d$ are our calculated state and desired state where,

$$\mathbf{x} = \begin{bmatrix} \mathbf{p} \\ \mathbf{v} \\ \mathbf{q} \\ \boldsymbol{\omega} \end{bmatrix}. \tag{6}$$

Furthermore, Q is our cost matrix and initialized as,

$$Q = \text{diagonal}(\begin{bmatrix} 0, 0, 100, 0, 0, 0, 50, 50, 50, 0, 0, 0 \end{bmatrix}). \tag{7}$$

We choose this particular cost because we don't necessarily want the aircraft to hit a particular location but to maintain a desired altitude and heading.

The desired heading is calculated by the position of the aircraft through a vector field. An example of a vector field can be seen in Figure 1. In this figure the aircraft is some distance away from the waypoint path w_{i-1} to w_i. Figure 1 shows the vector field which slowly places the aircraft onto a course to the next waypoint. This vector field allows us to define the commanded heading of the aircraft as we are far away from the desired path. this vector field is defined as,

$$\check{\chi} = \chi_l - \chi_\infty \frac{2}{\pi} \tan^{-1}\left(k_{path} \mathbf{e}_2\right), \tag{8}$$

where χ_l is the course angle of our line, and $\check{\chi}$ is the desired course angle. Furthermore, k_{path} is a positive gain and χ_∞ is the maximum allowed difference between $\check{\chi}$ and χ_l.

We also choose to implement a vector field for maintaining a course on a circular path. The vector field in this case is defined as

$$\begin{aligned} \check{\chi} =& \varphi + \lambda \left(\frac{2}{\pi} + \tan^{-1}\left(k_{orbit} \frac{d - \rho}{\rho} \right) \right) \\ \varphi =& atan2\left(p_e - c_e, p_n - c_n\right) + 2\pi m, \end{aligned} \tag{9}$$

where the orbit angle φ needs to wrapped, λ determines the direction of the vector field, p_e and p_n and east and north coordinates of the aircraft, c_e and c_n are the coordinates of the circle, d is the distance of the aircraft away from the center of the circle, and ρ is the circle radius (see [7] for more details on vector fields).

Our design variables $\mathbf{u}_k = \begin{bmatrix} s_a & s_e & s_t & s_r \end{bmatrix}^\top$ represent the control inputs to the system and are constrained to be within certain bounds. The aileron, elevator, and rudder are all constrained to be within $-\pi/2$ to $\pi/2$ radians and the throttle is constrained from 0 to 1.

We also experimented with the number of generations and population size. Figure 2 highlights convergence for different number of populations. The number of the population appears to matter less with this particular optimization than the number of generations. To allow for our optimization to converge, we choose to run the optimization for 200 generations to start the optimization and then for 1 generation every subsequent MPC solve.

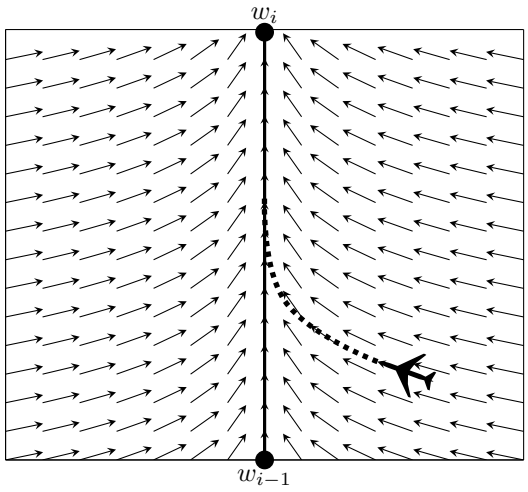


Fig. 1. A vector field which converges onto a path from one waypoint to another.

VI. Results

This section will highlight our results for both vehicles.

1) Quadrotor: Figure 3 shows the analysis of the quadrotor system to determine an appropriate population size and how many generations should occur in the first MPC solve. After around 200 generations, the cost appears to have converged; however, the small difference in cost from generation 200 to 500 makes a difference. The solution is very close to the actual minimum around generation 200, but the small amount of noise remaining shows up when using the evolutionary algorithm to control the quadrotor. A scenario was simulated where a quadrotor was commanded to move 1 meter East and 1 meter up in altitude from its initial position. This was simulated using a population of both 5000 and 500 where the first MPC solve ran for 200 generations. The simulation results are shown in Figures 4 and 5. Using a population size of 5000 provided smoother inputs and outputs with the system compared to the population of 500, which is to be expected. However, both of these simulations generated quite noisy inputs even though they both successfully reached the commanded state. Using the smaller population size allowed the simulation to run in real time, which is a promising sign considering that the algorithm has not yet been parallelized.

