Lab 3: Cruise Control Design and Automatic Parking using Lyapunov's Direct Method

Maxwell Cobar, Clifton Huett, and Carlos Montalvo

ME 494 – Nonlinear Systems Dynamics and Controls

March 8, 2020

1 Introduction:

There are various types of stability for the solutions of a dynamical system modeled by a differential equation. One of the most important types of these stabilities is the stability of the solution near an equilibrium point. An equilibrium point of a system is a state at which the system will tend towards and remains forever. Lyapunov theory of stability is used for determining stability of a dynamical system around an equilibrium point. Lyapunov theory of stability has two methods to solve for stability. The first method of Lyapunov theory develops the solution in series which is proven convergent within limits. The second method of Lyapunov theory uses a Lyapunov function which is a scalar function to prove stability. The Lyapunov function must be differentiable, locally positive definite, and the derivative of the function along all trajectories of the system must be locally negative semi-definite [2]. Lyapunov's direct method is shown to determine stability for both the cruise control and automatic parking designs for a hypothetical vehicle.

2 Math Model:

Before a cruise control and an automatic parking designs could be created, vehicle parameters for the hypothetical vehicle were needed for simulation. Parameters for the hypothetical vehicle were estimated from actual vehicle parameters and chosen as follows: Max torque(T_{max}) 175 lbft, Min torque(T_{min}) 100 lbft, gravity(g) 32.2ft/s^2, friction factor(μ) 0.1, coefficient of drag(C_D) 0.2, density of air(ρ) 0.00238 slug/ft^3, frontal area(S) 19.3 ft^2, mass(m) 2000 lb. With the parameters accurately estimated, a free body diagram (FBD) was drawn for the vehicle, and using the FBD, equations of motion were derived for the vehicle.

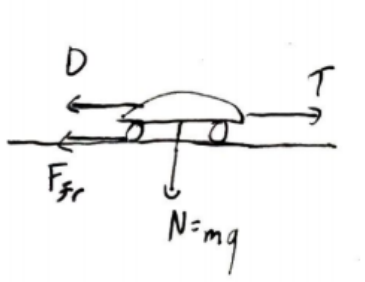

Figure 1: Free Body Diagram of Vehicle

From the FBD, the total force (Eq. 1) acting on the hypothetical vehicle comes from the thrust(T), drag(D), and friction force(F_{fr}).

$$F = T-D-F_{fr} \qquad (1)$$

To calculate the total force, the drag is calculated using Eq. 2,

$$D = 0.5*\rho*v^2*S*C_D \qquad (2)$$

Where v is the velocity of the vehicle. After finding the drag, the friction force is calculated using Eq. 3,

$$F_{fr} = \mu * N * sign(\dot{x}) \qquad (3)$$

Where N is the normal force of the car and sign(ẋ) is the sign function of the velocity. Finally, the thrust is calculated; however, the engine dynamics are taken into effect while solving for the thrust. As so, the thrust is found from the torque applied to the wheels divided by the wheel radius(r). The torque applied to the wheels was found using a power curve for the hypothetical vehicle and using the equation of the power curve (Eq. 4) to calculate the thrust.

$$T_w = \delta_T * T_{max} * e^{-\sigma|v|} \qquad (4)$$

Where σ is the speed of decay for the power curve and δ_T is either 0 or 1 depending on if the torque is going to the wheels. To accurately control the vehicle, the applied torque equation was substituted into the thrust equation to model all the vehicle dynamics (Eq. 5).

$$T = T_{max} - T_{min}\left[\frac{e^{\sigma x}}{1+e^{\sigma x}} - \frac{1}{2}\right] + 0.5(T_{max} + T_{min})\frac{\delta_T}{rm} \qquad (5)$$

Substituting the thrust, friction force, and drag equations into the total force equation, the cruise control design was solved as a function of velocity using simulation, and the automatic parking design was solved as a function of position using simulation.

3 Simulation Results:

To design the cruise control system as a function of velocity for the hypothetical vehicle, the available torque needed to be plotted against the velocity of the vehicle. Fig. 2 shows this plot for the hypothetical vehicle. For the available torque versus the velocity, the maximum torque was set at 175 ft-lbs, the minimum torque was set at 100 ft-lbs, and the speed of decay was set at 0.1. These parameters were plugged into Eq. 4 with velocity changing from 0 to 80 mph.

