

# Dynamic Modeling, Control and Simulation of an Autonomous Quadrotor

Due February 1 2013 11:59 PM

Vijay Kumar, Caitlin Powers and Koushil Sreenath  
Department of Mechanical Engineering and Applied Mechanics  
University of Pennsylvania

January 21, 2013

## 1 Introduction

In the last five years, advances in materials, electronics, sensors and batteries have fueled a growth in the development of Micro Unmanned Aerial Vehicles (MAVs) that are between 0.1-0.5 meters in length and 0.1-0.5 kilograms in mass [1]. A few groups have built and analyzed MAVs in the 10 cm range [2, 3]. One of the smallest is the Picoflyer with a 60 mm propellor diameter and a mass of 3.3 grams [4]. Platforms in the 50 cm range are more prevalent with several groups having built and flown systems of this size [5–7]. In fact, there are several commercially available RC helicopters and research grade helicopters in this size range [8]. A review of the challenges in develop autonomous micro UAVs is presented in [9].

In this project, we introduce the modeling of quad rotors used in the GRASP Multiple MAV testbed to support research on coordinated, dynamic flight of MAVs. This document is derived from the complete reference [10] which describes the approach to modeling, control and integrating off-the-shelf quad rotors.

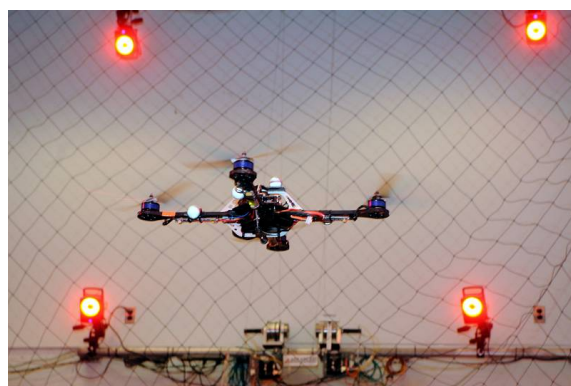
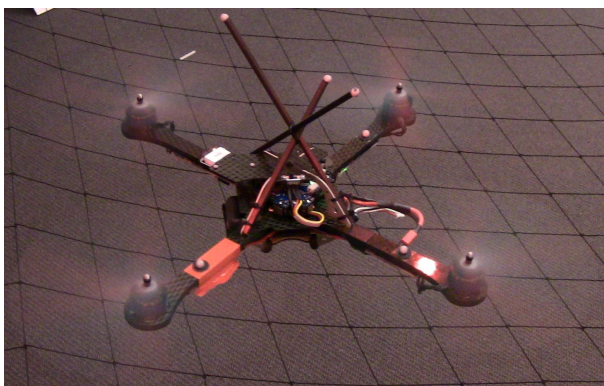


Figure 1: A quadrotor platform with VICON markers.



Figure 2: KMel's Nano+, with forward facing camera and GPS.

## 2 System Description

### 2.1 Quadrotor Platform

For this project we will be using the Nano+ platform made by KMel Robotics [11], shown in Fig. 2. The Nano+ has a tip to tip wingspan of 12 cm, a height of 6 cm, and a mass of 176 g, including a battery. We will use the Nano+ equipped with a downward facing camera but no GPS.

An onboard microcontroller allows low-level control and estimation to be done onboard the robot. An onboard IMU provides feedback of angular velocities and accelerations.

### 2.2 VICON Motion Capture System

The VICON Motion Capture System provides a state estimate for the quadrotor, which is nearly ground truth [12]. The VICON system offers three important features. First, *it is fast*: the system can be run at or below 375 Hz. Second, *it is precise*: experimental tests show that the deviations of position estimates for single static markers are on the order of 50 microns which is well beyond the requirements for flight. Lastly, *it is robust*: the VICON *Tracker* software assumes that markers in models are rigidly attached which enables the system to maintain tracking even if all but one camera are occluded. Using this software, tracking of quadrotors is rarely lost, even during extreme situations such as fast maneuvers (speeds of  $3.5 \frac{m}{s}$ , accelerations of  $15 \frac{m}{s^2}$  and angular speeds of  $1000 \frac{^\circ}{s}$ ).

### 2.3 Software and Integration

Position control and other high level commands are computed in Matlab at 100Hz and sent to the robot via Zigbee. Attitude control is performed onboard using the microcontroller.

## 3 Modeling

### 3.1 Coordinate Systems and Reference frames

The coordinate systems and free body diagram for the quadrotor are shown in Fig. 3. The inertial frame,  $\mathcal{A}$ , is defined by the triad  $\mathbf{a}_1$ ,  $\mathbf{a}_2$ , and  $\mathbf{a}_3$  with  $\mathbf{a}_3$  pointing upward. The body frame,  $\mathcal{B}$ , is attached to the center of mass of the quadrotor with  $\mathbf{b}_1$  coinciding with the preferred forward direction and  $\mathbf{b}_3$  perpendicular to the plane of the rotors