2) Fixed-Wing: Figures 6 to 8 show three test cases for the fixed-wing, level flight, merging into level flight, and circular flight. For each case we assumed that the fixed-wing would maintain an altitude of 100 meters and perform the necessary maneuvers to achieve the desired path.

VII. Conclusion

Overall, we consider this a proof of concept for using an evolutionary algorithm for optimizing the control of the

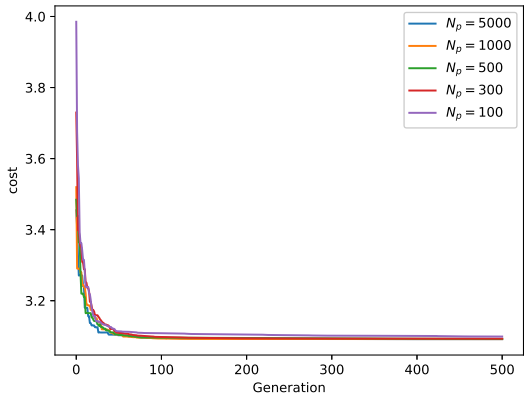


Fig. 2. Fixed-wing convergence.

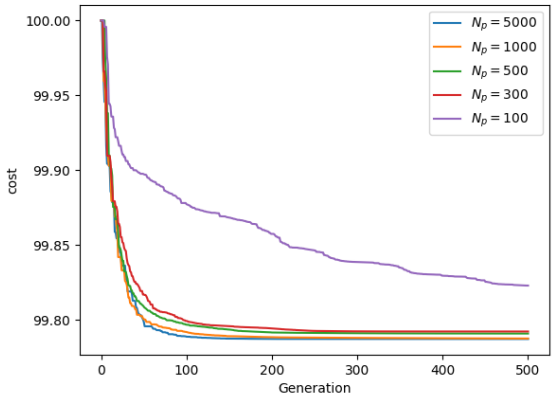


Fig. 3. Quadrotor convergence.

UAVs. We are pleased that we were able to control the aircraft in simulation but overall we do not think that the results are better than linear MPC. However, we do think that this is a good starting point for other research and believe that the methods presented can be improved upon.

Future work involves implementing this algorithm on a GPU to speed up the solving time. This might allow for more than one intermediate generation between time steps and result in more accurate and robust solutions. There is also a lot more room to explore all the variety of evolutionary algorithms. Furthermore, we would want to compare with method to other optimal control schemes such as MPC or LQR and compare these schemes on hardware.

APPENDIX I
AIRCRAFT DYNAMICS

The aircraft's position, velocity, attitude, and angular rate evolve in time according to

$$\dot{\mathbf{p}}_{b/I}^{I} = \left(R_I^b\right)^\top \mathbf{v}_{b/I}^{b} \tag{10}$$

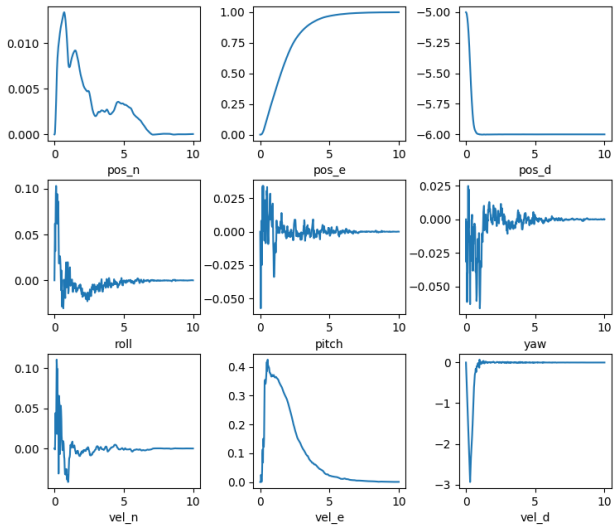
$$\dot{\mathbf{v}}_{b/I}^{b} = \frac{1}{m}\mathbf{f}^b - \boldsymbol{\omega}_{b/I}^{b} \times \mathbf{v}_{b/I}^{b} \tag{11}$$

