SOEN 385 Project: Telescope

To:

Dr. Javad Sadri

Hossein Khodadi

Team:

|  |  |
| --- | --- |
| Name | Student ID |
| Jordan Hubscher | 27019696 |
| Mathieu Diab | 26525318 |
| Dmitry Svoisky | 26893750 |

# 

# Table of Contents

[1. Analysis](#h.s3hur1wmnu69)

[2. Design](#h.vvu67wj17rhj)

[3. Implementation](#h.rznz7wgv4k3g)

[4. Testing Results](#h.elrdftsxblq1)

[5. References](#h.j2j40rc7elf1)

# 

# 

# 

# 

# 

# 

# 

# 1. Analysis

General PID controller Transfer function:

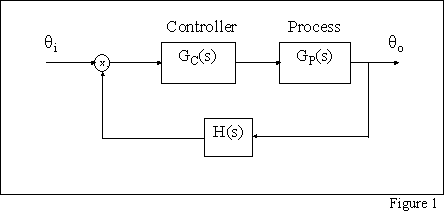


Figure 1.1 [1]

Normally, there would be a combined transfer function that would be manually adapted to reveal and adjust the appropriate constants in order to stabilize the system. However, in order to save time and efficiency, it was decided to the pre-implemented Simulink PID controller block that provides an automated tuner and input fields for each respective constant for complete customizability. Each permutation of the controller is represented by Simulinks controller block function listed below:

P I D :

P I :

P D :

Where N represents a coefficient that aids in normalizing the output. However, this controller block along with its unconventional coefficient ended up causing a little bit of trouble in the end when it came to specifically, and more accurately stabilize the system to the required constraints.

Below are the stabilized values for the for the given PD and PID controllers as well as the graphs of the unstable systems with adjusted P and PI values [2].

No tune/stabilized system exists for P or PI since P simply because P will continue to increase the overshoot and further disrupt the system which required a delicate 2% overshoot. As for PI, the I integral time variant constant will continue increase the system’s settling time which also is not good and is not a viable stabilizing controller method since the requirements of the system was to have a settling time of up to 1 second. For the tuned values;

PD was;

P = 0.506676436235224

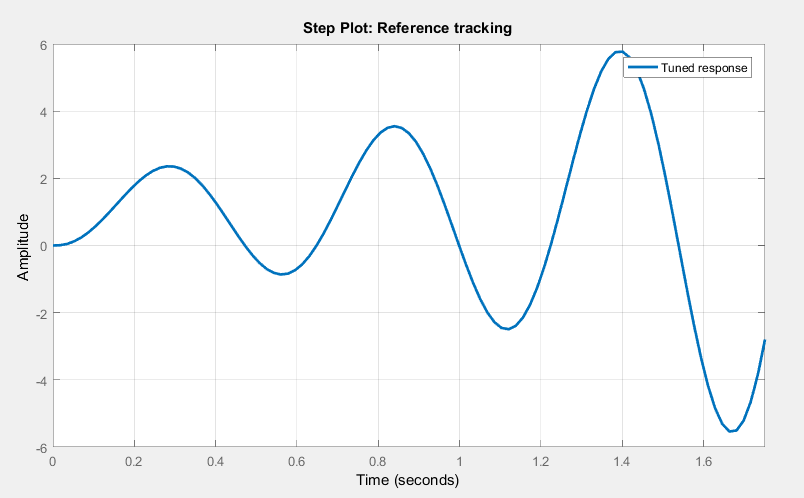
D = 3.9314301358739

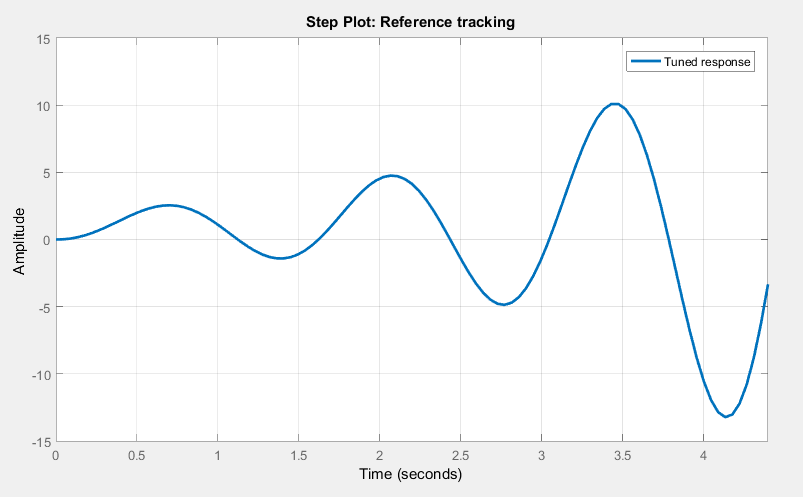
And PID was;