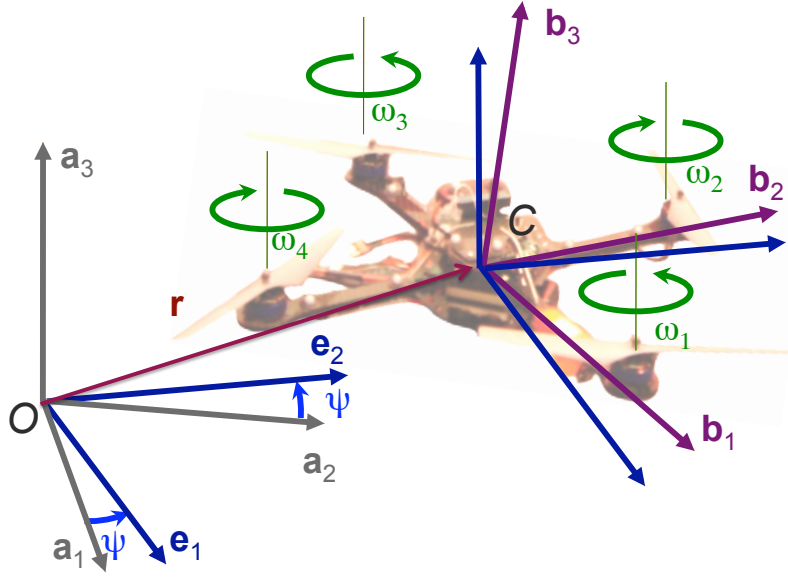


Figure 3: The body-fixed and the inertial reference frames. A pair of motors spins counter clockwise while the other pair spins clockwise. The pitches on the corresponding propellers are reversed so that the thrust is always pointing in the  $\mathbf{b}_3$  direction for all propellers. However, while the reaction moments on the frame of the robot are also in the vertical direction, the signs are such that they oppose the direction of the angular velocity of the propeller.

pointing vertically up during perfect hover (see Fig. 3). These vectors are parallel to the principal axes. The center of mass is  $C$ . Rotor 1 is a distance  $L$  away along  $\mathbf{b}_1$ , 2 is  $L$  away along  $\mathbf{b}_2$ , while 3 and 4 are similarly  $L$  away along the negative  $\mathbf{b}_1$  and  $\mathbf{b}_2$  respectively.

### 3.2 Inertial Properties

Since  $\mathbf{b}_i$  are principal axes, the inertia matrix referenced to the center of mass along the  $\mathbf{b}_i$  reference triad,  $I$ , is a diagonal matrix. In practice, the three moments of inertia can be estimated by weighting individual components of the quadrotor and building a physically accurate model in SolidWorks. The key parameters for the rigid body dynamics for the Nano+ platform are as follows:

- (a) mass:  $m \approx 0.176 \text{ kg}$ ;
- (b) the distance from the center of mass to the axis of a motor:  $L = .085 \text{ m}$ ; and
- (c) the components of the inertia dyadic using  $\mathbf{b}_i$  as the SRT:

$$[I_C]^{\mathbf{b}_i} = \begin{bmatrix} 2.5 \times 10^{-4} & 0 & 2.55 \times 10^{-6} \\ 0 & 2.32 \times 10^{-4} & 0 \\ 2.55 \times 10^{-6} & 0 & 3.73 \times 10^{-4} \end{bmatrix}.$$

The off-diagonal elements here are produced by the camera.

### 3.3 Motor Model

Each rotor has an angular speed  $\omega_i$  and produces a vertical force  $F_i$  according to

$$F_i = k_F \omega_i^2. \quad (1)$$

The rotors also produce a moment according to

$$M_i = k_M \omega_i^2. \quad (2)$$

For a given rotor, these constants can be determined through experimentation with a fixed rotor or through system identification techniques with a full quadrotor platform.

Data obtain from system identification experiments suggest that the rotor speed is related to the commanded speed by a first-order differential equation

$$\dot{\omega}_i = k_m(\omega_i^{\text{des}} - \omega_i).$$

This motor gain,  $k_m$ , is found to be about  $20 \text{ s}^{-1}$  by matching the performance of the simulation to the real system. The desired angular velocities,  $\omega_i^{\text{des}}$ , are limited to minimum and maximum values determined through experimentation.

However, as a first approximation, we can assume the motor controllers to be perfect and the time constant  $k_m$  associated with the motor response to be arbitrarily small. In other words, we can assume the actual motor velocities  $\omega_i$  are equal to the commanded motor velocities,  $\omega_i^{\text{des}}$ .

### 3.4 Rigid Body Dynamics

**Kinematics** We will use  $Z - X - Y$  Euler angles to model the rotation of the quadrotor in the world frame. To get from  $\mathcal{A}$  to  $\mathcal{B}$ , we first rotate about  $\mathbf{a}_3$  through the yaw angle,  $\psi$ , to get the triad  $\mathbf{e}_i$ . A rotation about the  $\mathbf{e}_1$  through the roll angle,  $\phi$  gets us to the triad  $\mathbf{f}_i$  (not shown in the figure). A third pitch rotation about  $\mathbf{f}_2$  through  $\theta$  results in the body-fixed triad  $\mathbf{b}_i$ . You will need the rotation matrix for transforming components of vectors in  $\mathcal{B}$  to components of vectors in  $\mathcal{A}$ :

$${}^{\mathcal{A}}[R]_{\mathcal{B}} = \begin{bmatrix} c\psi c\theta - s\phi s\psi s\theta & -c\phi s\psi & c\psi s\theta + c\theta s\phi s\psi \\ c\theta s\psi + c\psi s\phi s\theta & c\phi c\psi & s\psi s\theta - c\psi c\theta s\phi \\ -c\phi s\theta & s\phi & c\phi c\theta \end{bmatrix}. \quad (3)$$

