Single-degree-of-freedom angular-velocity control for CubeSat's

Karla Sever (undergraduate), Ivan Vnučec (undergraduate), Josip Lončar (postdoctorate)
FERSat project, Faculty of Electrical Engineering and Computing, University of Zagreb

Key words: CubeSat, PID, Angular Velocity Control

ABSTRACT

Single-degree-of-freedom angular-velocity control is the beginner's entry into the full Three-degree-of-freedom orientation control. PID controller was chosen because of its widely use and simplicity. Our objective was to find transfer function of the system which was then used to find the optimal PID controller parameters.

INTRODUCTION

To develop Attitude control for CubeSat's, one must first be able to control the angular-velocity with respect to all three axes in the body-reference-system. To show proof-of-concept, the easiest approach is to firstly control the angular velocity about one axis and then go to the full three-axis control.

The testbed system consists of an 3D printed air-bearing which provides a low friction interface between two surfaces. Figure 1 depicts the spherical body carrying CubeSat placed on the spherical air-bearing. With that configuration, the system is acting like it is in a torque-free space environment.

In this report, we will summarize our steps for finding optimal PID controller parameters.

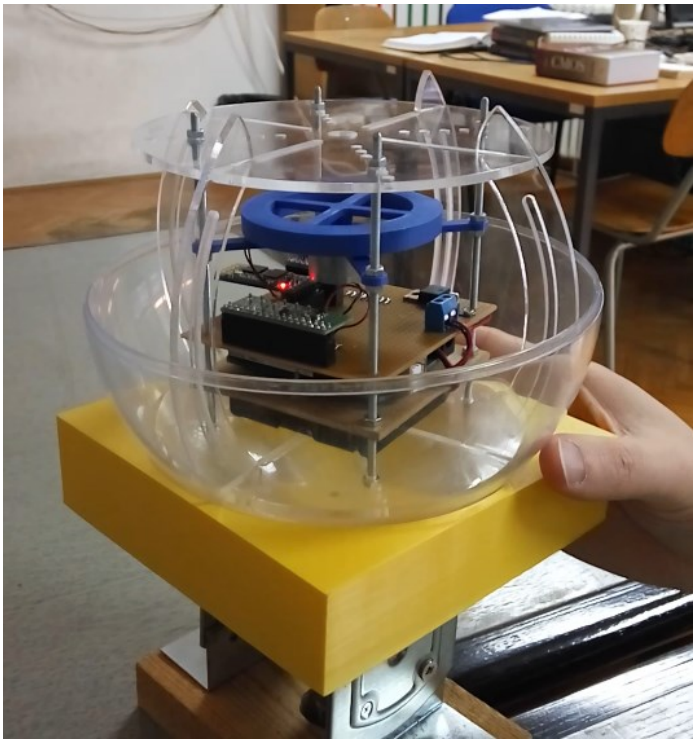

Figure 1 - Test-bed

METHODS AND MATERIALS

The system consists of spherical air-bearing and the sphere holding CubeSat. CubeSat has the Inertial Measurement Unit (IMU) consisting of the Gyroscope for the angular-velocity measurement and one Reaction wheel (blue wheel in Figure 1) which generates torque over one-axis. System is measuring angular-velocity about down-pointing body-axis. Because the center of gravity is not in the center of the sphere but much closer to the table surface, the sphere has the tendency to position itself so that the center of gravity is in its closest distance with the table surface.

If we tweak the position of the CubeSat in the sphere so that the resulting angular velocity generated by the reaction wheel causes rotation only in one plane, we will get the one-degree-of-freedom system with the rotational motion equation:

$$I\dot{\omega} = \sum T \,,$$

where I is the moment of inertia of the system, $\dot{\omega}$ is the angular acceleration and $\sum T$ is the sum of the torques generated by the reaction wheel and friction of the spherical air bearing respectively.

As for the controller, we choose the parallel PI controller with the equation:

$$u(t) = K_p e(t) + K_i \int_0^t e(t')dt' \,,$$

where K_p and K_i are coefficients for the proportional and integral term, $u(t)$ is the output of the regulator and $e(t)$ is the input of the regulator denoting difference between desired setpoint and measured process variable.

