TWO-AXIS CAMERA GANTRY SYSTEM DESIGN REPORT

by
Chizara Okoye

This report covers each component of the system, discussing
the design and optimisation approaches.
GitHub link: https://github.com/zaraokoye/Two-Axis-Camera-Gantry-.git

Reference input

The target performance of the system can be characterised by the target position of the gantry. In the y axis, the camera should have a range of motion 130mm either side of the position bias 217.5mm, enabling full left and right movement without any part of the camera coming in contact with the frame. In the z axis, it was more beneficial to assume the position bias as 287.5mm and have a range of motion 235mm either side of that. We needed to account for the top and bottom of the camera not touching the top and bottom of the frame as well as the runner.

PID Controller

In the y axis, a PI controller was implemented while in the z axis a PID controller was used. The final decided gain values for both are detailed in the table 1. Initially, a response optimisation was carried out to automatically tune the controllers. There were several difficulties experienced in the use of blocks and auto-tuning. When a step test was carried out, the optimisation would fail to complete due to "step size tolerance" [5]. The decision was made to tune it manually through trial and error, which eliminated the difficulties [3] [4]. Through a series of trial and error [1] [2], the step response acquired from the initial trial gain values also detailed in table 1, to the final values had the following improvements:

- In the y axis, the increased K_i resulted in a decreased steady-state error. The elimination of K_d resulted in a minimisation of noise in the system and a smoother signal.
- In the z axis, K_p and K_i was reduced to improve the stability of the signal and decrease the settling time. The decreasing of each of the gain values also benefits the system by minimising cost and operating power.

A fail-safe mechanism used to avoid collisions with the frame was the implementation of an input saturation that set limits on the reference input fed to the PID controller – shown in table 1. If an input was provided by a user that is beyond the limits of the saturation, the output position will be clipped at the limits set in the block [6].

DC motor and Plant

A Simscape modelling of the DC motor was chosen because it physically models the system enabling us to visualise its behaviour. A reducing gearbox was used in both axes. The role here was to increase the output torque of the motors. This is essential to create enough power for the motor to overcome friction (particularly in the y axis). In the z axis, a significant amount of power is needed to overcome the force of gravity and move the camera back upwards once it descends. However, the trade-off is the decrease in motor speed. A balance must be struck here; the gear ratio must be high enough to move the camera smoothly but low enough for movement at an adequate speed. Therefore, the gear ratio is higher in the z axis and the no-load speed must also be higher to counteract the inverse relationship between torque and speed.

Encoder and State estimation

The state estimation measurement is fed back to the PID controller to create the feedback loop [7][8]. Despite being unavoidable, the encoder and IMU were major sources of electrical noise in the system. The encoder in both axes is seen to generate notable noise as seen by the jagged nature of the signals in the below.

Tests and Model Verification

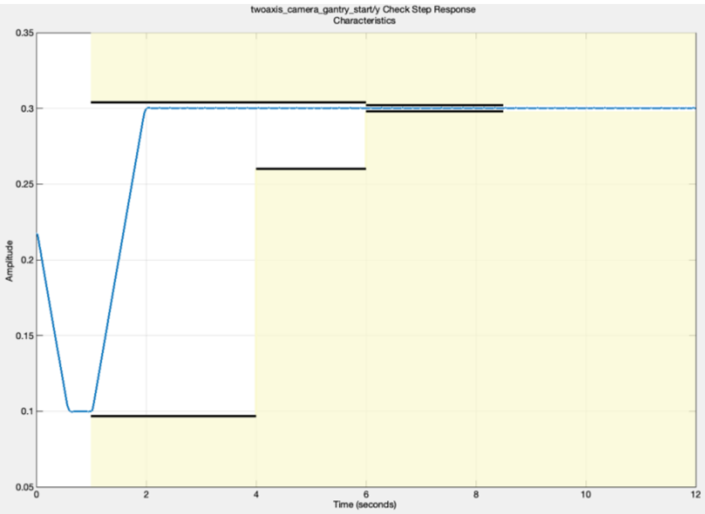
The following response tests were carried out;

- Step Response: A step input was sent to the system to test how fast it reaches a specified target position and the response was checked according to the design requirements. In the y axis, through extensive tuning of the PID controller, a step response was generated which passed all the requirements [3]. The z axis had a slower response to the step input. There was a persistent settling time discrepancy which was resolved with the manual tuning of the integral of the controller – lowering K_i [4].
- Reference Tracking: A sine input was sent to the system and the response was tracked according to the tracking goal constraints. The mean was calculated according to each signal. A relative error of approx. 2.7% was seen in the y axis [9] while the z axis experiences an average of 0.29%
- Impulse response: An impulse signal was sent to the system to model noise. After the spike is sent in both y and z directions, we see that the output does not deviate significantly from its target and it restores itself quickly to its intended response [11] [12].

