University of Missouri-Kansas City Fall 2021

**ME 457 – Mechatronic System Design**

**CHECKPOINT #2: Due in labs the week of November 1st, 2021**

**Checkpoint Synopsis:**

The objectives of Checkpoint #2 is to create a short How-To manual that describes how to implement and tune a **roll rate** stabilization controller, demonstrate the stabilization for roll rate, **AND** describe how the cascade controller works along with the stabilized roll angle test data. You will need to describe how a PID controller works, how to implement the controllers, and how to perform **closed loop** gain tuning. The manual should be written such that you could read the manual in several years and be capable of implementing and tuning a cascade PID controller.

1. Describe in detail how a PID controller works **conceptually** in the context of stabilizing the quadcopter roll **rate**. What do each of the gains do?

The x component(north) data from the gyroscope must be ran through a second order

Butterworth filter. The datasheet for the gyroscope will provide the scalar value that the filtered

signal can be divided by to provide accurate roll rate data. The PID rate controller will have three

components, proportional, integral, and derivative, and use the filtered roll rate data and a

desired roll rate variable. The error is the difference between the desired roll rate and actual filtered roll rate. The error will be used in the controller. To roll a plus configuration quadcopter motor commands must be sent to motors 3(west) and motor 4(east).

Diagram

Description automatically generated

Motor Commands:

Each controller component has a scalar component, and (gains). The proportional

controller is the product of and the error. The integral controller is the product of and the

integral of the error. The derivative controller is the product of and the derivative of the error. The purpose of the proportional controller is to increase/decrease the rise time. Increasing will increase % overshoot and decreasing will decrease % overshoot. The purpose of the integral controller is to reduce steady state error. Increasing will increase % overshoot and

decreasing will decrease % overshoot. The purpose of the derivative controller is to reduce %

overshoot. Increasing will decrease rise time and decreasing will increase rise time.

1. Describe in detail how to experimentally tune the roll **rate** stabilization controllers using closed loop gain tuning.

Once the controllers are built a gain tuning method (Ziegler-Nichols) must be implemented to establish starting and values. To perform the Ziegler-Nichols method set the desired roll rate to 25 and using only the proportional controller scale up or down until the drone oscillates without making full rotations. The value used to produce the oscillations is known as . The period between the oscillations is known as . and are used in a Ziegler-Nichols table to provide starting and values.

Ziegler-Nichols Table With Equations:

|  |  |  |  |
| --- | --- | --- | --- |
|  | **K**P | **T**I | **T**D |
| P | .5Ku | - | - |
| PI | 0.45Ku | Pu/1.2 | - |
| PID | .6Ku | Pu/2 | Pu/8 |
| PD | 0.8Ku | - | Pu/8 |

From here and need to be increased/decreased according to the and relations stated above until the following target parameters are met.

Parameters:

* Percent Overshoot 15%
* Settling Time 1 second
* Rise Time .4 seconds

1. Show the closed loop Z-N test with the roll oscillations. Show your values/calculations for

the PID gains.

Using :

Chart

Description automatically generated

Therefore

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **K**P | **T**I | **T**D | **K**U | **P**U |
| P | 0.0035 | - | - | 0.007 | 3.15 |
| PI | 0.00315 | 2.625 | - |  |  |
| PID | 0.0042 | 1.575 | 0.39375 |  |  |
| PD | 0.0056 | - | 0.39375 |  |  |
| P controller: | |  |  |  |  |
|  | **K**P |  |  |  |  |
|  | 0.0035 |  |  |  |  |
| PI controller: | |  |  |  |  |
|  | **K**P | KI |  |  |  |
|  | 0.00315 | 0.0012 |  |  |  |
| PID controller: | |  |  |  |  |
|  | **K**P | KI | **KD** |  |  |
|  | 0.0042 | 0.002667 | 0.001654 |  |  |
| PD controller: | |  |  |  |  |
|  | **K**P | **KD** |  |  |  |
|  | 0.0056 | 0.002205 |  |  |  |

1. Conduct a step response **using your Z-N gains** where you the quadcopter goes from a desired rate of zero to 30 degrees/s. Show the results (**clearly indicate where the step input is started)** and estimate the percent overshoot, settling time, and rise time (time to go from 0 degrees to 30 degrees/s). Comment on your results.

Using and :

Chart

Description automatically generated

It appears that the steady state value is approximately 250 deg/sec. If this is true, then the % overshoot is overperforming. Based on this assumption and the fairly controlled oscillations, appears to be performing decently. Increasing would reduce the large steady state error and increase the small % overshoot. It appears needs to be increased.

(The roll rate will never settle due to the unbalanced system. This is a predicted estimate of what the settling time might be if the system was balanced.)

The estimated settling time is overperforming. This also points to performing decently.

Although the rise time is below the acceptable threshold, the steady state error is way to large. The gain is currently too large. It appears that is dominant so it will be decreased during the initial tuning trials. It is suspected that will need to increase but during the initial trials it will remain constant.

