Longitudinal_Vehicle_Model

July 27, 2019

In this notebook, you will implement the forward longitudinal vehicle model. The model accepts throttle inputs and steps through the longitudinal dynamic equations. Once implemented, you will be given a set of inputs that drives over a small road slope to test your model.

The input to the model is a throttle percentage $x_\theta \in [0, 1]$ which provides torque to the engine and subsequently accelerates the vehicle for forward motion.

The dynamic equations consist of many stages to convert throttle inputs to wheel speed (engine -> torque converter -> transmission -> wheel). These stages are bundled together in a single inertia term J_e which is used in the following combined engine dynamic equations.

$$\begin{aligned} J_e \dot{\omega}_e &= T_e - (GR)(r_{eff} F_{load}) && (1) \\ m\ddot{x} &= F_x - F_{load} && (2) \end{aligned}$$

Where T_e is the engine torque, GR is the gear ratio, r_{eff} is the effective radius, m is the vehicle mass, x is the vehicle position, F_x is the tire force, and F_{load} is the total load force.

The engine torque is computed from the throttle input and the engine angular velocity ω_e using a simplified quadratic model.

$$T_e = x_\theta (a_0 + a_1 \omega_e + a_2 \omega_e^2) \tag{3}$$

The load forces consist of aerodynamic drag F_{aero}, rolling friction R_x, and gravitational force F_g from an incline at angle α. The aerodynamic drag is a quadratic model and the friction is a linear model.

$$\begin{aligned} F_{load} &= F_{aero} + R_x + F_g && (4) \\ F_{aero} &= \frac{1}{2} C_a \rho A \dot{x}^2 = c_a \dot{x}^2 && (5) \\ R_x &= N(\hat{c}_{r,0} + \hat{c}_{r,1}|\dot{x}| + \hat{c}_{r,2}\dot{x}^2) \approx c_{r,1}\dot{x} && (6) \\ F_g &= mg \sin\alpha && (7) \end{aligned}$$

Note that the absolute value is ignored for friction since the model is used for only forward motion ($\dot{x} \geq 0$).

The tire force is computed using the engine speed and wheel slip equations.

$$\omega_w = (GR)\omega_e \tag{8}$$

$$s = \frac{\omega_w r_e - \dot{x}}{\dot{x}} \tag{9}$$

$$F_x = \left\{ \begin{array}{ll} cs, & |s| < 1 \\ F_{max}, & \text{otherwise} \end{array} \right\} \tag{10}$$

Where ω_w is the wheel angular velocity and s is the slip ratio.

We setup the longitudinal model inside a Python class below. The vehicle begins with an initial velocity of 5 m/s and engine speed of 100 rad/s. All the relevant parameters are defined and like the bicycle model, a sampling time of 10ms is used for numerical integration.

In [7]:
```
import sys
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.image as mpimg

class Vehicle():
    def __init__(self):

        # ==================================
        #  Parameters
        # ==================================

        #Throttle to engine torque
        self.a_0 = 400
        self.a_1 = 0.1
        self.a_2 = -0.0002

        # Gear ratio, effective radius, mass + inertia
        self.GR = 0.35
        self.r_e = 0.3
        self.J_e = 10
        self.m = 2000
        self.g = 9.81

        # Aerodynamic and friction coefficients
        self.c_a = 1.36
        self.c_r1 = 0.01

        # Tire force
        self.c = 10000
        self.F_max = 10000

        # State variables
        self.x = 0
        self.v = 5
```

```
        self.a = 0
        self.w_e = 100
        self.w_e_dot = 0

        self.sample_time = 0.01

    def reset(self):
        # reset state variables
        self.x = 0
        self.v = 5
        self.a = 0
        self.w_e = 100
        self.w_e_dot = 0
```

Implement the combined engine dynamic equations along with the force equations in the cell below. The function *step* takes the throttle x_θ and incline angle α as inputs and performs numerical integration over one timestep to update the state variables. Hint: Integrate to find the current position, velocity, and engine speed first, then propagate those values into the set of equations.

```
In [8]: class Vehicle(Vehicle):
            def step(self, throttle, alpha):
                w_w = self.GR * self.w_e
                s = (w_w * self.r_e - self.v) / self.v
                if abs(s) < 1:
                    F_x = self.c * s
                else:
                    F_x = self.F_max
                F_aero = self.c_a * (self.v ** 2)
                R_x = self.c_r1 * self.v
                F_g = self.m * self.g * np.sin(alpha)
                F_load = R_x + F_aero + F_g
                self.a = (F_x - F_load) / self.m
                self.v += self.a * self.sample_time
                self.x += self.v * self.sample_time
                T_e = throttle * (self.a_0 + self.a_1 * self.w_e + self.a_2 * self.w_e ** 2)
                self.w_e_dot = (T_e - self.GR * self.r_e * F_load) / self.J_e
                self.w_e += self.w_e_dot * self.sample_time
                pass
```

Using the model, you can send constant throttle inputs to the vehicle in the cell below. You will observe that the velocity converges to a fixed value based on the throttle input due to the aerodynamic drag and tire force limit. A similar velocity profile can be seen by setting a negative incline angle α. In this case, gravity accelerates the vehicle to a terminal velocity where it is balanced by the drag force.

```
In [9]: sample_time = 0.01
        time_end = 100
        model = Vehicle()
```