Segway Olympics Report

Coach's Notes -- Coach Prava & Coach Sam E.
Mostly describing Segway Survivor

Athlete Demographics

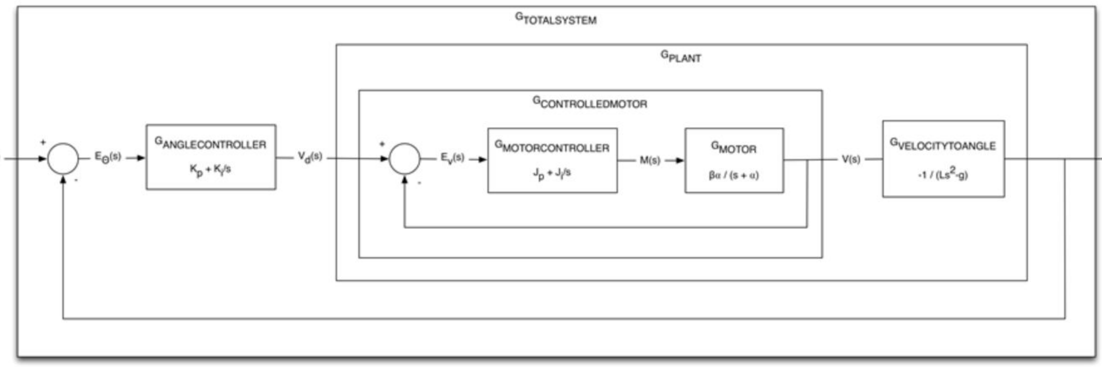

Figure 1: Overall Block Diagram - Credits to Paul Ruvolo for constructing it

This block diagram is essentially the combination of two block diagrams - one for cruise control, and the other for pendulum motion.

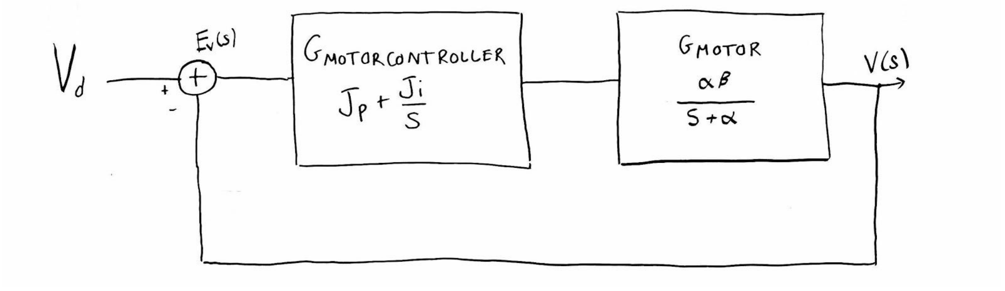

Figure 2: Cruise control block diagram

One problem with controlling the segways is, that we can't simply send a power to the motors, because the velocity output won't be what we expect - there is some error. In order to remove this error, we implemented PI control. This uses both proportional control (multiply some constant by the error) and integral control (multiply some constant by the sum of the error over time). Thus, with the cruise control block diagram, we are able to drive the segway at a desired velocity.

But what is this desired velocity? For the segway survivor, all we want is to keep the segway upright and stable. Thus, our desired input is the velocity that will allow the pendulum to remain upright and stable as fast as possible.

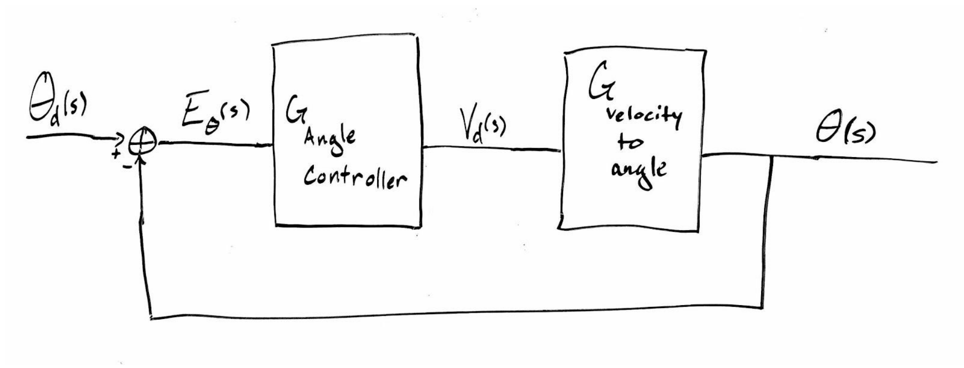

Figure 3: Keeping the inverted pendulum upright

The transfer function for $G_{ANGLECONTROLLER}$ is:
$Kp + \frac{Ki}{s}$
The transfer function for $G_{VELOCITYTOANGLE}$ is:
$-\frac{s}{Ls^2-g}$

This is the control diagram in keeping the pendulum stable. The desired angle, in this case, is zero (so that the pendulum will remain upright). This PI control diagram will take the angle at which the segway is positioned at a certain time and correct itself to reach zero. This block diagram uses the relationship between velocity and the angle in order to correct itself.