We will denote the components of angular velocity of the robot in the body frame by  $p$ ,  $q$ , and  $r$ :

$${}^{\mathcal{A}}\omega_{\mathcal{B}} = p\mathbf{b}_1 + q\mathbf{b}_2 + r\mathbf{b}_3.$$

These values are related to the derivatives of the roll, pitch, and yaw angles according to

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} c\theta & 0 & -c\phi s\theta \\ 0 & 1 & s\phi \\ s\theta & 0 & c\phi c\theta \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \quad (4)$$

**Newton's Equations of Motion** Let  $\mathbf{r}$  denote the position vector of  $C$  in  $\mathcal{A}$ . The forces on the system are gravity, in the  $-\mathbf{a}_3$  direction, and the forces from each of the rotors,  $F_i$ , in the  $\mathbf{b}_3$  direction. The equations governing the acceleration of the center of mass are

$$m\ddot{\mathbf{r}} = \begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix} + R \begin{bmatrix} 0 \\ 0 \\ F_1 + F_2 + F_3 + F_4 \end{bmatrix}. \quad (5)$$

We will define our first input  $u_1$  to be

$$u_1 = \sum_{i=1}^4 F_i.$$

**Euler's Equations of Motion** In addition to forces, each rotor produces a moment perpendicular to the plane of rotation of the blade,  $M_i$ . Rotors 1 and 3 rotate in the  $-\mathbf{b}_3$  direction while 2 and 4 rotate in the  $+\mathbf{b}_3$  direction. Since the moment produced on the quadrotor is opposite to the direction of rotation of the blades,  $M_1$  and  $M_3$  act in the  $\mathbf{b}_3$  direction while  $M_2$  and  $M_4$  act in the  $-\mathbf{b}_3$  direction.  $L$  is the distance from the axis of rotation of the rotors to the center of mass of the quadrotor.

The angular acceleration determined by the Euler equations is

$$I \begin{bmatrix} \ddot{p} \\ \ddot{q} \\ \ddot{r} \end{bmatrix} = \begin{bmatrix} L(F_2 - F_4) \\ L(F_3 - F_1) \\ M_1 - M_2 + M_3 - M_4 \end{bmatrix} - \begin{bmatrix} p \\ q \\ r \end{bmatrix} \times I \begin{bmatrix} p \\ q \\ r \end{bmatrix}. \quad (6)$$

We can rewrite this as:

$$I \begin{bmatrix} \ddot{p} \\ \ddot{q} \\ \ddot{r} \end{bmatrix} = \begin{bmatrix} 0 & L & 0 & -L \\ -L & 0 & L & 0 \\ \gamma & -\gamma & \gamma & -\gamma \end{bmatrix} \begin{bmatrix} F_1 \\ F_2 \\ F_3 \\ F_4 \end{bmatrix} - \begin{bmatrix} p \\ q \\ r \end{bmatrix} \times I \begin{bmatrix} p \\ q \\ r \end{bmatrix}. \quad (7)$$

where  $\gamma = \frac{k_M}{k_F}$  is the relationship between lift and drag given by Equations (1-2). Accordingly, we will define our second set of inputs to be the vector of moments  $\mathbf{u}_2$  given by:

$$\mathbf{u}_2 = \begin{bmatrix} 0 & L & 0 & -L \\ -L & 0 & L & 0 \\ \gamma & -\gamma & \gamma & -\gamma \end{bmatrix} \begin{bmatrix} F_1 \\ F_2 \\ F_3 \\ F_4 \end{bmatrix}.$$

## 4 Robot Controllers

### 4.1 The Nominal State

Our controllers are derived by linearizing the equations of motion and motor models (5-2) at an operating point that corresponds to the nominal hover state,  $\mathbf{r} = \mathbf{r}_0$ ,  $\theta = \phi = 0$ ,  $\psi = \psi_0$ ,  $\dot{\mathbf{r}} = 0$ , and  $\dot{\phi} = \dot{\theta} = \dot{\psi} = 0$ , where the roll and pitch angles are small ( $c\phi \approx 1$ ,  $c\theta \approx 1$ ,  $s\phi \approx \phi$ , and  $s\theta \approx \theta$ ). At this state the lift from the propellers is given by:

$$F_{i,0} = \frac{mg}{4},$$

The nominal values for the inputs at hover are  $u_{1,0} = mg$ ,  $\mathbf{u}_{2,0} = \mathbf{0}$ .

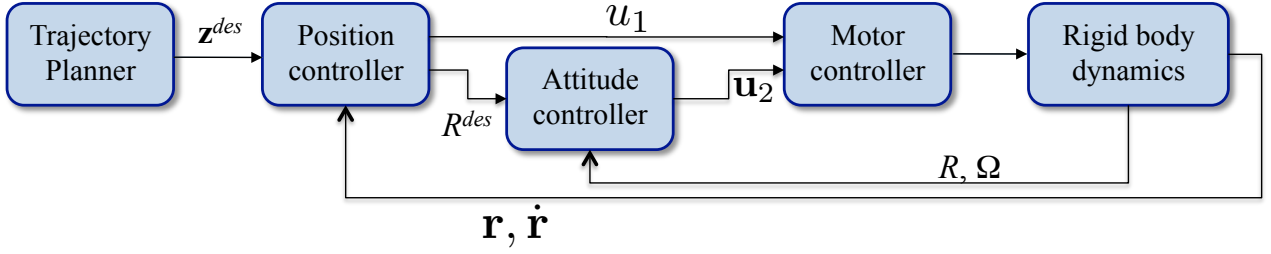


Figure 4: The position and attitude control loops.