1. In order to safely fly a quadcopter, your rate stabilization controller needs to be fast and accurate. Modify your gains until you have met the required performance for a 30-degree/s step response. **Percent overshoot < 15%, settling time < 1.0s,** and **rise time < 0.4s.** Show the results for **PID controller**, with the step input time clearly indicated. Make sure to provide your estimated percent overshoot, settling time, and rise time. Remember, you want your quadcopter to be as stable as possible (you will be performing free-flight testing in several weeks and the more stable your controller is the easier it will be to fly your quadcopter).

PID Ziegler-Nichols:

Noise from the derivative controller is too large to see proportional and integral controller so below is a plot of just the proportional and integral controller.

PID Ziegler-Nichols:

Plotting the PID controller components individually revealed that the derivative controller was noisy and the integral controller is subject to drift. The integral and derivative controller filters only successfully filtered the derivative controller so the tuning trials were done with a PD controller starting with Ziegler-Nichols values produced the following results

PD:

Chart, line chart

Description automatically generated

It appears after the max peak the predicted steady state, , is about the peak oscillations. Rise time is approximately .25 seconds and the signal never settles. Percent Overshoot is difficult to determine because the max peak is a the second peak. The percent overshoot is approximately 50%-70%. These gains provide the current best performance but don’t meet the required parameters.

1. Describe the cascade roll **angle** control architecture. What does the outer loop do? Why do we need the outer loop?

A rate controller is useful but difficult to control therefore it is not at all practical. This is why it must be integrated with a angle controller to produce a cascade controller.

Diagram

Description automatically generated

The figure above is a block diagram of the cascade controller. The inner loop is rate controller. In a cascade controller the desired rate is the output from the angle controller. The input to the angle controller is the angle error. The angle error is the difference between the desired input from the user and the actual angle.

1. Experimentally tune your roll **angle** control system (cascaded inner and outer loop controller) with a 20 degree step input in the desired **angle**. Your controller must be capable of satisfying the following performance metrics for the 20 degree step input: **Percent overshoot < 15%, settling time < 1.2s,** and **rise time < 0.4s**. Show your results in a figure, provide your final gains for the full controller, and describe what you found.

I could not get my roll controller to function correctly. The erratic behavior of the drone is so much so that a roll vs. time graph would not provide any useful information.

Below is the cascade controller code:

# initilized variables outside the loop

roll\_rate\_error\_old = 0

error\_i = 0

error\_i1 = 0

error\_i2 = 0

f\_error\_i1 = 0

f\_error\_i2 = 0

error\_d1 = 0

error\_d2 = 0

f\_error\_d1 = 0

f\_error\_d2 = 0

throttle = 50

#desired\_roll\_rate = 25

desired\_roll = 45

kp = 0.003

ki = 0

kd = 0.002

kp\_angle = 1

# cascade controller

desired\_angle\_error = desired\_roll - filtered\_acc\_roll

desired\_roll\_rate = kp\_angle \* desired\_angle\_error

roll\_rate\_error = desired\_roll\_rate - filtered\_gx

# proportional roll controller

proportional\_controller = kp \* roll\_rate\_error

# roll error integral

error\_i\_new = ((roll\_rate\_error + roll\_rate\_error\_old)/2)\*(now-then)

error\_i = error\_i + error\_i\_new

# roll error integral filter

filtered\_error\_i = (b0he\*error\_i) + (b1he\*error\_i1) + (b2he\*error\_i2) - (a1e\*f\_error\_i1) - (a2e\*f\_error\_i2)

error\_i2 = error\_i1

error\_i1 = error\_i

f\_error\_i2 = f\_error\_i1

f\_error\_i1 = filtered\_error\_i

# integral roll controller

integration\_controller = ki\*error\_i

# roll error differentiation for Kd

error\_d = (roll\_rate\_error-roll\_rate\_error\_old)/(now-then)

roll\_rate\_error\_old = roll\_rate\_error

# roll error derivative filter

filtered\_error\_d = (b0l\*error\_d) + (b1l\*error\_d1) + (b2l\*error\_d2) - (a1\*f\_error\_d1) - (a2\*f\_error\_d2)

error\_d2 = error\_d1

error\_d1 = error\_d

f\_error\_d2 = f\_error\_d1

f\_error\_d1 = filtered\_error\_d

# derivative controller

differentiation\_controller = kp\*filtered\_error\_d

cmd.channel[3] = throttle - proportional\_controller- integration\_controller - differentiation\_controller

cmd.channel[2] = throttle + proportional\_controller+ integration\_controller + differentiation\_controller

References

Hema, K. “Aircraft Flight Control Simulation Using Parallel Cascade Control.” *International Journal of Instrumentation Science*, Scientific & Academic Publishing, http://article.sapub.org/10.5923.j.instrument.20130202.02.html.