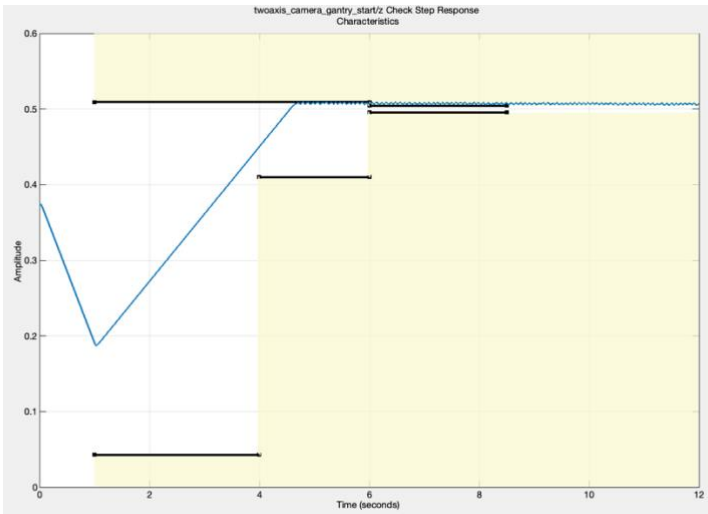
Appendix

Table 1: Design Parameters

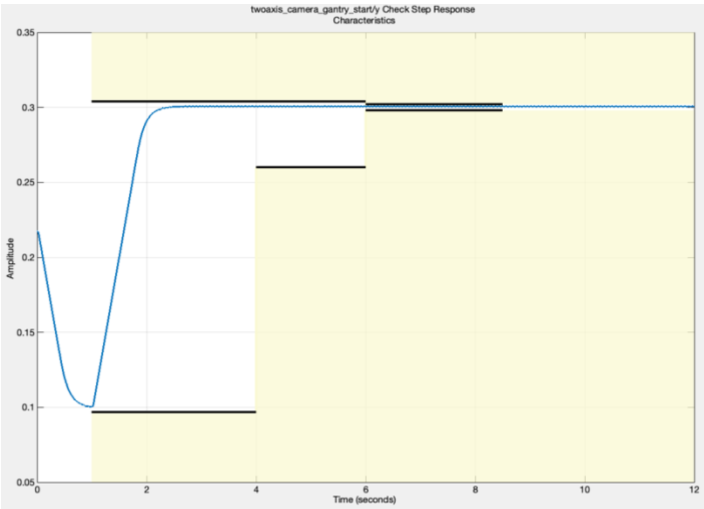
	Y axis	**Z axis**		**Y axis**	**Z axis**
Battery Voltage	12V	12V	**PID output saturation**	12 to -12	
Stall torque	0.05Nm	0.05Nm	**No-load speed**	14000rpm	16500rpm
Gear ratio	8	18	**Battery Resistance**	1Ω	1Ω
Measured Acceleration	$0m/s^2$	$10m/s^2$	**Battery Lifespan**	43.5 minutes	
Rise time	3 seconds		**Settling time**	5 seconds	
Overshoot	2%		**Undershoot**	1.6%	
Reference input amplitude	130mm	235mm	**Reference input frequency**	1rad/s	0.3rad/s
Position bias	217.5mm	280mm	**Input saturation**	348 to 30mm	575 to 0mm
Relative error	≈2.7%	≈0.29%	**State estimation bias**	217.5mm	375mm
Initial PID gain values	$K_p = 1000$ $K_i = 10$ $K_d = 90$	$K_p = 10000$ $K_i = 300$ $K_d = 500$	**Final PID gain values**	$K_p = 1000$ $K_i = 1$ $K_d = 0$	$K_p = 4500$ $K_i = 10$ $K_d = 10$