Linearizing (5), we get:

$$\begin{aligned}\ddot{r}_1 &= g(\Delta\theta \cos \psi_0 + \Delta\phi \sin \psi_0) \\ \ddot{r}_2 &= g(\Delta\theta \sin \psi_0 - \Delta\phi \cos \psi_0) \\ \ddot{r}_3 &= \frac{1}{m}u_1 - g\end{aligned}\tag{8}$$

Linearizing (6), we get:

$$\begin{bmatrix} \ddot{p} \\ \ddot{q} \\ \ddot{r} \end{bmatrix} = I^{-1} \begin{bmatrix} 0 & L & 0 & -L \\ -L & 0 & L & 0 \\ \gamma & -\gamma & \gamma & -\gamma \end{bmatrix} \begin{bmatrix} F_1 \\ F_2 \\ F_3 \\ F_4 \end{bmatrix}\tag{9}$$

If we assume the rotor craft is symmetric so  $I_{xx} = I_{yy}$ , we get:

$$\begin{aligned}\dot{p} &= \frac{u_{2,x}}{I_{xx}} = \frac{L}{I_{xx}}(F_2 - F_4) \\ \dot{q} &= \frac{u_{2,y}}{I_{yy}} = \frac{L}{I_{yy}}(F_3 - F_1), \\ \dot{r} &= \frac{u_{2,z}}{I_{zz}} = \frac{\gamma}{I_{zz}}(F_1 - F_2 + F_3 - F_4).\end{aligned}$$

In other words, the equations of motion are decoupled in terms of angular accelerations. Each component of angular acceleration depends only on the appropriate component of  $\mathbf{u}_2$ .

## 4.2 Position and Attitude Control

The control problem is to determine the four inputs,  $\{u_1, \mathbf{u}_2\}$  required to hover or to follow a desired trajectory,  $\mathbf{z}^{des}$ . As shown in Figure 4, we will use errors in the robot's position to drive a position controller from (8) which directly determines  $u_1$ . The model in (8) also allows us to derive a desired orientation. The attitude controller for this orientation is derived from the model in (9).

The transformation of the inputs into  $\{u_1, \mathbf{u}_2\}$  is straightforward and is described in [10]. The inner attitude control loop uses onboard accelerometers and gyros to control the roll, pitch, and yaw and runs at approximately 1 kHz [5], while the outer position control loop uses estimates of position and velocity of the center of mass to control the trajectory in three dimensions at 100-200 Hz.

**Attitude Control** We now present a proportional plus derivative (PD) attitude controller to track a trajectory in  $SO(3)$  specified in terms of a desired roll, pitch and yaw angle. Since our development of the controller will be based on linearized equations of motion, the attitude must be close to the nominal hover state where the roll and pitch angles are small.

Near the nominal hover state the proportional plus derivative control laws take the form:

$$\mathbf{u}_2 = \begin{bmatrix} k_{p,\phi}(\phi^{\text{des}} - \phi) + k_{d,\phi}(\dot{\phi}^{\text{des}} - \dot{\phi}) \\ k_{p,\theta}(\theta^{\text{des}} - \theta) + k_{d,\theta}(\dot{\theta}^{\text{des}} - \dot{\theta}) \\ k_{p,\psi}(\psi^{\text{des}} - \psi) + k_{d,\psi}(\dot{\psi}^{\text{des}} - \dot{\psi}) \end{bmatrix} \quad (10)$$

**Position Control** In the next subsection, we will present two position control methods that use the roll and pitch angles as inputs to drive the position of the quadrotor. In both methods, the position control algorithm will determine the desired roll and pitch angles,  $\theta^{\text{des}}$  and  $\phi^{\text{des}}$ , which can be used to compute the desired speeds from (10). The first, a hover controller, is used for station-keeping or maintaining the position at a desired position vector,  $\mathbf{r}_0$ . The second tracks a specified trajectory,  $\mathbf{r}_T(t)$ , in three dimensions. In both cases, the desired yaw angle, is specified independently. It can either be a constant,  $\psi_0$  or a time varying quantity,  $\psi_T(t)$ . We will assume that the desired trajectory:

$$\mathbf{z}^{\text{des}} = \begin{bmatrix} \mathbf{r}_T(t) \\ \psi_T(t) \end{bmatrix},$$

will be provided by a trajectory planner as an input to specify the trajectory of the position vector and the yaw angle we are trying to track.

### 4.3 Hover Controller

For hovering,  $\mathbf{r}_T(t) = \mathbf{r}_0$  and  $\psi_T(t) = \psi_0$ . The command accelerations,  $\ddot{r}_i^{\text{des}}$ , are calculated from a proportional plus derivative (PD) controller. Define the position error in terms of components using the standard reference triad  $\mathbf{a}_i$  by:

$$e_i = (r_{i,T} - r_i).$$

In order to guarantee that this error goes exponentially to zero, we require

$$(\ddot{r}_{i,T} - \ddot{r}_i^{\text{des}}) + k_{d,i}(\dot{r}_{i,T} - \dot{r}_i) + k_{p,i}(r_{i,T} - r_i) = 0, \quad (11)$$

where  $\dot{r}_{i,T} = \ddot{r}_{i,T} = 0$  for hover.