P = 0.480536537696423

I = 0.0555394282551153

D = 4.00054874676089

Figure 1.2: Output P

Figure 1.3 : Output PI

# 2. Design

As far as design is concerned, the Simulink model has two separate open feedback loops for each respective motor. Each feedback loop is fed their respective phi and theta angle rotations (for the given star coordinate). Once each rotation had been achieved by each motor, both angles as well as the axis’ that they were rotating around (these axis represented by a combination of 1 or 0 for each x, y, and z axis or linear combination therein) were fed into a matlab function block. So essentially, those eight inputs into the function block represent two quaternions (one for each motor), and the function was able to perform the transformation of combining them to output the single quaternion representing the final coordinate rotation/position of the telescope.

The 3D telescope was modeled in two parts; a transparent hemisphere for the base, first motor for rotation in the xy-plane. The second part is the main head of the telescope represented by a cylinder which will rotate about the zx-plane, and appear to rotate accordingly for each star coordinate.

Finally, the GUI of the system (shown by Figure 2.1 below) was simply designed to start and stop the system based off the provided star coordinates text file. Beforehand, the star coordinates were imported into a 3x100 matrix and then outputted the all but the first column to the Simulink model (since the first column was not being used. Rho in this case). Also, there are a selection of radio buttons provided, in order to test the different tuned values of the controller’s proportionality, integral, and derivative constants of the controllers.

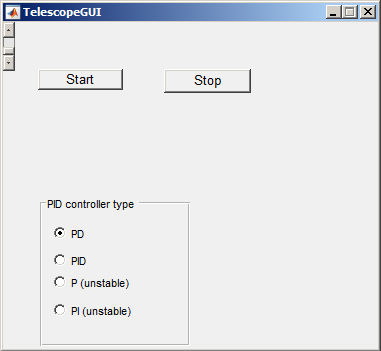


Figure 2.1

# 3. Implementation

The system was simulated using Matlab’s Simulink library to place each transfer function and controller as discrete blocks. Matlab function blocks were used to iterate over the the angles of each star: once the angle fell within a certain margin of error, the program was set to pause for 10 seconds to simulate two photographs being taken. Another matlab function block was used to translate the given angles into an appropriate vector format that was passed to a VR sink: the sink was used to display the telescope as a 3D simulation of where it was pointing at a given time.

Implementation was not without fault. Tuning the stable PD and PID controllers to achieve a maximum 2% overshoot and settling time under 1 second turned out to be a difficult process which produced a controller transfer function that seemed possibly “too fast”. The method used to determine when the telescope had achieved its ideal angle was also required a little bit of trial-and-error and relied upon the telescope pointing within 0.001 radians of the desired location. Increasing this acceptable error produced movement in the telescope that would immediately change its intended direction as soon as it fell within the value once (before it had even reached a peak and naturally overshoot, for instance), while decreasing it meant the telescope spent longer than it should have simply adjusting itself to the needed angle. Finally, ending the simulation to log time once every star was photographed was not 100% precise either. So the simulation’s end is simply marked by a denoted movement pattern and a console status message in the console stating the end of the simulation.

# 4. Testing Results

When running the Simulink model, each star took approximately 10 seconds (total settling and rise time for each motor, as well as the feedback error). So, to go through the entire list of star coordinates, it would be 1000 seconds or 16.66 minutes total worth of photographing.

# 

# 5. References

[1] "PID Tutorial." 12 Feb. 1999. Web. 15 Apr. 2016. <http://webber.physik.uni-freiburg.de/~hon/vorlss02/Literatur/Ingenieurswiss/pid/pid matlab/ien systems tutorial.htm>.

[2] "PID Control Design." Web. 15 Apr. 2016. <http://ctms.engin.umich.edu/CTMS/index.php?example=Introduction&section=ControlPID>.