Design of Model Reference Adaptive Controller for Uncertainty in Quadcopter Mass

Ryan Koeppen
Department of Mechanical Engineering
Fall 2019 16.31 Final Project Report

***Abstract*—For many dynamical systems, designing a stable controller to achieve a desired closed-loop dynamic response requires reasonably accurate knowledge or measurement of the system's open-loop dynamics. In some cases, it is not possible to accurately predict those dynamics due to uncertainties in system parameters (such as mass). One solution to this problem is to use an adaptive controller, which adjusts controller gains in response to errors in closed-loop response relative to a predicted response. In this work, a model reference adaptive controller (MRAC) was developed and implemented on a Parrot Mambo quadcopter to demonstrate controller adaptation in response to unexpected or changes in vehicle mass. Solutions to implementation problems are discussed, and simulation results demonstrate stability of the controller for a pulse maneuver in drone altitude.**

I. Introduction

In classical methods for controller design of dynamical systems, development of a controller typically requires accurate knowledge of the system of interest. For example, for a linear, time-invariant system with following form $\dot{x} = Ax + Bu$, the matrices A and B are typically known either by modeling the system or through empirical system identification. Controllers can then be designed using various methods (such as pole placement or LQR design) to achieve desired closed-loop dynamic behavior.

One limitation to these control schemes is that they are not typically robust to uncertainties or changes in the system's open-loop dynamics. One specific example is that many systems may experience an unknown change in mass directly preceding or during operation. For example, commercial airplanes may vary widely in mass depending on the number of people and bags onboard but must still maintain stable operation. Manufacturing robots may need to move objects of varying size and weight while maintaining desired closed-loop response times. Quadrotor drones, like the one used in this work, are also susceptible to instability due to added mass. The drone may, for example, have an attachment that allows it to pick up and drop masses on command. The attachment may be off-center from the drone's center of mass, potentially leading to steady state error and instability in altitude as well as attitude.

One solution to overcome uncertainties in plant dynamics is to use an adaptive controller. An initial condition for the controller, assuming the system dynamics are as expected or measured, is designed using a classical control method. As the system operates, an adaptation mechanism is used to update the controller parameters and/or the estimated plant parameters. The ultimate goal of this control scheme is to adjust these parameters such that the closed-loop dynamics of the unknown system match the desired closed-loop dynamics.

In this work, a full-state adaptive controller was designed in order to compensate for off-center, added mass on a Parrot Mambo minidrone. The controller was implemented in Simulink, simulated with a model of the minidrone, and tuned to achieve stable and desirable results. The controller was then implemented on the physical minidrone and experiments were conducted to verify similarity to simulation performance. This paper focuses on the theoretical basis for the adaptive controller as well as its implementation in Simulink for simulation and deployment to hardware.

II. Model Reference Adaptive Control for MIMO Systems

One popular adaptive control scheme is known as *model reference adaptive control* (MRAC). With MRAC, the goal is to update controller parameters without necessarily knowing or estimating the plant parameters. For the purposes of this work, the drone mass and center-of-mass are unknown or different from modeled or measured values, and the controller evolves over time to compensate for these uncertainties.

To understand this concept, consider the following controller and linear, time-invariant plant:

$$\dot{x} = Ax(t) + Bu(t) \quad (1)$$

$$u(t) = K_x^T(t)x(t) + K_r^T(t)r(t) \quad (2)$$

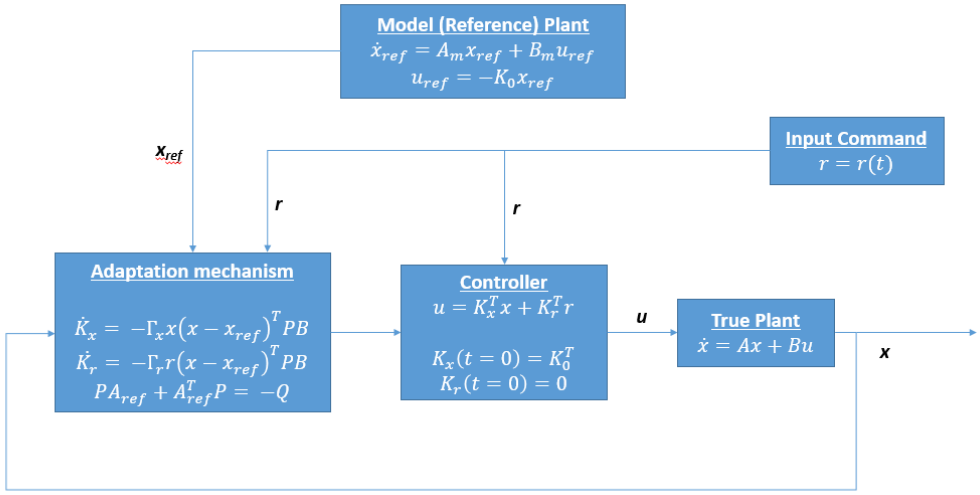

Fig. 1. Block diagram of a model reference adaptive control scheme as developed by Lavretsky et al [1]. The reference model is simulated separately from the real drone dynamics and is used to update controller gains with an adaptation mechanism.