From (8) we can obtain the relationship between the desired accelerations and roll and pitch angles. Given that  $\Delta\theta = \theta - \theta_0 = \theta$  and  $\Delta\phi = \phi - \phi_0 = \phi$ , we can write

$$\begin{aligned} \ddot{r}_1^{\text{des}} &= g(\theta^{\text{des}} \cos \psi_T + \phi^{\text{des}} \sin \psi_T) \\ \ddot{r}_2^{\text{des}} &= g(\theta^{\text{des}} \sin \psi_T - \phi^{\text{des}} \cos \psi_T) \\ \ddot{r}_3^{\text{des}} &= \frac{1}{m}u_1 - g. \end{aligned}$$

For hover, the last equation yields:

$$u_1 = mg + m\ddot{r}_3^{\text{des}} = mg - m(k_{d,3}\dot{r}_3 + k_{p,3}(r_3 - r_{3,0})) \quad (13)$$

The other two equations can be used to compute the desired roll and pitch angles for the attitude controller:

$$\phi^{\text{des}} = \frac{1}{g}(\ddot{r}_1^{\text{des}} \sin \psi_T - \ddot{r}_2^{\text{des}} \cos \psi_T) \quad (14a)$$

$$\theta^{\text{des}} = \frac{1}{g}(\ddot{r}_1^{\text{des}} \cos \psi_T + \ddot{r}_2^{\text{des}} \sin \psi_T) \quad (14b)$$

The desired roll and pitch velocities are taken to be zero.

$$p^{\text{des}} = 0 \quad (15a)$$

$$q^{\text{des}} = 0 \quad (15b)$$

Since the yaw,  $\psi_T(t)$  is prescribed independently by the trajectory generator, we get:

$$\psi^{\text{des}} = \psi_T(t) \quad (16a)$$

$$\dot{\psi}^{\text{des}} = \dot{\psi}_T(t) \quad (16b)$$

These equations provide the setpoints for the attitude controller in (10).

Note that the attitude controller must run an order of magnitude faster than the position control loop in order for this to work. In practice as discussed in [10], the position controller runs at 100 Hz, while the inner attitude control loop runs at 1 kHz.

#### 4.4 3D Trajectory Control

The 3-D Trajectory Controller is used to follow three-dimensional trajectories with modest accelerations so the near-hover assumptions hold. To derive this controller we follow the same steps as in (11) except that  $\dot{r}_{i,T}$  and  $\ddot{r}_{i,T}$  are no longer zero but are obtained from the specification of the trajectory. If the near-hover assumption holds and the dynamics are linear with no saturation on the inputs, a controller that generates the desired acceleration  $\ddot{r}_i^{\text{des}}$  according to (11) is guaranteed to drive the error exponentially to zero. However, it is possible that the commanded trajectory has twists and turns that are too hard to follow perfectly because of errors in the model or limitations on the input thrusts. We suggest a modification that allows the robot to follow the path even if it may not be able to follow the time-parameterized trajectory.

Let  $\mathbf{r}_T$  denote the closest point on the desired trajectory to the the current position  $\mathbf{r}$ , and let the desired velocity and acceleration obtained by differentiating the specified trajectory be given by  $\dot{\mathbf{r}}_T$  and  $\ddot{\mathbf{r}}_T$  respectively. Let the unit tangent vector of the trajectory (unit vector along  $\dot{\mathbf{r}}_T$ ) be  $\hat{\mathbf{t}}$ . The unit normal to the trajectory,  $\hat{\mathbf{n}}$ , is derived by differentiating the tangent vector with respect to time or arc length, and finally the unit binormal vector is the cross product  $\hat{\mathbf{b}} = \hat{\mathbf{t}} \times \hat{\mathbf{n}}$ . We define the position and velocity errors according to the following equations:

$$\mathbf{e}_p = ((\mathbf{r}_T - \mathbf{r}) \cdot \hat{\mathbf{n}})\hat{\mathbf{n}} + ((\mathbf{r}_T - \mathbf{r}) \cdot \hat{\mathbf{b}})\hat{\mathbf{b}}$$

and

$$\mathbf{e}_v = \dot{\mathbf{r}}_T - \dot{\mathbf{r}}.$$

Note that here we ignore position error in the tangential direction and only considering position error in the plane that is normal to the curve at the closest point. Once again we calculate the commanded acceleration,  $\ddot{r}_i^{\text{des}}$ , from the PD feedback as shown in (11). In vector notation, this is:

$$(\ddot{\mathbf{r}}_T - \ddot{\mathbf{r}}^{\text{des}}) + \mathbf{k}_d \mathbf{e}_v + \mathbf{k}_p \mathbf{e}_p = 0, \quad (17)$$

Finally we use (14 - 16) to compute the desired roll and pitch angles. Again we refer the readers to [10] for more details including experimental results obtained from these controllers.



