**Controls**

The controls portion of the project was broken up into several steps. The initial step consisted of determining the open-loop model of the system. We approximated the continuous-time system of each wheel using a discrete model, with distance traveled and instantaneous velocity being state variables. Since this system is discretized, a bit of estimation was required between time steps in order to calculate a sufficiently accurate value for distance traveled, so the velocity from the previous time step was multiplied by the time between samples and added to the previous distance traveled. With this model, we could then describe our system with the following matrices

where d is the distance traveled so far, v is the velocity, Ts is the time between samples, and u is the input that we apply.

The second step involved determining the eigenvalues of the closed loop matrix and the feedback values that would enable us to set those eigenvalues. Since our system has 2 state variables and u is a scalar input, B had to be a column vector in order to satisfy dimensionality requirements. The B values were determined by running the linear least squares algorithm on each row of our system using the data collected from running the dynamics\_data.ino file. The algorithm was used four times, since it had to be done twice for each wheel. We employed a state feedback system wherein our scalar input was the product of the feedback row vector [F1 F2] and the state of the system. This yielded the following closed loop matrix

with the corresponding characteristic polynomial

which was equated to the following polynomial

after which a system of equations was formed and the feedback values were solved for as a function of the desired eigenvalues. The eigenvalues were chosen to give a reasonable convergence time while also ameliorating any oscillatory behavior. We settled on a value of 0.9 once we determined that our car drove reasonably straight.

The final step of the controls portion consisted of implementing turning. To turn in a given direction, the wheel on that side needed to be slowed. Implementation was easy as we simply changed the desired velocity of that wheel to a fraction of that of the other wheel. We settled on a value of 0.045 for the slower wheel.