$$\dot{\mathbf{q}}_I^b = \boldsymbol{\omega}_{b/I}^{b} \tag{12}$$

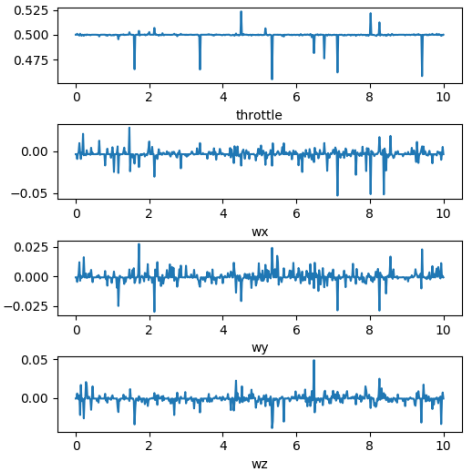
$$\dot{\boldsymbol{\omega}}_{b/I}^{b} = J^{-1}\left(\boldsymbol{\tau}^b - \boldsymbol{\omega}_{b/I}^{b} \times J\boldsymbol{\omega}_{b/I}^{b}\right), \tag{13}$$

where m is the aircraft's mass, J is the aircraft's inertia matrix, and $\mathbf{f}^b$ and $\boldsymbol{\tau}^b$ are the force and torque applied to the aircraft body [7].

We assume that the aircraft is equipped with the four control inputs: aileron, elevator, throttle, and rudder. The aircraft receives a throttle signal $s_t \in [0, 1]$ and signals for

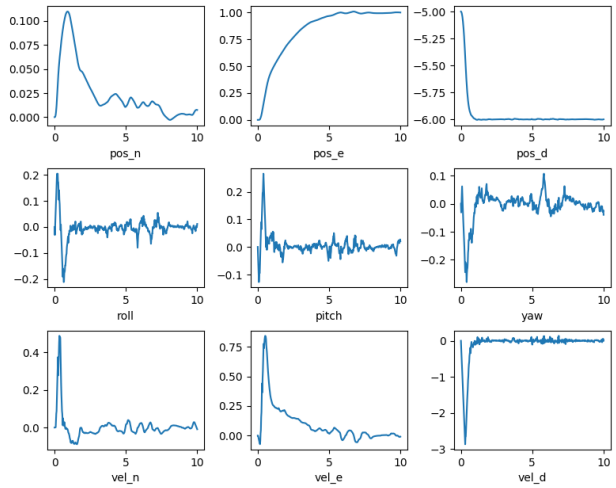


(a)

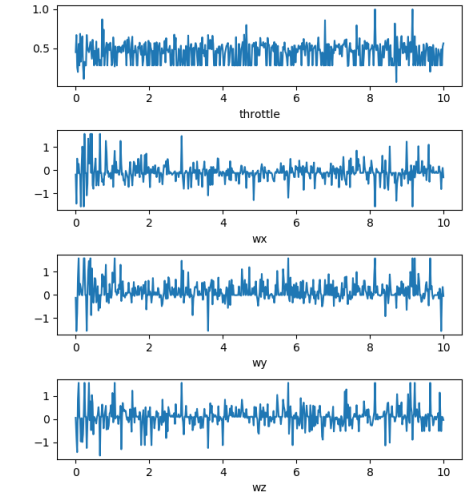


(b)

Fig. 4. Quadrotor simulation using a population size of 5000. Step commands of 1 meter East, 1 meter up in altitude, and 0 for all other states were used as the reference signal. This should theoretically provide the best inputs possible from the evolutionary algorithm after the original 200 generations. Figure 4a shows the state plots with positions states on the top row, attitude states on the middle row, and velocity states on the bottom row with units of meters and radians accordingly. Figure 4b shows the inputs generated from the NEMPC controller.

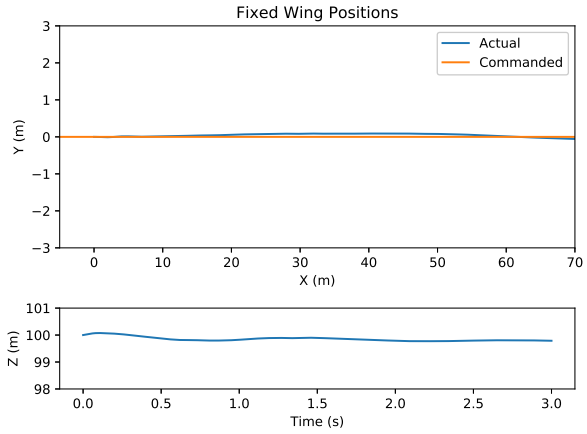


(a)