If we look at the system block diagram (Figure 2), we see that in the Regulator block (left box with dashed lines) there is *PWM generator* block. PWM (*Pulse width modulation*) generator is used as the Reaction wheel driver. PWM generator for its input uses discrete values in range $[-255, 255]$ where higher absolute values generate greater torque and the negative values are denoting rotation in the opposite direction.

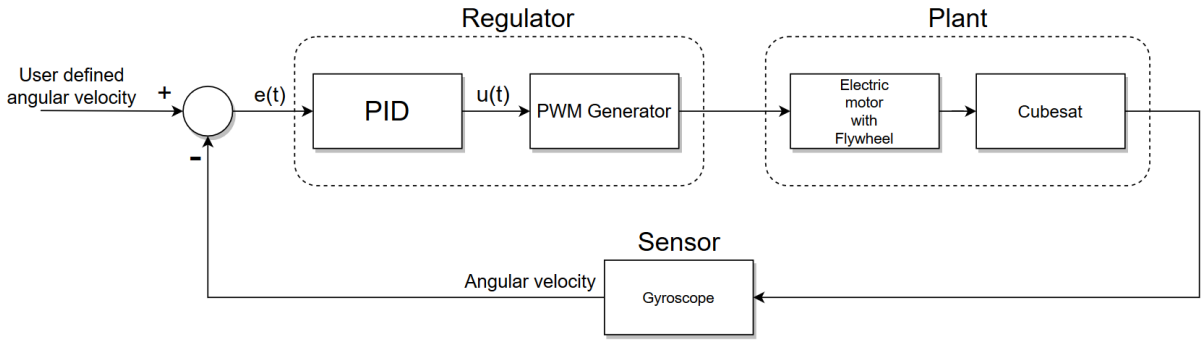

Figure 2 - Block diagram of the system

To find the transfer function of the Plant, firstly we needed to capture the system step response. The system block diagram for capturing the step response is shown in Figure 3. Notice that we left out the PID controller from the block diagram. The generated step function is directly fed into the PWM Generator block. As for the step function, we used the 200 as the final value (out of 255). That value seemed reasonable to us since it puts the Reaction wheel to about 78% of the maximum angular velocity (assuming the electric motor has linear characteristics).

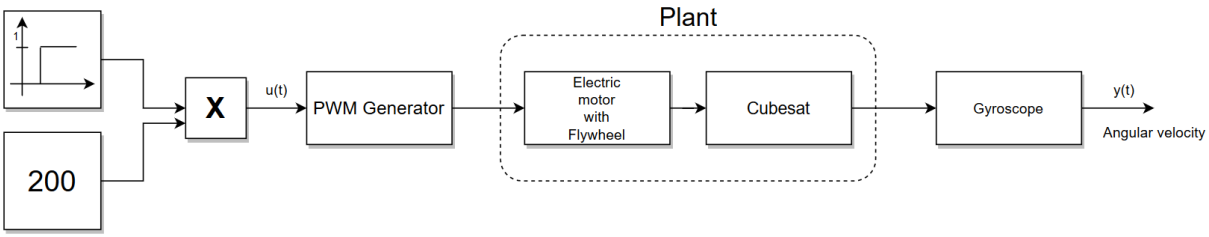

Figure 3 - Block diagram for the step response

RESULTS

Based on the step response of the system (Figure 4) assumption was that the Plant has transfer function with one dominant pole. Notice that after about 5 seconds the angular velocity starts to decrease with constant rate. This is due to the imperfections in the spherical air-bearing.

