In4073 QR Controller Theory (2011-2012)

Arjan J.C. van Gemund

Embedded Software Lab
Software Technology Department
Faculty of Electrical Engineering, Mathematics, and Computer Science
Delft University of Technology

February 2012

1 Introduction

This document describes how to develop an attitude controller for the Delft AeroVinci "quad-pilot 2 F.3" Quad-Rotor Aerial Vehicle (QR), which is used in the MSc lab course in4073 Embedded Real-time Systems [1]. The document assumes introductory knowledge on the assignment [2], control theory, and filtering as lectured within in4073.

2 Control Principles

In order to allow easy pilot handling the QR should be attitude-controlled with respect to roll and pitch, rather than rate-controlled, whereas yaw should be rate-controlled. That is, the joystick (and keyboard) roll and pitch setpoints have the semantics of *angles* φ and θ, respectively, while the yaw setpoint has the semantics of *rate* r (twist the JS and the QR will yaw, neutral JS and the QR will stabilize at the current angle ψ). Apart from pilot convenience, there is little point in yaw attitude control as there is no sensor to estimate ψ anyway, whereas for φ and θ feedback we can use the Y and X axis accelerometers, respectively [1].

2.1 Yaw control

As may have become clear during the in4073 control theory lecture, PID controllers can be used for each of the 3 dimensions. The yaw controller implementation is straightforward, and can be implemented by the simple *rate* P controller

```
N_s = P_r * (yaw_s - sr)
```

where `N_s` is the torque setpoint for the QR engines, and `yaw_s` is the twist handle input (rate setpoint). Since, the QR integrates torque `N_s` into rate `sr` there is no need for an additional I term in the above P controller. On the other hand, an additional D term might be needed to compensate for phase lags due to latency, filtering, etc., but should initially be avoided, and is therefore not included in the above controller.

2.2 Roll/Pitch control

The roll and pitch controllers are slightly more complicated and will be treated in more depth. As they are identical, we will only discuss the roll controller. Note that mapping an angle setpoint to a torque setpoint (i.e., two integrations) using just one controller is impossible due to the associated 180 degrees phase lag, unless derivative control would be added. Consequently, in (roll or pitch) attitude control, there are *two* nested P(ID) controllers involved: an *attitude* controller that maps the joystick angle setpoint `roll_s` to a rate setpoint `p_s` for the *rate* controller, that maps the rate setpoint into the torque setpoint `K_s` for the QR engines. The cascaded controller is shown in Figure 1. Similar to the previous yaw controller we restrict ourselves to simple P control. QR' models a QR that has rate input due to the second P controller. As QR‘ represents a single integration

[1] Note that φ and θ are formally Euler angles (i.e., φ is a body rotation) while in the current paper we refer to φ as attitude relative to the *earth* frame. However, as we assume (near) hovering conditions the (fundamental) difference is immaterial.

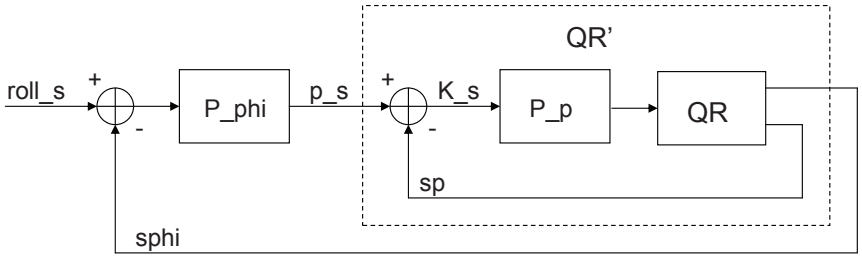

Figure 1: Cascaded P controller

only, angle control can be implemented by a single (second) P controller. The cascaded controller equations are given by

```
p_s = P_phi * (roll_s - sphi)
K_s = P_p * (p_s - sp)
```

The two cascaded P controller equations can be simply combined into one equation

```
K_s = P_p * (P_phi * (roll_s - sphi) - sp)
```

which reduces to

```
K_s = P1 * (roll_s - sphi) - P2 * sp
```

where `P1` and `P2` are positive control parameters. The above form resembles a PD controller (with `P1` and `P2` proportional and derivative control, respectively) since `sp` is the time derivative of `sphi`. Note that a real PD controller includes a derivative `sphi` term in the `D` subexpression, but the effect of the current D subexpression on feedback stability is the same, which is due to the `sp` term (which is the derivative of `sphi`).