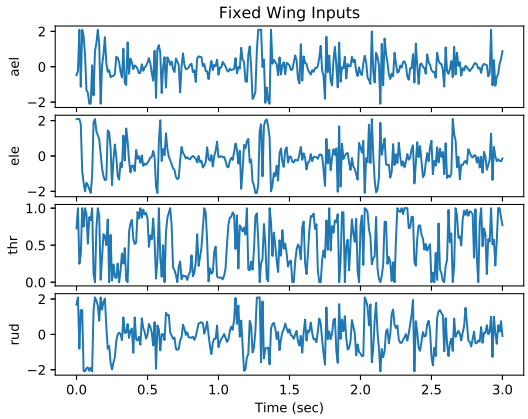


(b)

Fig. 5. Quadrotor simulation using a population size of 500. Step commands of 1 meter East, 1 meter up in altitude, and 0 for all other states were used as the reference signal. This should provide a realistic notion for the controller's capabilities since this simulation ran in real time. Figure 5a shows the state plots with positions states on the top row, attitude states on the middle row, and velocity states on the bottom row with units of meters and radians accordingly. Figure 5b shows the inputs generated from the NEMPC controller.

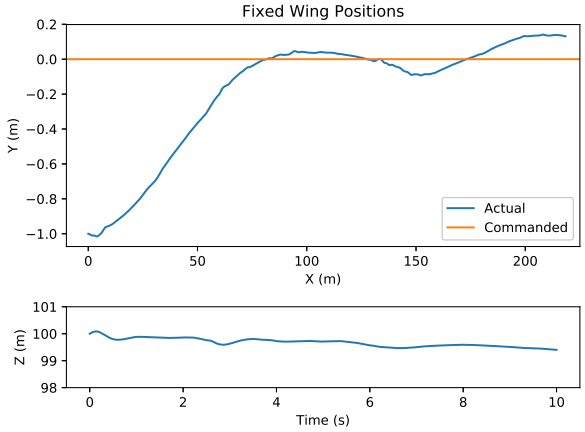


(a)

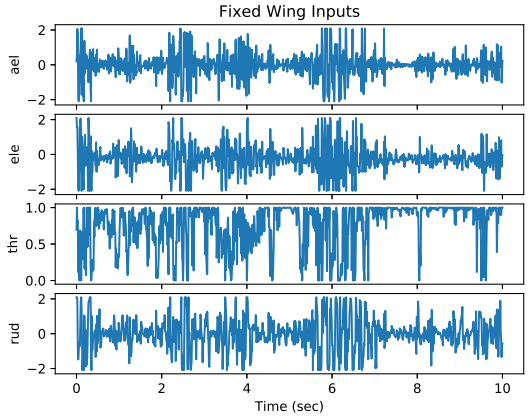


(b)

Fig. 6. Case 1 is the optimization of the aircraft in level flight. The aircraft started at the point (0,0,100) and was commanded to maintain the course on the commanded straight line. It took 37.45 seconds to run 3 seconds of simulation.

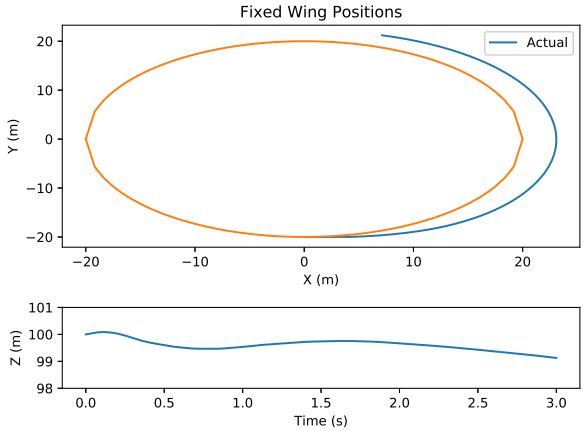


(a)

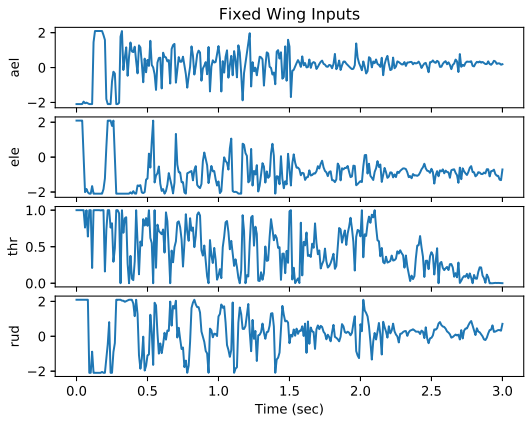


(b)

Fig. 7. Case 2 is the optimization of the aircraft in level flight but with an initial offset starting point. The aircraft started at the point (0,-1,100) and was commanded to maintain the course on the commanded straight line. It took 120.64 seconds to run 10 seconds of simulation.