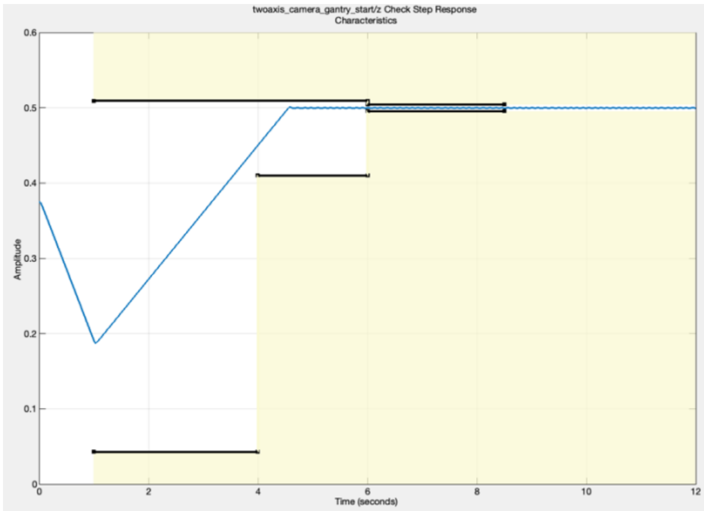
[1] – y Step Response with **initial** PID values



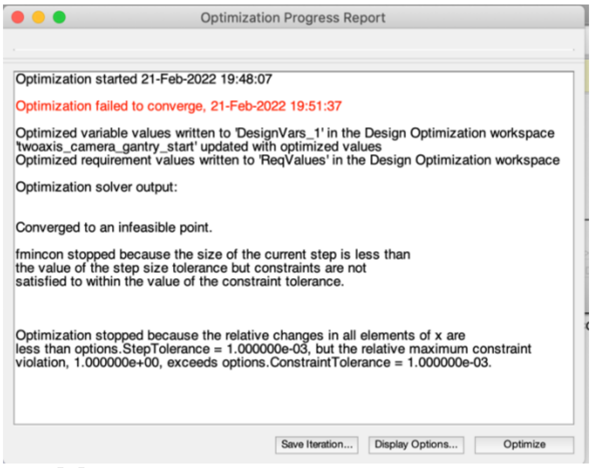
[2] – z step response with **initial** PID values



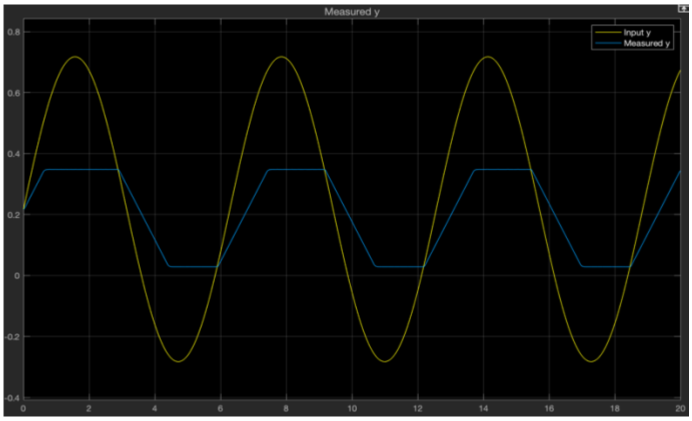
[3] – y step response with **final** PID values