where $x \in \mathbb{R}^n$ is a vector of states of the drone, $u \in \mathbb{R}^m$ is a vector of inputs to the drone , and $r \in \mathbb{R}^n$ is a bounded command signal for the drone. In our case, r represents the desired trajectory, attitude, and velocities of the drone at a particular time t. It should be noted that, in classical control, K_x and K_r would be time-invariant. With MRAC, however, these gains must evolve over time to compensate for the un-modeled dynamics from the added mass. Because this added mass may affect multiple states (for example, altitude and pitch), this system requires a full-state adaptive controller.

The controller developed in this work was based off the direct model reference adaptive controller detailed by Lavretsky et al [1]. A high level block diagram of the control scheme is shown in Figure 1. To update K_x and K_r, a reference model of the system is needed to compare the actual drone performance to the desired closed-loop dynamics. The reference model takes the following form:

$$\dot{x}_{ref} = A_{ref} x_{ref} + B_{ref} r \quad (3)$$
$$A_{ref} = A_m - B_m K_0 \quad (4)$$

A_{ref} represents the *linearized* desired closed-loop dynamics of the system. A_m and B_m correspond to the reference model plant. For this application, we assume that there is no uncertainty associated with the input commands, so $B_m \approx B$. K_0 is the controller gain matrix found using classical controller design methods such that when there is no added mass on the drone, $u(t) = -K_0 x(t)$. $x_{ref}(t)$ is the estimate for the system's states based on this given reference model.

For a given reference signal $r(t)$, the reference model is simulated alongside the actual system, allowing x_{ref} and x to be compared as the system evolves. Lavretsky et al derives the multi-input, multi-output (MIMO) adaptive controller using Lyapunov stability arguments subject to the constraints that $A + BK_x^T = A_{ref}$ and $BK_r^T = B_{ref}$. The result of the analysis is the following adaptation laws for the controller gains K_x and K_r:

$$\dot{K}_x = -\Gamma_x x e^T PB \quad (5)$$
$$\dot{K}_r = -\Gamma_r r e^T PB \quad (6)$$

In the above equations, P solves the algebraic Lyapunov equation $PA_{ref} + A_{ref}^T P = -Q$ (for some positive definite matrix Q) and $e = x - x_{ref}$ represents the error between the actual system's states and the simulated reference model's states. It is clear that as $e \to 0$, the above adaptation laws will go to zero and the controller gains will remain constant. Γ_x and Γ_r represent adaptation gains and are analogous to how quickly K_x and K_r (respectively) should adapt to modeling errors. With this MRAC scheme, there are four matrices which can be tuned to achieve desired results: A_{ref} (the desired closed-loop dynamics), Γ_x, Γ_r, and Q.

III. Simulink Implementation

A. MRAC Implementation for Hardware

Once the MRAC theory was understood from the derivation by Lavretsky et al, the adaptive controller needed to be implemented in software with the eventual goal of being deployed to the Parrot Mambo hardware. The adaptive controller required the following parameters to be defined:

a) $r(t)$
b) A_{ref}, B, and K_0
c) Γ_x, Γ_r, and Q

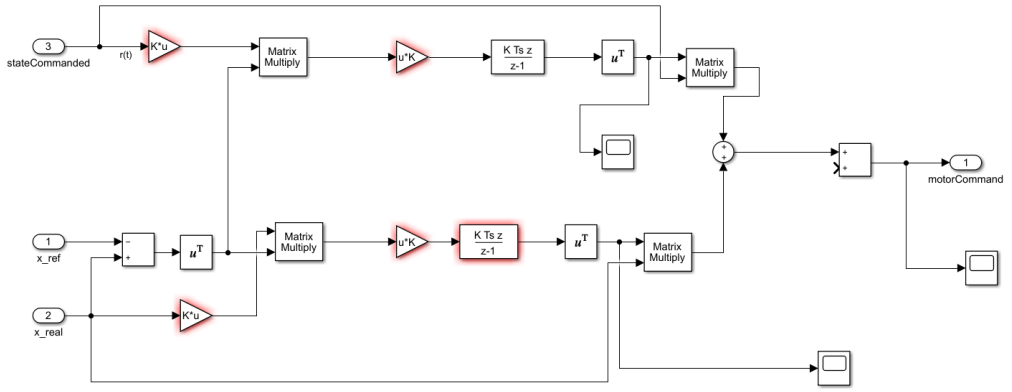

Fig. 2. Simulink diagram of the implemented model reference adaptive controller. Discrete time integration blocks are used to allow the controller to be used both in simulation and onboard the drone.