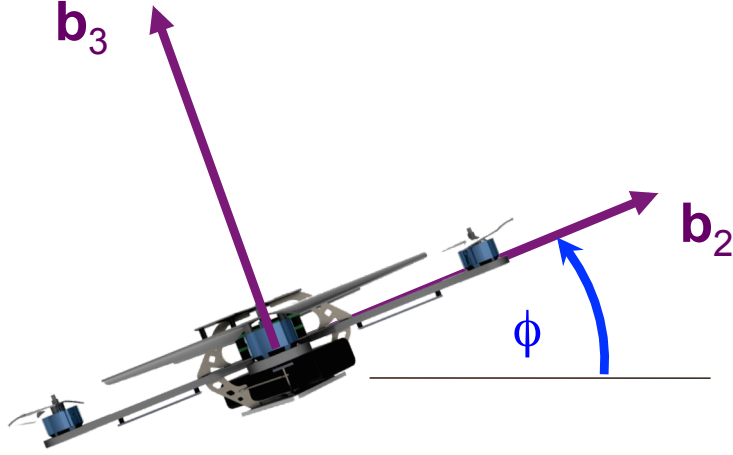


Figure 5: A planar model with zero roll and yaw in the  $y - z$  plane. .

## 5 Controller Design

The controller design in the previous sections was based on the decoupling of the position and attitude control subsystems as motivated in Fig. 4. This is only valid when the attitude control loop is much faster than the position control loop permitting us to use, what is called in the control theory literature (see [13]), a *backstepping* approach to controller design. This is roughly the approach that justifies the development in Sections 4.3-4.4.

It is also possible to consider the coupled system with positions and orientations concurrently. To see this, we consider a planar model of the quadrotor which can be obtained by restricting the motion of the quadrotor to the  $y - z$  plane with zero roll ( $\theta$ ) and yaw ( $\psi$ ). In such a case, (5-6) reduce to:

$$m \begin{bmatrix} \ddot{y} \\ \ddot{z} \\ \ddot{\phi} \end{bmatrix} = \begin{bmatrix} 0 \\ -mg \\ 0 \end{bmatrix} + \begin{bmatrix} -\frac{1}{m} \sin \phi & 0 \\ \frac{1}{m} \cos \phi & 0 \\ 0 & \frac{1}{I_{xx}} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \quad (18)$$

We can write this in state space notation in the usual form:

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}) \quad (19)$$

where

$$\mathbf{x} = [y, z, \phi, \dot{y}, \dot{z}, \dot{\phi}]^T, \quad \mathbf{u} = [u_1, u_2]^T.$$

## 5.1 Control of the linearized system

As before we linearize the system about the hover state. The hover state,  $(\mathbf{x}_0, \mathbf{u}_0)$ , can be written as (*why?*):

$$\mathbf{x}_0 = \begin{bmatrix} y_0 \\ z_0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad \mathbf{u}_0 = \begin{bmatrix} mg \\ 0 \end{bmatrix}.$$

We now linearize the system about this point to get:

$$\begin{aligned} \dot{\mathbf{x}} &= \frac{\partial}{\partial \mathbf{x}} \mathbf{f}(\mathbf{x}, \mathbf{u})|_{(\mathbf{x}_0, \mathbf{u}_0)} (\mathbf{x} - \mathbf{x}_0) + \frac{\partial}{\partial \mathbf{u}} \mathbf{f}(\mathbf{x}, \mathbf{u})|_{(\mathbf{x}_0, \mathbf{u}_0)} (\mathbf{u} - \mathbf{u}_0) \\ &= \mathbf{A}(\mathbf{x} - \mathbf{x}_0) + \mathbf{B}(\mathbf{u} - \mathbf{u}_0) \end{aligned}$$

where

$$\mathbf{A} = \dots, \mathbf{B} = \dots$$

The controller takes the form of a linear feedback law:

$$(\mathbf{u} - \mathbf{u}_0) = \mathbf{K}(\mathbf{x} - \mathbf{x}_0) \quad (20)$$

where the  $2 \times 6$  gain matrix,  $\mathbf{K}$ , must be synthesized to optimize the performance of the system. For a linear system, this is generally posed as a *linear-quadratic feedback regulator* (LQR) design problem where  $\mathbf{K}$  is chosen to minimize the cost functional:

$$\int_0^\infty (\mathbf{x} - \mathbf{x}_0)^T \mathbf{Q} (\mathbf{x} - \mathbf{x}_0) + (\mathbf{u} - \mathbf{u}_0)^T \mathbf{R} (\mathbf{u} - \mathbf{u}_0) dt \quad (21)$$

which weights excursions from the error position and magnitudes of the input effort over the trajectory via suitably chosen matrices  $\mathbf{Q}$  and  $\mathbf{R}$ . See matlab function `lqr` for the continuous LQR design problem and its close relative `dlqr` for the discrete controller design problem.

## 5.2 Control of the nonlinear system

In Equation (18), the nonlinearity arises from the sine and cosine functions of the pitch  $\phi$ . Returning to the discussion in Section 4.3, is it possible to design a *nonlinear* controller that can guarantee performance over a wider range of conditions including large excursions from hover? The answer is yes and the approach presented here to address this is based closely on [14, 15].