3 Kalman Filtering

Although the above controller will work very well for ideal sensors, in practice, accelerometers and, most notably, gyroscopes exhibit drift (and noise). Consequently, after initial calibration has been performed `sphi` and `sp` will start to drift away from the actual QR state variables `phi` and `p`, respectively, due to vibration, temperature, voltage changes due to varying rotor engine RPM, battery depletion, etc. As a result, the above (cascaded P) controller will drift in terms of both attitude and rate. The attitude drift from accelerometers is typically small, and can be easily compensated with the JS as one is manually controlling the angle setpoints anyway. A rate drift from the gyros, however, is not easily compensated by a JS that controls angles and would required constant re-trimming.

A typical solution in IMU design for aerial vehicles such as QRs is to apply *Kalman filtering*, a sensor fusing technique due to Kalman (see [3] for papers on Kalman filtering). A Kalman filter dynamically adjusts the calibration of `sp` which is based on exploiting information on `sphi` and `sp`. Due to this dynamic calibration, the Kalman filter yields (on average) a perfect (i.e., drift-free) estimation of `p` which greatly improves QR control. Apart from virtually drift-free rate sensing, QR control is also improved due to the following. While gyros have non-negligible drift, they experience low noise pickup from the QR engines, which, despite the drift, make them quite suitable for fast movement control (i.e., compensating for the quick, random attitude changes during hover). Accelerometers, on the other hand, are almost drift-free, but pick up much more noise due to engine vibration. This implies that they would have to be significantly filtered at low frequencies, severely degrading their use in controlling fast QR movements. Since Kalman filtering removes gyro drift, gyros now effectively become the main source for QR control instead of the accelerometers. An extra reason why the gyros are the most important source of attitude control is that accelerometers also pick up on body accelerations next to gravity. Although in hover the lateral movements of the QR will cancel out in the long term, a short-term lateral drift of the QR will influence the acclererometer readings and introduce an error in the computed angle. In fact, this error can be disastrous. Suppose the QR pitches down which causes the QR to accelerate in forward direction. This results in a decrease in acceleration as measured by the X-axis accelerometer. Consequently, the angle returned to the controller is too small and the QR is not sufficiently compensated. This, in turn,

leads to further acceleration, i.e., an unintended feed-forward effect next to the intended feedback control. As we will show, in a Kalman scenario accelerometers are primarily used as reference for the Kalman filter, rather than directly being used in the earlier, cascaded P controller. In fact, the `sphi` signal in the controller will now primarily be based on integrating gyro rate, rather than on the accelerometer signal.

In the following, we restrict ourselves to QR flying conditions close to hover (which is the target of in4073). During hover, the average value of `sphi` corresponds to level flight. This information is exploited as we will show. A formal, mathematical treatment of Kalman filtering is beyond the scope of this course. We simply present the equations for an inertia-only Kalman filter (i.e., for hovering), and give a rationale for each equation.

Let `p` denote the QR's actual angular rate (which, of course, we cannot directly measure), according to

```
p = sp - p_b
```

where `p_b` denotes the error (b for bias) in `p` due to drift. Later on, we show how `p_b` is computed by the Kalman filter. Rather than estimating `phi` from a noisy, and/or slow `sphi`, we now use the fact that `p` is (approximately) ideal, so we integrate `p` to obtain `phi`. Note that integrating rates into angles is generally a bad idea as any offset (drift) is accumulated into a potentially unbounded error. However, in Kalman filtering this effect doesn't happen, as discussed later on. Thus the Kalman filter prediction is given by

```
phi = phi + p * P2PHI
```