$r(t)$ is defined by the desired task. For the initial simulations discussed in this paper, $r(t)$ consisted of a pulse maneuver for the drone altitude and was zero for all other states. For the items in (b), the drone dynamics were modeled using the "complexDrone" model from class assignments. The dynamics were then linearized around a hover point of 1 meter above the ground. K_0 was defined to be the pole placement controller developed in lab 2 corresponding to the fast closed-loop hover dynamics.

The items in (c) were design parameters to be chosen. These matrices were initially chosen to be diagonal matrices since, to first approximation, each element on the diagonal corresponds to the adaptation rate of a different state. Using diagonal matrices eliminates complexities associated with tuning coupled adaptation rates. The exact values for these matrices are discussed in later sections with simulation results. Once these matrices were chosen, the matrix P could be calculated using the "lyap" function in MATLAB, which solved the algebraic Lyapunov equation automatically.

The Simulink block diagram for the implemented adaptive controller is shown in Figure 2. An important note is that because controller gains generally operate on linearized systems to yield system inputs about a linearization point, x and x_{ref} in the previously discussed theory correspond to state *errors* with respect to the hover (linearization) point. As a result, the reference command $r(t)$ is subtracted from actual and reference model states before passing the values to the controller. Because this controller also needs to be deployable to the drone, all operators (such as the integrator block) needed to be in discrete-time. The controller gains K_x and K_r were initialized such that $K_x(t=0) = -K_0^T$ and $K_r(t=0) = 0^{nxm}$ to ensure that $u(t=0) = -K_0 x(t=0)$. Therefore, if the drone experiences no added mass, the controller gains will experience little adaptation and $u(t) \approx -K_0 x(t)$ for all time t.

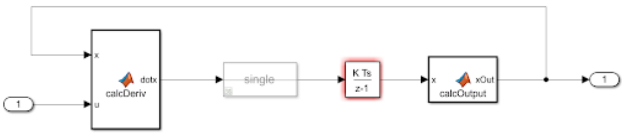

Fig. 3. Simulink subsystem of the discretized drone dynamics (derived from the "quadrotor_dynamics" s-function from Parrot). The first MATLAB function calculates derivatives which are then passed to an integration block. The states undergo a coordinate transformation before eing outputted to the controller.

B. Reference Model in Discrete Time

This MRAC scheme required that the drone dynamics (without added mass) be simulated in parallel with the measurement of the drone's actual dynamics in order to adapt the controller gains. In developing this reference model, it was necessary for the output to be reasonably accurate while also being simulated in discrete time to ensure compatibility with the drone hardware. The first option considered for the reference model was a linear system using the A_{ref} matrix generated from the linearization of the "complexDrone" model. While this system can easily be implemented in discrete time, the response too easily diverges if the drone's state becomes too far from the linearization point. The second option was to use the "complexDrone" nonlinear model. While the non-linear dynamics provide more accuracy far from the linearization point, the MATLAB s-function "quadrotor_dynamics" (used by the "complexDrone" model) simulates the dynamics in continuous time. While attempts were made to discretize this s-function output, the simulations became unstable.

To overcome these problems, a reference model subsystem was built using the functions in the "quadrotor_dynamics" s-function. One function calculated the system derivatives, and another function performed a

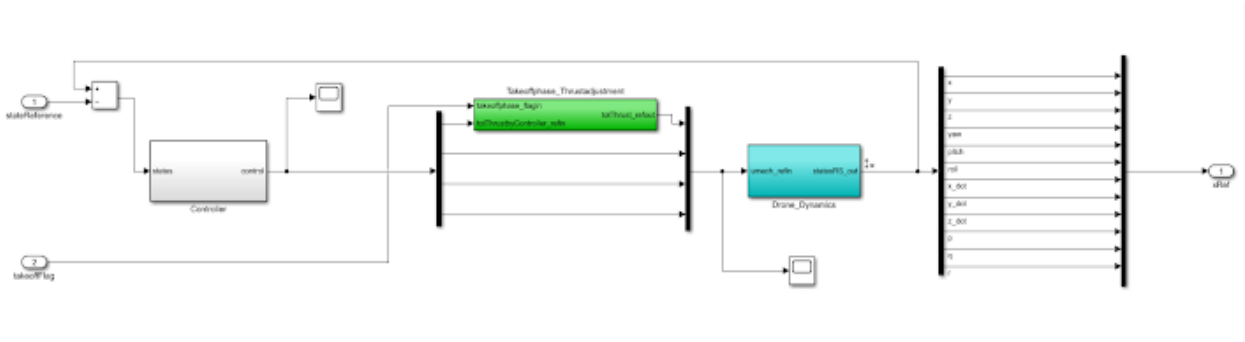

Fig. 4. Simulink model of the full reference model. It consists of a pole-placement controller which calculates an input to pass to the drone dynamics. The green block in the center of the model accounts for any takeoff time required when the drone is deployed to the Parrot drone.

coordinate transformation for the drone states. Mathematically:

$$\dot{x}' = f(x,u) \quad (7)$$
$$x = g(x') \quad (8)$$

The above equations were each implemented as MATLAB function blocks in Simulink. The output of Equation 7 was integrated using a discrete-time integrator block and then passed as the input for Equation 8. Figure 3 shows the Simulink model corresponding to the equations for the drone dynamics.