We start with (11) and write:

$$\begin{bmatrix} \ddot{y}^{\text{des}} \\ \ddot{z}^{\text{des}} \end{bmatrix} = \begin{bmatrix} \ddot{y}_T \\ \ddot{z}_T \end{bmatrix} + \mathbf{k}_d \begin{bmatrix} \dot{y}_T - \dot{y} \\ \dot{z}_T - \dot{z} \end{bmatrix} + \mathbf{k}_p \begin{bmatrix} y_T - y \\ z_T - z \end{bmatrix}. \quad (22)$$

Because it is impossible to directly produce accelerations along the  $\mathbf{b}_2$  direction, we do the next best thing by projecting the desired acceleration vector,  $\mathbf{k}$ , along the  $\mathbf{b}_3$  direction and compute the inputs required to produce this acceleration:

$$u_1 = mg + [-\sin \phi \quad \cos \phi] \left[ \begin{bmatrix} \ddot{y}_T \\ \ddot{z}_T \end{bmatrix} + \mathbf{k}_d \begin{bmatrix} \dot{y}_T - \dot{y} \\ \dot{z}_T - \dot{z} \end{bmatrix} + \mathbf{k}_p \begin{bmatrix} y_T - y \\ z_T - z \end{bmatrix} \right]. \quad (23)$$

We then compute the input that is required to align the  $\mathbf{b}_3$  vector along the desired acceleration. This is derived from the error computed as follows:

$$e_{rot} = \frac{\mathbf{b}_3 \times \mathbf{k}}{\|\mathbf{k}\|} \cdot \mathbf{b}_1, \quad (24)$$

where  $\mathbf{k}$  is the desired acceleration vector in (22). This is effectively the sine of the angle between the desired acceleration vector and the vector along which the robot can produce an acceleration. Thus the second control input must drive this error to zero. Accordingly we write:

$$u_2 = I_{xx}(-k_r e_{rot} - k_\omega \dot{e}_{rot}) \quad (25)$$

Equations (23, 25) with any positive definite  $\mathbf{k}_p$  and  $\mathbf{k}_d$  and positive  $k_r$  and  $k_\omega$  guarantee global convergence. See [14] for a discussion of convergence for this nonlinear controller.

## 6 LQR Design for the linearized model

LQR is a design technique from optimal control theory [16] to design controllers for linear systems to minimize a quadratic cost function subject to the dynamic constraint  $\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u}$ , resulting in a linear state-feedback controller.

**Theorem 1** *If the linear system  $\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u}$  is controllable and if  $\mathbf{Q}$  is a positive semi-definite matrix and  $\mathbf{R}$  is positive definite matrix, then there exists a solution to the optimization problem*

$$\min_{\mathbf{u}(t)} \int_0^T [\mathbf{x}^T(t)\mathbf{Q}\mathbf{x}(t) + \mathbf{u}^T(t)\mathbf{R}\mathbf{u}(t)] dt, \quad (26)$$

and the solution is given by

- $\mathbf{u}^*(t) = -\mathbf{K}(t)\mathbf{x}(t)$ , where
- $\mathbf{K}(t) = \mathbf{R}^{-1}\mathbf{B}^T\mathbf{S}(t)$ , and  $\mathbf{S}(t)$  is the solution of the Riccati equation:
- $-\dot{\mathbf{S}}(t) = \mathbf{Q} + \mathbf{S}(t)\mathbf{A} + \mathbf{A}^T\mathbf{S}(t) - \mathbf{S}(t)\mathbf{B}\mathbf{R}^{-1}\mathbf{B}^T\mathbf{S}(t)$ .

The solution of the infinite-horizon ( $T \rightarrow \infty$ ) optimization problem,

$$\min_{\mathbf{u}(t)} \int_0^\infty [\mathbf{x}^T(t)\mathbf{Q}\mathbf{x}(t) + \mathbf{u}^T(t)\mathbf{R}\mathbf{u}(t)] dt, \quad (27)$$

also exists and is given by

- $\mathbf{u}^*(t) = -\mathbf{K}\mathbf{x}(t)$ , where
- $\mathbf{K} = \mathbf{R}^{-1}\mathbf{B}^T\mathbf{S}$ , and  $\mathbf{S}$  is the solution of the Algebraic Riccati Equation (ARE):
- $\mathbf{Q} + \mathbf{S}\mathbf{A} + \mathbf{A}^T\mathbf{S} - \mathbf{S}\mathbf{B}\mathbf{R}^{-1}\mathbf{B}^T\mathbf{S} = 0$ .
- *Matlab Command:*  $[K \ S] = \text{lqr}(A, B, Q, R)$ .

**Remark 1** *There is a trade-off between speed of response and the magnitude of control signals. Penalizing the controls more heavily than the states ( $R$  large in comparison to  $Q$ ), tends to result in slower response, while penalizing the states more heavily than the controls ( $Q$  large in comparison to  $R$ ) tends to result in faster response but at the expense of larger control signals.*

**Remark 2** *A few choices of  $Q$  and  $R$  for an example system  $\dot{x} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} x + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u$  are below:*

- $Q = 0.1 \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ ,  $R = 1$ . (This would result in slow convergence but with small control signals.)
- $Q = 100 \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ ,  $R = 1$ . (This results in fast convergence but at the expense large control signals)
- $Q = \begin{bmatrix} \frac{1}{1^2} & 0 \\ 0 & \frac{1}{0.2^2} \end{bmatrix}$ ,  $R = \frac{1}{1^2}$ . (This requests  $\max|x_1| \leq 1$ ,  $\max|x_2| \leq 0.2$  and  $\max|u| \leq 1$ , which is respected by the lqr controller)

**Remark 3** *To stabilize to a final goal state, (27) is used, while (26) is used for stabilizing to a time-dependent state along the desired trajectory rather than to a single goal state.*

## 7 Project Work

### 7.1 Controller

You will simulate the quadrotor dynamics and control using the matlab simulator posted on the website. The simulator relies on the numerical solver `ode45`. You will need to write a file `controller.m` which has inputs and outputs specified in the code. In order to run the simulation, fill in the appropriate variables (`trajhandle`, `controlhandle`, and `stop`) and type `runsim` in the matlab command prompt.

