QEA Module 5 - Rocky

Utsav Gupta and Nathan Estill

November 4, 2018

1 Athlete Demographics

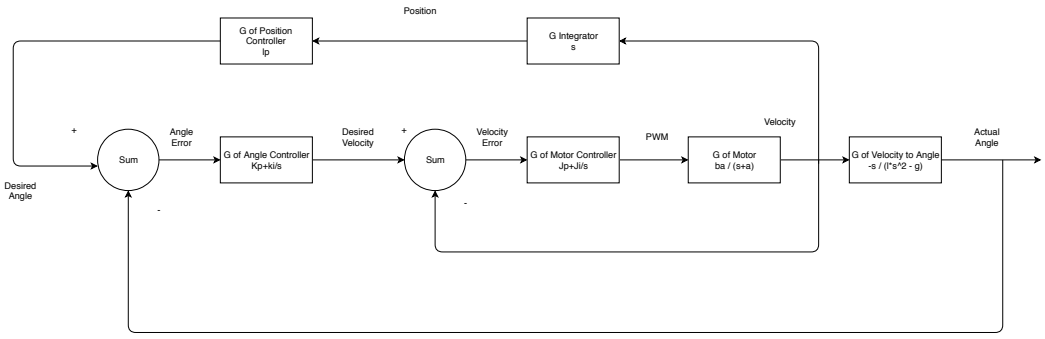

Figure 1: Block Diagram For Sprint Event

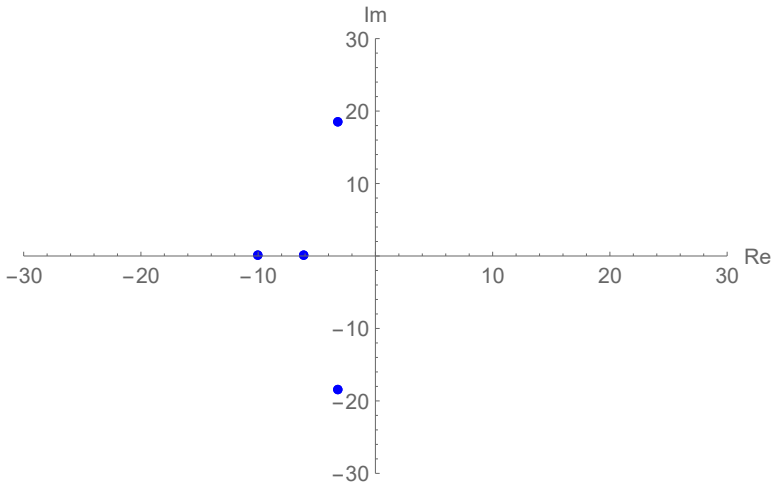

Figure 2: Poles for our values

$$T = \frac{-a * b * (J_i + s * J_p) * (K_i + s * K_p)}{(s(-g + Ls^2)(s + a) - ab(J_i + J_p s)(g - Ls^2 + K_i + s * K_p)} \tag{1}$$

2 Athlete Performance Information

The parameters we used were K_p = -7, K_i = -44, J_p = 500, J_i = 5000, and l_p = -0.45. K_p and K_i, the values that change the PI loop of the angle, were determined by analyzing the poles with different values,

and trying to get the imaginary portion closer to 0. J_p and J_i, the values that change the PI loop of the velocity, were determined from previous classes to best help the system. l_p, the value that changes the P loop of the position, was determined experimentally to get the oscillations to a minimum.

3 Training Session Description

We improved the robot to have controlled velocity, no runaway tendencies, and taught it how to run. For controlled velocity, we fed the outputted velocity into an error function with the desired velocity to get a consistent speed. We noticed that the robot would tend to go in the direction that it was leaning towards. To remove the runaway tendencies, we made the desired angle a linear function of the position from the starting point, so that it would tend to go back towards the center. To make it run, we made the "center" position go forward with time. That way, the robot would tend to go forward as well, while still correcting the angle along the way.

We trained our robo-athlete for the following events:

3.1 Survivor

In this event, we aimed to stabilize our robot, Apollo Creed, in an inverted pendulum position and confine its movements within a 2 sq.ft. box.

3.1.1 Hysteresis

Our initial attempt, on Day 1, involved controlling Apollo's movements by setting a velocity threshold in both directions it could move. Whenever Apollo would cross that threshold, it would trigger a command to invert Apollo's angle to be opposite to the direction of velocity.

This approach worked well and we found that Apollo would stay in the box for about 35 seconds before it wandered out.

3.1.2 Robo Breakdown

On Day 2, we received news that our beloved Apollo has had a breakdown. The other team at our table were working on QEA when an accident happened - Apollo's right motor stopped responding to commands. At this point we were scared that this might end Apollo's Robolympics career before it even began.

Our troubled mental state led us to adopt another Rocky, whom we dubbed as Ricky, and we tried working with it.

3.1.3 Hysteresis again

On Day 3, we tried implementing the same Hysteresis loop on Ricky, but we found that it did not work well. After a bit of experimenting with other Rockies, we concluded that our Hysteresis loop was extremely well suited to Apollo due to some weird coincidence, but would drive the other Rockies crazy.

At this point, we decided that Apollo needed to make a comeback. Thus we called Dr. Nathan Estill, who has a PhD in Robotic Repairs, to revive our long lost friend Apollo. What we found later was that Dr. Estill was the modern day version of Dr. Frankenstein, and he revived Apollo in the form of Estill's Monster.

Estill's Monster ran rampant throughout the 2 sq.ft. box and we decided to tame it using PI control loops to increase its chances of winning Robolympics, as described below.

3.1.4 Double Control Loop

On Day 4, we ditched the hysteresis approach in favor of having one control loop controlling angle, and another controlling velocity. This worked decently well, however Rocky would tend to go in one direction after a decent amount of time.

3.1.5 Triple Control Loop

To fix the problem of Rocky running away, we made Rocky go change its angle depending on how far away it was from the resting point. As such, when Rocky starts to go one way, the desired angle changes enough for it to stop, and then goes the other direction. This required some parameter tuning to make Rocky's oscillations tend to decrease rather than increase out of control.

3.2 Sprint

In this event we made our robot run a 20 ft dash to get to the finish line in the least amount of time possible, while remaining stable.

3.2.1 Constant Angle

At first we tried to set the desired angle to a constant slightly forward angle in hopes that it would naturally balance while going forward at a constant velocity. However, accelerating to that velocity while ending at the desired angle is very difficult, and doesn't use much feedback.

3.2.2 Triple Control Loop

We slightly altered the same approach as the triple control loop for the survivor event. Because the robot oscillates around a centralized point, we simply made that center point move forward as time went on. That way, it would still oscillate around the center point, but would tend to go forward as time went on, while sometimes going backwards to balance itself. When Rocky first starts, it is unstable, so we set a delay to let it become more stable before it began to go forward. To get Rocky to go faster, we increased the rate at which the point would move forwards until it seemed unstable.

4 Qualification Videos

To view the video for the Survivor challenge video please click here.
To view the video for the Sprint challenge video please click here.