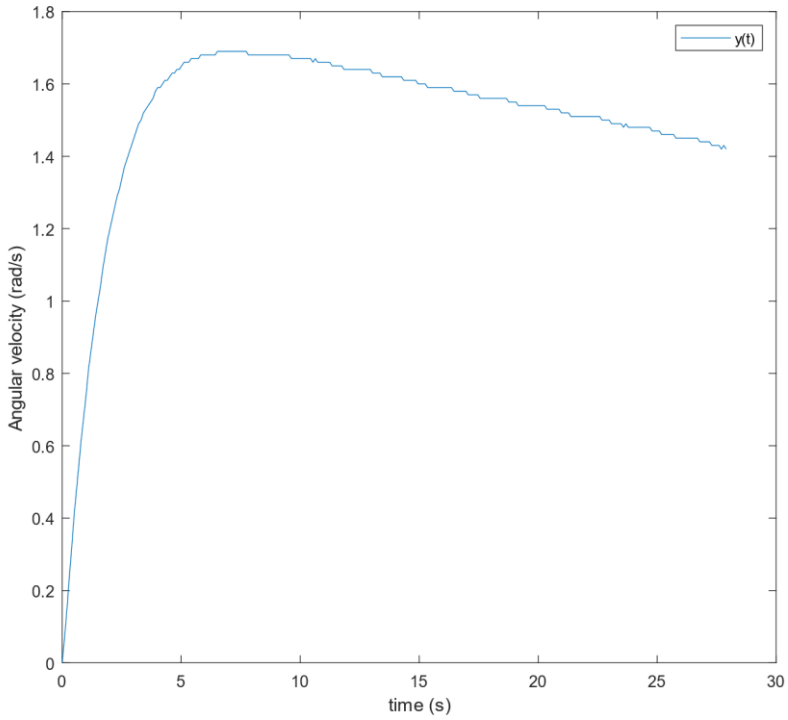

Figure 4 - System step response

To get the step response equation we used *MATLAB Curve fitting* app to fit the step response measurements to the user defined function defined as:

$$y(t) = Ke^{-at} + c,$$

where $K, c, a \in \mathbf{R}$.

The app window is shown in Figure 5. Notice that the red points on the graph are excluded from the equation fitting. The blue line is the fitted function with the coefficients below.

The app gave us values for constants with 95% confidence bounds:

- $K = -1.783\ (-1.794, -1.773)$,
- $a = 0.5999\ (0.5911, 0.6088)$ and
- $c = 1.734\ (1.728, 1.74)$.

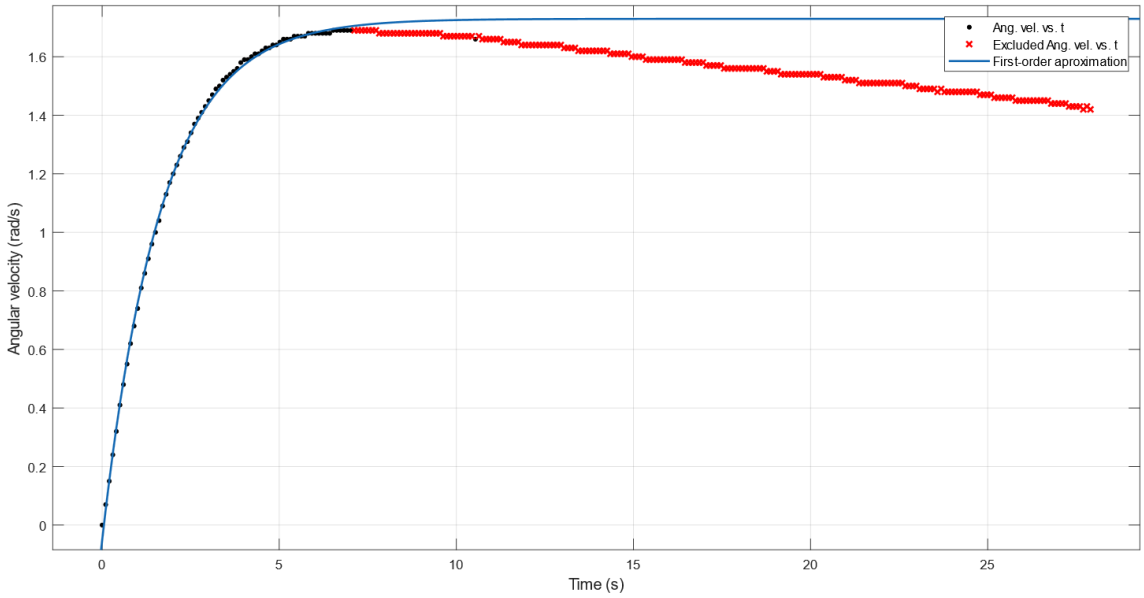

Figure 5 - Equation fitting

To get the transfer function of the Plant defined as:

$$G(s) = \frac{Y(S)}{U(S)},$$

user must find Laplace transformation for both $y(t)$ and $u(t)$.