(a)



(b)

Fig. 8. Case 2 is a base case to see if the optimization knows how to keep the aircraft in flight. It took 37.21 seconds to run 3 seconds of simulation.

aileron, elevator, and rudder given by

$$s_a = \frac{\delta_a}{\delta_{a_{max}}} \in [-1, 1] \tag{14}$$

$$s_e = \frac{\delta_e}{\delta_{e_{max}}} \in [-1, 1] \tag{15}$$

$$s_r = \frac{\delta_r}{\delta_{r_{max}}} \in [-1, 1], \tag{16}$$

where δ_* denotes deflection angle in radians and $\delta_{*_{max}}$ is the physically defined, maximum angle of deflection.

Vehicle air velocity, air speed, angle of attack, and side slip angle are defined by

$$\mathbf{v}^b_{a/I} = \mathbf{v}^b_{b/I} - R^b_I \mathbf{v}^I_{w/I} \tag{17}$$

$$V_a = \left\| \mathbf{v}^b_{a/I} \right\| \tag{18}$$

$$\alpha = \tan^{-1}\left(\frac{\mathbf{e}_3^\top \mathbf{v}^b_{a/I}}{\mathbf{e}_1^\top \mathbf{v}^b_{a/I}} \right) \tag{19}$$

$$\beta = \sin^{-1}\left(\frac{\mathbf{e}_2^\top \mathbf{v}^b_{a/I}}{V_a} \right), \tag{20}$$

where $\mathbf{v}^I_{w/I}$ is the wind velocity expressed in the inertial frame.

Nondimensionalized coefficients of lift and drag are defined by

$$C_L(\alpha) = (1 - \sigma(\alpha))\left[C_{L_0} + C_{L_\alpha}\alpha\right] + \sigma(\alpha) \left[2\text{sign}(\alpha)\sin^2\alpha\cos\alpha\right] \tag{21}$$

$$C_D(\alpha) = C_{D_p} + \frac{S\left(C_{L_0} + C_{L_\alpha}\alpha\right)^2}{\pi e b^2}, \tag{22}$$

where

$$\sigma(\alpha) = \frac{1 + e^{-M(\alpha-\alpha_0)} + e^{M(\alpha+\alpha_0)}}{\left(1 + e^{-M(\alpha-\alpha_0)}\right)\left(1 + e^{M(\alpha+\alpha_0)}\right)}. \tag{23}$$

Nondimensionalized coefficients of force in the body x and z axes are therefore given by

$$C_X(\alpha) = -C_D(\alpha)\cos\alpha + C_L(\alpha)\sin\alpha \tag{24}$$

$$C_{X_q}(\alpha) = -C_{D_q}\cos\alpha + C_{L_q}\sin\alpha \tag{25}$$

$$C_{X_{\delta_e}}(\alpha) = -C_{D_{\delta_e}}\cos\alpha + C_{L_{\delta_e}}\sin\alpha \tag{26}$$

$$C_Z(\alpha) = -C_D(\alpha)\sin\alpha - C_L(\alpha)\cos\alpha \tag{27}$$

$$C_{Z_q}(\alpha) = -C_{D_q}\sin\alpha - C_{L_q}\cos\alpha \tag{28}$$

$$C_{Z_{\delta_e}}(\alpha) = -C_{D_{\delta_e}}\sin\alpha - C_{L_{\delta_e}}\cos\alpha. \tag{29}$$