### 7.2 Trajectories

You will need to command the quadrotor through two sample trajectories. In order to specify the trajectory in the simulator, you will need to change the variable `trajhandle`. `trajhandle` is the name of an m file that returns a desired state as a function of time. The first output from `trajhandle` must be a  $3 \times 1$  vector representing the desired position. This file may return additional outputs if you wish, such as desired velocity, acceleration, or any other useful thing for your controller. The names of the files you will need to create and descriptions of the paths they should produce are below. Note that you will need to decide on a velocity profile to turn the path into a trajectory.

`circle.m`: A helix in the xy plane of radius 5m centered about the point  $(0, 0, 0)$  with the quadrotor starting at the point  $(0, 0, 0)$ . The z coordinate should start at 0 and end at 2.5.

`diamond.m`: A “diamond helix” with corners at  $(0, 0, 0)$ ,  $(0, 2 \sin 45, 2 \cos 45)$ ,  $(0, 0, 4 \cos 45)$ , and  $(0, -2 \sin 45, 2 \cos 45)$  when projected into the yz plane, and an x coordinate starting at 0 and ending at 1. The quadrotor should start at  $(0, 0, 0)$  and end at  $(1, 0, 0)$ .

These trajectories only dictate the position of the center of the quadrotor. You may experiment with different rotations to find the one that best suits the trajectory. For instance, there are several ways a quadrotor can traverse a circle while rotating.

### 7.3 Submission

When you are finished you may submit your code via turnin. The project name for this assignment is titled “proj1phase2” so the command to submit should be

```
turnin -c meam620 -p proj1phase2 -v *
```

Your turnin submission should contain:

1. A README file detailing anything we should be aware of.
2. Files `controller.m`, `diamond.m`, `circle.m`, as well as any other Matlab files you need to run your code.

Shortly after submitting you should receive an e-mail from `meam620@seas.upenn.edu` stating that your submission has been received. You can check on the status of your submission at <https://alliance.seas.upenn.edu/~meam620/monitor/>. Once the automated tests finish running, you should receive another e-mail containing your test results. This report will tell you how long you took for each trajectory and what the total error was, or if it failed with an error. Your code will be given at most 10 minutes to complete all the tests. There is no limit on the number of times you can submit your code.

Please do not attempt to determine what the automated tests are or otherwise try to game the automated test system. Any attempt to do so will be considered cheating, resulting in a 0 on this assignment and possible disciplinary action. You will be graded on successful completion of the code and how quickly and accurately your quadrotor follows the circle and diamond paths and one other trajectory which will not be released. The grading will depend on the LAST results you achieved so make sure to submit your best code before the deadline.

### References

- [1] D. Pines and F. Bohorquez, “Challenges facing future micro air vehicle development,” *AIAA Journal of Aircraft*, vol. 43, no. 2, pp. 290–305, 2006.
- [2] I. Kroo, F. Prinz, M. Shantz, P. Kunz, G. Fay, S. Cheng, T. Fabian, and C. Partridge, “The mesicopter: A miniature rotorcraft concept, phase ii interim report,” 2000.
- [3] B. Hein and I. Chopra, “Hover performance of a micro air vehicle: Rotor at low reynolds number,” *Journal of the American Helicopter Society*, vol. 52, no. 3, pp. 254–262, July 2007.
- [4] “Proxflyer,” <http://www.proxflyer.com>.
- [5] D. Gurdan, J. Stumpf, M. Achtelik, K. Doth, G. Hirzinger, and D. Rus, “Energy-efficient autonomous four-rotor flying robot controlled at 1 khz,” in *Proc. of the IEEE Int. Conf. on Robotics and Automation*, Roma, Italy, Apr. 2007.
- [6] S. Bouabdallah, “Design and control of quadrotors with application to autonomous flying,” Ph.D. dissertation, Ecole Polytechnique Federale de Lausanne, 2007.
- [7] G. Hoffmann, S. Waslander, and C. Tomlin, “Quadrotor helicopter trajectory tracking control,” in *AIAA Guidance, Navigation and Control Conference and Exhibit*, Honolulu, Hawaii, Apr. 2008.
- [8] “Ascending technologies,” <http://www.asctec.de>.

- [9] V. Kumar and N. Michael, "Opportunities and challenges with autonomous micro aerial vehicles," in *Int. Symposium of Robotics Research*, Flagstaff, AZ, Aug. 2011.
- [10] N. Michael, D. Mellinger, Q. Lindsey, and V. Kumar, "The grasp multiple micro uav testbed," *IEEE Robotics and Automation Magazine*, 2010.
- [11] "Kmel robotics," <http://kmelrobotics.com/>.
- [12] "Vicon MX Systems," <http://www.vicon.com/products/viconmx.html>.
- [13] H. Khalil, *Nonlinear Systems*. Prentice Hall, 1996.
- [14] T. Lee, M. Leok, and N. McClamroch, "Geometric tracking control of a quadrotor uav on  $SE(3)$ ," in *Proc. of the IEEE Conf. on Decision and Control*, 2010.
- [15] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *Proc. of the IEEE Int. Conf. on Robotics and Automation*, Shanghai, China, May 2011.
- [16] D. E. Kirk, *Optimal Control Theory*. Dover Publications, 2004.