When we combine both of these diagrams, we get the resultant block diagram. The velocity at which we need to drive the segway is then fed into cruise control, and then the output is the actual velocity of the system. This then drives the segway to reach our desired angle, which in this case is zero radians.

Most of our variables in the block diagrams are known. Within the transfer functions, we know the values for the various coefficients, and define inputs and output variables when we implement PI control within our Arduino code. The values that we can affect are the control constants, which are Kp and Ki (the angle controller constants) and Jp and Ji (the motor controller constants).

To determine the motor controller constants, we first had to determine how we wanted the system to behave. We wanted to make sure it was stable - essentially we wanted to make sure it would reach our desired velocity. Secondly, we wanted it to reach steady state as fast as possible. The inverted pendulum, without any control, would fall to the ground within a short period of time, calculated to be $\sqrt{\frac{g}{L}}$. So, the output velocity would have to reach steady state as fast as possible as well. We don't want to oscillate either - we want the output velocity to match our desired velocity pretty closely over time. Due to these constraints, we wanted our system to be critically damped, and solved the Jp and Ji accordingly.

To determine the angle controller constants, we had a similar approach. We knew that we wanted the system to be stable because we wanted it to reach the desired angle (of zero radians, which would make it stable and upright). We knew that the desired velocity within the angle part of the control diagram had to be high enough so that it could stabilize before the segway would fall, but also not calculate a velocity that was so fast that the motors could not drive (the limit was about 0.75 m/s). We also knew that we wanted oscillations in our system - in order to keep it upright, we want the segway to oscillate around zero radians - and not just reach the upright position and essentially just 'stop'. Due to this, we actually wanted our system to be underdamped.

Keeping in mind the design constraints (maximum motor velocity, time it takes pendulum to fall), and the behavior of our system (critically damped, underdamped), we were able to solve for Jp, Ji, Kp, and Ki.

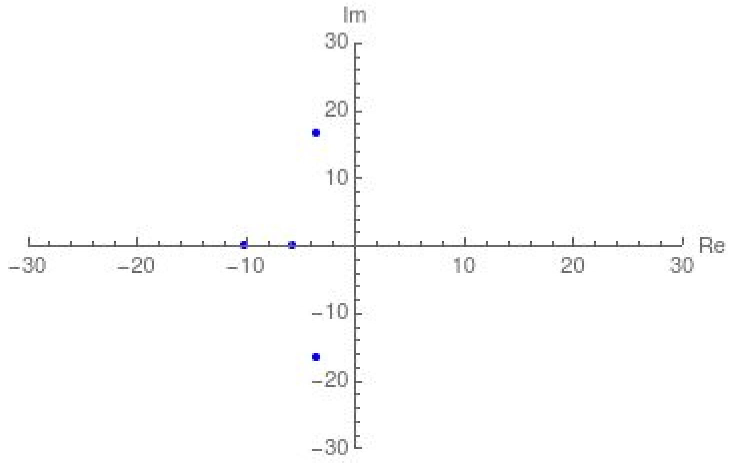

Figure 4: Pole-Zero Diagram of Chosen Jp, Ji, Kp, and Ki values

The specific values are not so much important as the poles and zeros the values cause (and the resulting behavior). With Jp = 500, Ji = 5062.5, Kp = -6, and Ki = -36, we were able to get the above pole-zero plot. With the motor controller (Jp and Ji), we wanted the system to be stable. This means both poles should be negative. We also wanted them to be critically damped. This means that the poles have no imaginary component (no oscillations), and there are two poles over the same point. We also wanted it to reach steady state as soon as possible so it doesn't take too much time to reach the desired velocity (before the segway would fall). The Jp and Ji value of 500 and 5062.5 met our criteria.

For Ki and Kp values, we wanted it to be stable, which means it should be negative as well. However, we also wanted the system to be underdamped. This means there is an imaginary component to it; this imaginary component is the natural frequency of the oscillations. We didn't want it to oscillation too much in order to stabilize, which means the magnitude of the imaginary component was minimized to the extent that it would still oscillate but not too much. We also wanted it to be as fast as possible in order to not fall (according to $\sqrt{\frac{g}{L}}$, with g being the acceleration constant and L = 0.25 m, we get a value of 6), so the poles will 'past' the 6-value point on the x-axis. However, we also want to make sure it's not too fast so that the wheels can't turn - to make it slower, it would involve moving the point farther to the left (which would decrease the time to steady state, thereby affecting the velocity at which the motors would try to turn). This would affect how far the poles are along the negative real axis - past the point that the velocity would just cause it to fall, but not so far (higher velocity being farther along the negative x-axis) that it would be too fast.