where `p` is estimated by the earlier equation, and `P2PHI` is the rate/angle conversion constant, determined by the gyro and accelerometer transfer functions, as well as by the frequency with which the Kalman equations are executed (e.g., 1000 Hz sample rate). Note that in the above we assume that gyro and angle signals have the *same* polarity, i.e., a positive gyro signal corresponds to a positive angle increment. Based on the *observed* attitude `sphi` the above prediction for `phi` is adjusted according to the update

```
phi = phi - (phi - sphi) / C1
```

where the constant `C1` determines how much weight must be given to the observed angle compared to the previous prediction. Given the superiority of the (unbiased) gyro information (`p`, and hence `phi`) compared to the accelerometer (`sphi`), `C1` has a high value of, e.g., 100. Thus we put more weight to `p` (via `sp`) than to `sphi` when estimating `phi`.

In turn, the above adjustment based on the real `sphi` bears consequences for `p` as well, and the bias in `sp` is slightly adjusted according to the update

```
p_b = p_b + (phi - sphi) / C2
```

where `C2` determines how fast the bias is updated (much slower than the angle, e.g., `C2` = 1,000,000). The updated drift `p_b` is fed back to the first Kalman equation that (re)computes `p`, and all subsequent equations are re-executed, etc. The set of equations is run iteratively, which makes the Kalman filter dynamically compute the actual drift, based on the fact that *on average* `sphi` varies around a (small) *constant* (i.e., the bias in `sphi`), given that the QR hovers around zero angle (consequently, the continuous integration of `p` will stay within a narrow band around 0).

In summary, the Kalman filter estimates the real QR state variables `phi` and `p` from `sphi` and `sp`, respectively. Note that while the average estimate of `p` is exact (in the limit), `phi` can never be estimated better than `sphi`. While `p` can be estimated due to the fact that its long-term integration must be zero (average hover conditions), no such knowledge is available for `phi`. Hence, in stationary conditions `phi` will approximate `sphi`. Consequently, when `sphi` is improperly calibrated, the Kalman filter will return a non-zero `phi`. Without proper trimming this will lead to non-level flight. As mentioned earlier, however, this constant attitude bias can be easily trimmed away (either by JS or keyboard).

4 Implementation Notes

From the above it follows that the implementation of a Kalman-based attitude controller for the Aero-Vinci QR comprises two stages, i.e., Kalman filter, and cascaded P controller. In the following example pseudo code we assume both are integrated within the same (control) loop (e.g., a timer ISR).

```
ISR begin

    ...

    // Kalman for p, phi
    p = sp - p_b
    phi = phi + p * P2PHI
    phi = phi - (phi - sphi) / C1
    p_b = p_b + (phi - sphi) / C2

    // Use p, phi in P controller
    K_s = P1 * (roll_s - phi) - P2 * p

    ...

ISR end
```

Apart from the constants `P2PHI`, `C1`, `C2`, `P`, and `D`, special attention must be given to a proper dimensioning of the signal values. After calibration the sensor values range in the order of tens of units. Without proper scaling this will immediately pose numerical (integer) problems as, e.g., `P2PHI` will be less than unity for realistic update frequencies (1000 Hz). A similar argument holds with respect to the divisions by `C1`, and, in particular, `C2` [2]. Note that the 32-bit X32 architecture provides sufficient headroom for proper scaling.

Acknowledgement

We extend our appreciation to dr.ir. Christophe De Wagter of the TUD Aerospace Faculty and Aero-Vinci BV, for the many helpful discussions on QR control.

References

[1] In4073 Course Web Site. `http://www.st.ewi.tudelft.nl/~gemund/Courses/In4073/index.html`.

[2] In4073 Assignment document. `http://www.st.ewi.tudelft.nl/~gemund/Courses/In4073/Resources/assignment.pdf`.

[3] In4073 Resources. `http://www.st.ewi.tudelft.nl/~gemund/Courses/In4073/Resources/index.html`.

[2] For performance reasons integer division should be avoided other than through bit shifts.