Senior Design
ENG EC 463

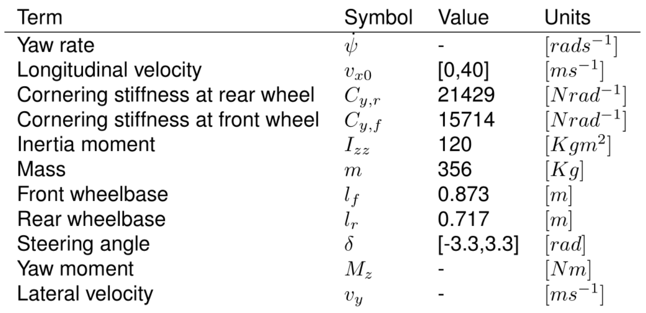
Memo
To: Professor Pisano
From: Team 9 (Giacomo Coraluppi, Alex Zhou, Will Krska, Jonathan Ye, Nick Marchuk)
Date: 11/19/2022
Subject: First Prototype Test Report

1.0 Equipment and Setup

1.1 Equipment

Simulink:

In Simulink we prepared a simple model based on a linear state function. There is a coefficient matrix A, which has parameters based on inputs from the environment, like tire stiffness and the distance between each tire and the center of mass. The model being linear requires some simplifications. The vehicle is modeled as if it were a two-wheeled vehicle, the only varying input from the driver being the steering angle. The angle between the fixed back wheel and the front wheel forms the steering angle δ. The coefficients $C_{y,r}$ and $C_{y,f}$ represent the cornering stiffness of the front and rear wheels, which will have to be determined for our specific car. A full set of definitions for the equations is below:



Term	Symbol	Value	Units
Yaw rate	$\dot{\psi}$	-	$[rads^{-1}]$
Longitudinal velocity	v_{x0}	[0,40]	$[ms^{-1}]$
Cornering stiffness at rear wheel	$C_{y,r}$	21429	$[Nrad^{-1}]$
Cornering stiffness at front wheel	$C_{y,f}$	15714	$[Nrad^{-1}]$
Inertia moment	I_{zz}	120	$[Kgm^2]$
Mass	m	356	$[Kg]$
Front wheelbase	l_f	0.873	$[m]$
Rear wheelbase	l_r	0.717	$[m]$
Steering angle	δ	[-3.3,3.3]	$[rad]$
Yaw moment	M_z	-	$[Nm]$
Lateral velocity	v_y	-	$[ms^{-1}]$

Figure 2: Variable names. Joao Antunes, Torque Vectoring pg. 24

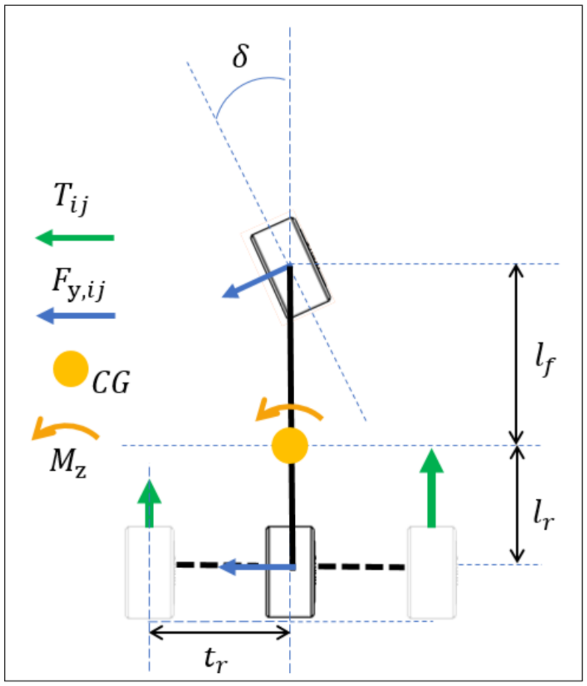


Figure 1: Bicycle model of a car. Joao Antunes, Torque Vectoring pg. 25

UART Interconnect:

We prepared a simple Arduino-based system that allows data to be read from the potentiometer sensors and transmitted over the UART communication protocol from one Arduino to the other. This emulates the connection that we will have between a remote microcontroller packaged at the front of the vehicle and the main control board towards the rear. The potentiometers represent a “throttle” and “steering” signal, which will also be coming from larger potentiometers on the real vehicle and output in the range of 0.5 to 4.5 volts.

1.2 Pre-Testing Setup Procedure

Simulink:

Launch Simulink and run the control model.

UART interconnect:

On a breadboard, wire two potentiometers, with the output (middle) pins connected to A0 and A1 of the microcontroller. Program and upload one microcontroller with the signal transmitter code and the other microcontroller with the signal sender code. Connect the serial interfaces between the two microcontrollers once both programs are successfully compiled and uploaded, as this may interfere with programming the controller. Run three jumper wires, connecting Rx to Tx in both directions, and connecting each microcontroller’s ground to ensure signal integrity.

2.0 Measurements Taken

2.1 Testing Procedure

Simulink:

Demonstrate that Simulink CodeGen works properly and can output interpretable C-code.

UART interconnect:

The inputs are the throttle and steering angle signals, which will be transmitted from the NANO to the UNO and displayed on the arduino serial monitor.

2.2 Measurable Criteria

Simulink:

C-code is generated.

UART interconnect:

- Connectivity: microcontrollers should have a stable connection between them
- Noise/variation: If each potentiometer is not being touched, then the transmitted signal should not vary by more than 1%, or ~10 out of a range of 0-1023
- Throttle output to left and right is varied based on steering input

2.3 Results

Simulink:

- We were able to show that the control model had a complete wiring of inputs to outputs, as well as a feedback loop to show the yaw rate
- We showed that the model compiled and could generate usable C code
- We demonstrated the various inputs and constants that must either come from sensors or be derived from C code.

UART interconnect:

- We were able to represent the connection between the two Arduinos as the signal calculated on one could be transmitted to the other and shown on the serial monitor of the receiver Arduino
- We were able to simulate how much power goes to each of the two rear wheels depending on the throttle and steering inputs. For example, when the throttle signal was at maximum (1023) and the steering signal was exactly at half (512),

this simulates the driver driving in a straight so the control signal would send nearly half of the total power to each of the two wheels.

- However, if the steering input was higher than 512, this would simulate for example turning the car to the right so the amount of power transmitted to the left wheel would be greater than the amount of power transmitted to the right wheel. Our program showed this by having the L signal be greater than the R signal. For example, if the throttle was 1000 and steering was 800, the L signal would be 800 while the R signal would be 200.

3.0 Conclusions

Through our first prototype test, we were able to demonstrate a very simplified version of our final product working. Obviously, in the final product the signals from the throttle and steering will be complemented by many other signals such as those coming from the accelerometer, etc. However, we were able to demonstrate that our product is feasible because we can transmit signals from one controller or sensor to another and use those signals to regulate another function of our product, in this case the power sent to each of the two rear wheels. In Simulink, we were able to show that we can create a basic model of the car and use that to generate C code which will be used in our controller. This provides a basis for both developing a broader model that could use pre-built vehicle dynamics control blocks, as well as further development of our own blocks that were made in a basic form for the test. The test model has places where constant values and assumptions can be replaced with permanent features, such as using gain scheduling rather than a constant gain.