The Laplace transformation of the PWM Generator input $u(t)$ is defined as:

$$U(S) = \frac{200}{s},$$

and the Laplace transformation of the output y(t) is defined as:

$$Y(S) = \frac{(K + c)s + a}{s(a + s)}.$$

With everything substituted and rewritten we get the Plant transfer function defined as:

$$G(S) = \frac{-0.05s + 1}{193.5s + 115.5}.$$

With that solved, we proceed to find the optimal K_p and K_i values with another MATLAB app called *PID Tuner*. *PID Tuner* allows user to interactively choose PID parameters based on the system response to the Heaviside step function. The parameters were selected in order to minimize Rise and Settling times while also preserving $\sim 0\%$ overshoot (see Figure 6). Another important restriction was that the PID controller output wouldn't get above ~255 (Figure 7).

The app gave us these optimal parameters:

- $K_p = 238$ and
- $K_i = 148$.

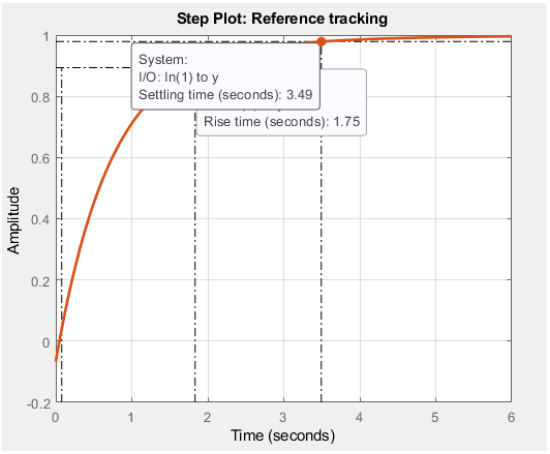

Figure 6 - Regulated plant response to the 1 rad/s set value

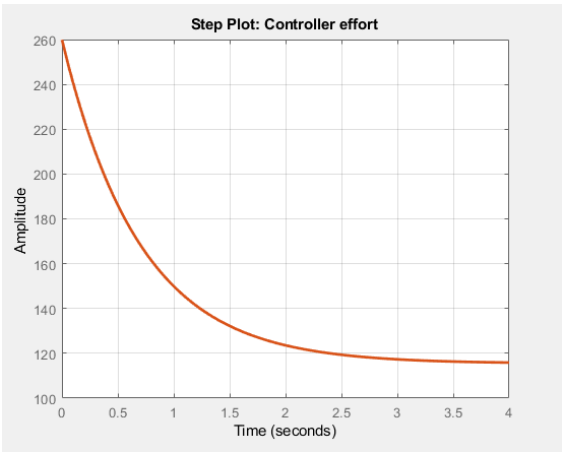

Figure 7 – Controller effort to the 1 rad/s set value

The system response and controller effort to the step function $\omega = 1\ {}^{\mathrm{rad}}/_{\mathrm{s}}$ are depicted in Figures below. These screenshots were directly obtained from the PID Tuner app. Notice the Rise and Settling times.

DISCUSSION

The results were tested, and the system is acting as predicted. This method proved to be much better method than the Ziegler–Nichols (ZN) for example. The main problem with ZN method is that one must put the Plant into oscillatory behavior. We couldn't manage to do that possibly because our Reaction wheel can't generate enough torque for the oscillations to occur.

As for the apps, MATLAB was chosen because it provides fast and user-friendly tools. If User wants to avoid MATLAB, he could fit the equations "by hand" because the equations are not so complex. Also, user could easily tweak PID parameters also by hand to get the optimal Plant behavior.