The full reference model block is shown in Figure 4. It was developed from the "complexDrone" model and includes the discretized drone dynamics from Figure 3. Figure 4 also shows an additional modification (the green subsystem block) from the "complexDrone" model to account for takeoff. During the takeoff time, the commanded thrust force is set to be a constant value rather than determined by the controller due to the unreliability of the sensors at low altitude. In the drone hardware, the takeoff time is automatically implemented. However, if the reference model does not also account for this command, then the reference model dynamics will be inaccurate compared to the actual drone dynamics. This takeoff block was copied into the reference model to allow deployment and use onboard the drone.

IV. Simulation Results

To help with initial gain tuning and ensure stability of the drone, the adaptive controller was simulated using the "quadrotor_dynamics" function as well as the animation in the asbQuadcopter project. The continuous time simulation with the "quadrotor_dynamics" function was developed by a colleague and is not discussed in detail here.

After much gain tuning, it was found that the following gains produced a stable response with adequately fast adaption: $\Gamma_r = 0.0001 \times I^{n\times n}$, $Q = 0.01 \times I^{n\times n}$, and $\Gamma_x = 0.0001 \times I^{n\times n}$ except for the third diagonal element (corresponding to the altitude) which had a value of 0.001. The response of the adaptive controller (compared to the reference model dynamics) is shown in Figure 5. In this simulation, the actual drone mass was changed from 0.068 kg to 0.200 kg, while the reference model mass remained at 0.068 kg. It is clear that with added mass, the controller is able to stabilize the drone reasonably well and the actual response closely follows the desired closed-loop dynamics. There is notable (but slight) overshoot near the step changes in the pulse maneuver that, with further gain tuning, could potentially be eliminated.

V. Conclusions and Future Work

In this work, a model reference adaptive controller (derived by Lavretsky et al) was developed and implemented for a Parrot Mambo minidrone in order to control for (unknown) added mass or changes in mass. The theoretical basis for a model reference adaptive controller was discussed along with design and implementation considerations that follow from it. Appropriate simplifying assumptions for the MRAC theory were also detailed.

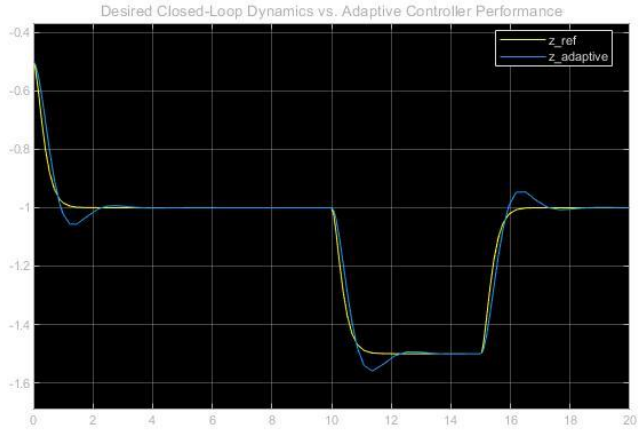

Fig. 5. A plot from simulation of the adaptive controller showing the altitude response of the drone. The yellow line represents the desired closed-loop dynamics and the blue line represents the response of the drone with the adaptive controller.

Solutions to hardware implementation issues associated with the reference model were presented. These solutions involved a discretization of the nonlinear dynamics of the "quadrotor_dynamics" and accounting for takeoff time when the controller is deployed to hardware. Finally, the controller was implemented in a simulation in which the simulated drone's mass was increased by nearly 3 times. The simulation showed a stable response of the adaptive controller, while also demonstrating areas for future improvement related to reducing overshoot.

Results from experiments involving the physical drone were not shown in this paper and will be discussed by my colleague in further detail. However, both the real-life and simulation results reveal that the transient response of the drone dynamics require improvement to eliminate overshoot, which indicates further need to refine the adaptive controller model or tune adaptation gains. In addition, this work primarily focused on adaptive control for altitude and attitude control. Future work may focus on incorporating planar motion of the drone in order to achieve stable lateral movement in addition to stability in hover.

REFERENCES

[1] Lavretsky, E., and Wise, K., 2013, *Robust and Adaptive Control With Aerospace Applications*, Springer.