Consequently, the force and torque expressed in the body frame are given by

$$\mathbf{f}^b = mR^b_I\mathbf{g}^I + \frac{\rho V_a^2 S}{2}\Bigg(C_F(\alpha,\beta) + \frac{1}{2V_a}C_{F_\omega}(\alpha)\boldsymbol{\omega}^b_{b/I} + \tag{30}$$

$$C_{F_u}(\alpha)\mathbf{u}\Bigg) + \rho S_{prop} C_{prop}\mathbf{e}_3^\top\mathbf{u}\left(V_a \right. \tag{31}$$

$$\left. + \mathbf{e}_3^\top\mathbf{u}\left(k_{motor} - V_a\right)\cdot\left(k_{motor} - V_a\right)\mathbf{e}_1 \right.$$

$$\boldsymbol{\tau}^b = \frac{\rho V_a^2 S}{2} C_{bc}\Bigg(C_\tau(\alpha,\beta) + \frac{1}{2V_a}C_{\tau_\omega}\boldsymbol{\omega}^b_{b/I} + C_{\tau_u}\mathbf{u}\Bigg) \tag{32}$$

$$- k_{T_p}\left(k_\Omega \mathbf{e}_3^\top\mathbf{u}\right)^2\mathbf{e}_1,$$

where

$$\mathbf{u} = \begin{bmatrix} s_a & s_e & s_t & s_r \end{bmatrix}^\top \tag{33}$$

$$C_F(\alpha,\beta) = \begin{bmatrix} C_X(\alpha) \\ C_{Y_0} + C_{Y_\beta}\beta \\ C_Z(\alpha) \end{bmatrix} \tag{34}$$

$$C_{F_\omega}(\alpha) = \begin{bmatrix} 0 & C_{X_q}(\alpha)\,c & 0 \\ C_{Y_p} b & 0 & C_{Y_r} b \\ 0 & C_{Z_q}(\alpha)\,c & 0 \end{bmatrix} \tag{35}$$

$$C_{F_u}(\alpha) = \begin{bmatrix} 0 & C_{X_{\delta_e}}(\alpha)\,\delta_{e_{max}} & 0 & 0 \\ C_{Y_{\delta_a}}\delta_{a_{max}} & 0 & 0 & C_{Y_{\delta_r}}\delta_{r_{max}} \\ 0 & C_{Z_{\delta_e}}(\alpha)\,\delta_{e_{max}} & 0 & 0 \end{bmatrix} \tag{36}$$

$$C_{bc} = \begin{bmatrix} b & 0 & 0 \\ 0 & c & 0 \\ 0 & 0 & b \end{bmatrix} \tag{37}$$

$$C_\tau(\alpha,\beta) = \begin{bmatrix} C_{l_0} + C_{l_\beta}\beta \\ C_{m_0} + C_{m_\alpha}\alpha \\ C_{n_0} + C_{n_\beta}\beta \end{bmatrix} \tag{38}$$

$$C_{\tau_\omega} = \begin{bmatrix} C_{l_p} b & 0 & C_{l_r} b \\ 0 & C_{m_q} c & 0 \\ C_{n_p} b & 0 & C_{n_r} b \end{bmatrix} \tag{39}$$

$$C_{\tau_u} = \begin{bmatrix} C_{l_{\delta_a}}\delta_{a_{max}} & 0 & 0 & C_{l_{\delta_r}}\delta_{r_{max}} \\ 0 & C_{m_{\delta_e}}\delta_{e_{max}} & 0 & 0 \\ C_{n_{\delta_a}}\delta_{a_{max}} & 0 & 0 & C_{n_{\delta_r}}\delta_{r_{max}} \end{bmatrix}. \tag{40}$$

REFERENCES

[1] Phillip Hyatt and Marc D Killpack. Real-time evolutionary model predictive control using a graphics processing unit. In *2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*, pages 569–576. IEEE, 2017.

[2] Phillip Hyatt. Real-time nonlinear model predictive control using a graphics processing unit. 2019.

[3] Phillip Hyatt, Connor S. Williams, and Marc D. Killpack. Parameterized and gpu-parallelized real-time model predictive control for high degree of freedom robots, 2020.

[4] Moses Bangura and Robert Mahony. Real-time model predictive control for quadrotors. *IFAC Proceedings Volumes*, 47(3):11773–11780, 2014.

[5] Thomas J Stastny, Adyasha Dash, and Roland Siegwart. Nonlinear mpc for fixed-wing uav trajectory tracking: Implementation and flight experiments. In *AIAA Guidance, Navigation, and Control Conference*, page 1512, 2017.

[6] JRRA Martins, A Ning, and J Hicken. Multidisciplinary design optimization. *Unpublished, Compiled January*, 5:9, 2017.

[7] Randal W Beard and Timothy W McLain. *Small unmanned aircraft: Theory and practice*. Princeton university press, 2012.