Athlete Performance Information

For the control parameters, we were able to calculate those according to the process described in *Athlete Demographics* and the calculations done out in the *Calculations* section.

For the other parameters located in our block diagram, please refer to the following table:

Parameter	**Determination**
$\theta_d = 0$ radians Desired angle at which we wanted the segway to remain	Our desired angle was primarily determine to how it was defined in the sensors that the Pololu Balboa had come with - in this case, the segway standing upright corresponded to an angle of 0 radians.
$\alpha = 10\ s^{-1}$ Rate constant for the transient response	To determine the transient response rate constant, we measured the time of the transient response (time leading up to steady state) when we set the motor from 0 to some non-zero power that could drive the motors. We could also look at the bode plot and look at the phase offset between the input and the output in order to determine this constant.
$\beta = 1/400$ Multiplicative factor between the system	We determined this by sending a velocity to the motors and determining what velocity was actually outputted.

input (motor command) and system output (velocity)	
$g = 9.81\ \frac{m}{s^2}$ Gravitational constant	Our boi Henry Cavendish.
L = 0.2 m Length of pivot point to mass - in this case, length from the center of the wheels to the center of the masses attached, or the length of the attached laser cut piece	Rulers are our favorite invention.

Training Session Description

We initially taught our athlete how to stand up by using the starter code we used in class. During our initial testing with our robot we attempted to remove the slight runaway tendencies we were observing through a couple methods. We tried resetting the integral of the error over time to 0 whenever the robot was within a small range of our desired theta. We additionally tried introducing a slight oscillation by toggling the desired forward velocity from a small negative to a small positive value - making it move back and forth. We also tried to account for runaway by taking the integral of the velocity and making the angle change based on how far away it was based on the position (so the angle would then lean towards the center and move back in order to correct). However none of these significantly accounted for our drift because we weren't experiencing drift initially for it to make a difference.

We trained our robot to sprint by subtracting the position (which was a function of time) from the integral of the velocities. By scaling the error angle up by the difference, we are able to set the robot to sprint forwards a scaled amount every second. The higher we set the position, the faster the robot would move - we eventually determined a change in position of 0.3 m was fast enough for our needs but would still allow the robot to remain upright.

Calculations

Note: these calculations use the transfer functions are defined above, and the overall control block diagram. The transfer functions above were found by converting their respective time domain equations into the Laplace domain - which we did in class, and therefore didn't feel a need to replicate the calculations here.

The overall transfer function for the motor controller system is:

$$G_{controlledmotor} = \frac{G_{motor}G_{motorcontroller}}{1+ G_{motor}G_{motorcontroller}}$$

Introducing the angle control, thus, is a trivial step as introducing the extra plant and control blocks into the overall system:

$$G_{totalsystem} = \frac{G_{anglecontroller}G_{controlledmotor}G_{velocitytoangle}}{1+G_{anglecontroller}G_{controlledmotor}G_{velocitytoangle}}$$

In order to get actual information from the system, I need to use the denominator of the resultant simplified equation (after plugging in the various individual transfer function). The total resulting equation is:

$$G_{totalsystem} = \frac{\alpha\,\beta\,(Ji + s\,Jp)\,(Ki + s\,Kp)}{s\,(-g + L\,s^2)\,(s+\alpha) - \alpha\,\beta\,(Ji + s\,Jp)\,(g - L\,s^2 + Ki + s\,Kp)}$$

To find the poles of the function, one would solve all four roots of the denominator. After plugging in the rest of parameters, to find stable roots, we would find the control parameters when the roots are negative. For Jp and Ji, we calculated them before using the transfer function of just the controlled motor, and found that the relationship between them was defined as follows:

$$Ji = \frac{-Jp^2-4gL}{4L}$$

We then chose Ji to be 500, mostly because we trust Paul Ruvolo and that is the initial Ji value he suggested. Through the above equal, we found Jp = 5062.5 when Ji = 500;

To find Kp and Ki, the process was a little different. We inputted the motor control parameters and the other parameters. Then we plugged in various Kp and Ki combination values. We ultimately ended up with the Kp and Ki value that Paul, once again, suggested - with Kp = -6 and Ki = -36. With these values, we got the following roots: -10, -5.7, -3.4 - 16.6 i, -3.4 + 16.6 i.

Note that they are all negative, meaning that they are stable and will reach steady state. The final two poles are the ones with the imaginary component - causing there to be oscillations in the system by the magnitude of the imaginary component (which is the natural frequency). The last two poles represent how fast the system will climb to steady state - and these values are large enough that the segway won't fall before it can stabilize, but small enough so that the maximum motor velocity would not be exceeded.

Qualification Video

Here is proof of our segway competing sans performance-enhancing radio transmitters.

Survivor: https://www.youtube.com/watch?v=qxl1vFxjbyU
Sprint: https://www.youtube.com/watch?v=arGC3LVTXEw