[4] – z step response with **final** PID values

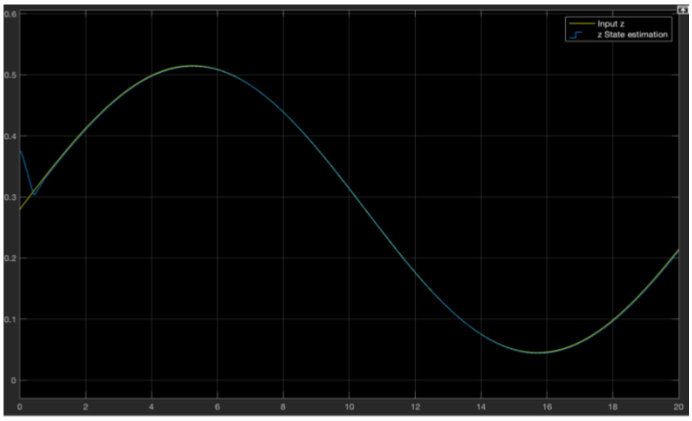


[5] – Failed Optimization Message

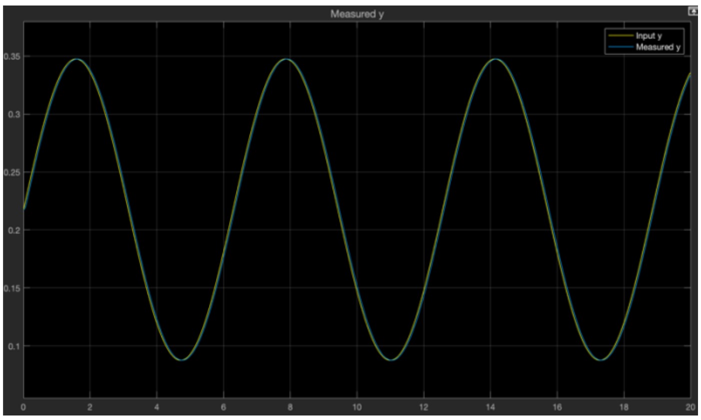


[6] – Example Input amplitude of 500mm produces a clipped output at 348mm to 30mm (input saturation limits)

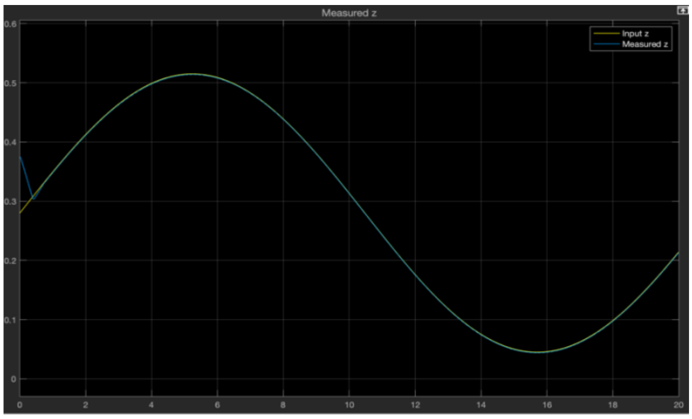
[7] – y axis state estimation reading



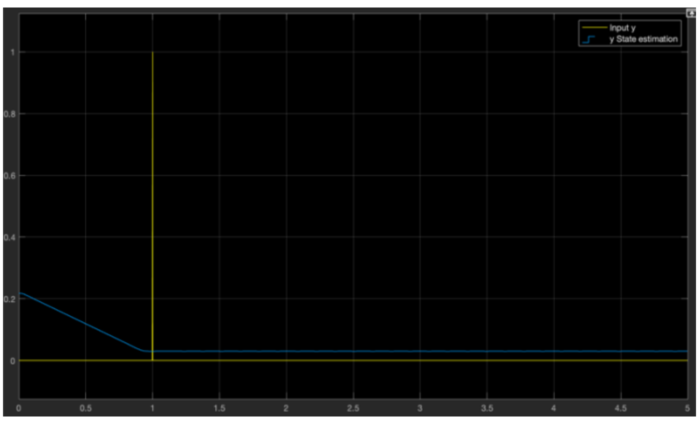
[8] – z axis state estimation readings



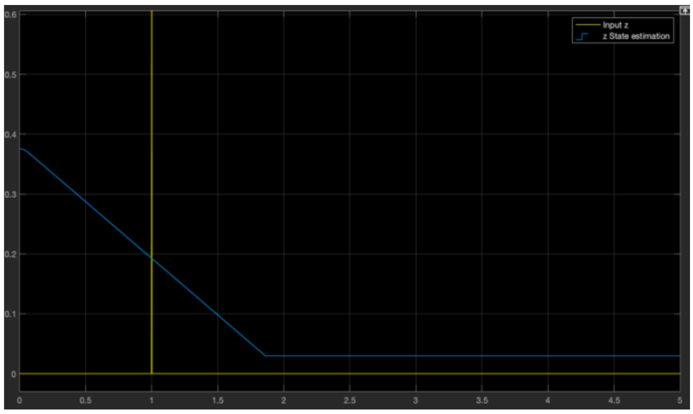
[9] – y axis reference and output position



[10] – z axis reference and output position



[11] – y axis impulse response



[12] – z axis impulse response