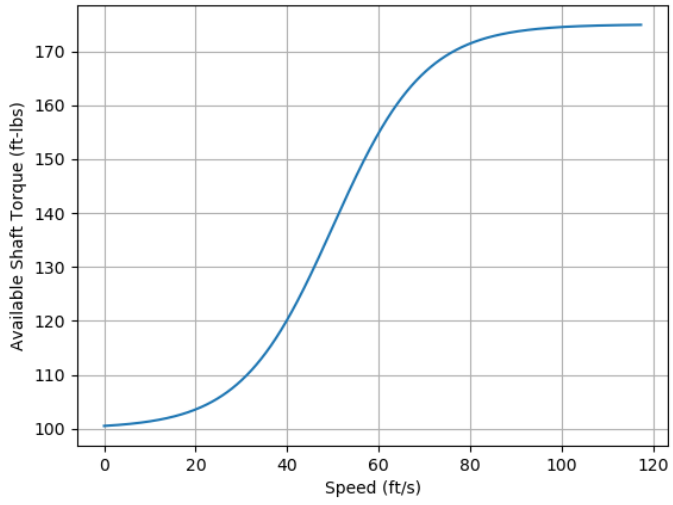

Figure 2: Cruise Control Graph of Velocity vs Available Torque

With the available torque calculated, the thrust is found, and the total force is calculated. With the total force, the Lyapunov function is created using Lyapunov's direct method. The Lyapunov function is a

function of both time and velocity. The Lyapunov function is plotted in Fig. 3 where the x- and y-axis are velocity and time respectively and the z-axis is the value of the Lyapunov function. Since all values of the Lyapunov function are positive the hypothetical vehicle meets one condition for global stability.

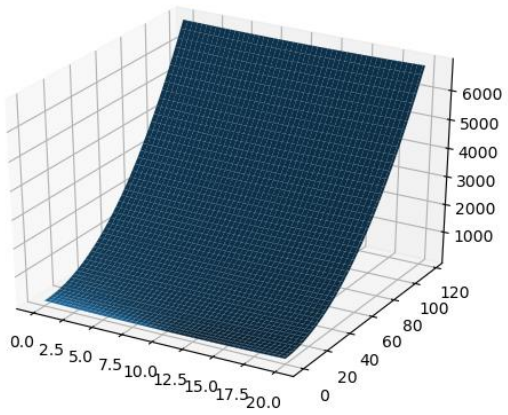

Figure 3: Lyapunov Method Showing All Positive

After the Lyapunov function is plotted, the derivative of the Lyapunov function is taken. With the derivative of the Lyapunov function, global stability can be determined if all values of the derivative are negative. The derivative of the Lyapunov function is plotted in Fig. 4 where the x- and y-axis are velocity and time respectively and the z-axis is the value of the Lyapunov function. Since all values of the derivative of the Lyapunov function are negative the hypothetical vehicle is globally stable.

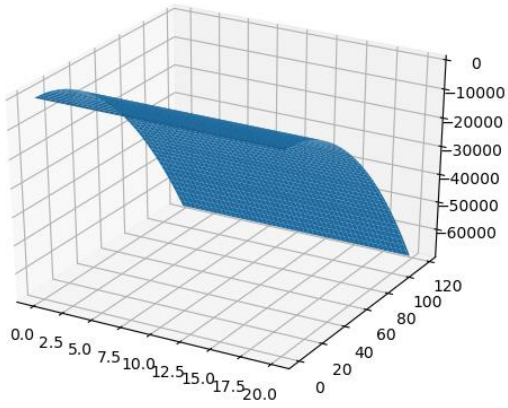

Figure 4: Derivative of Lyapunov Showing All Negative

To design the automatic parking system as a function of position for the hypothetical vehicle, the position of the vehicle needed to be plotted against time. Fig. 5 shows the position versus time graph of the automatic parking system. From Fig. 5, the hypothetical vehicle is shown to approach the parking spot

located 10 ft away. The vehicle does not overshoot the spot and does not have any steady state error. The automatic parking system is critically damped and stable and has a settling time of 20 seconds.

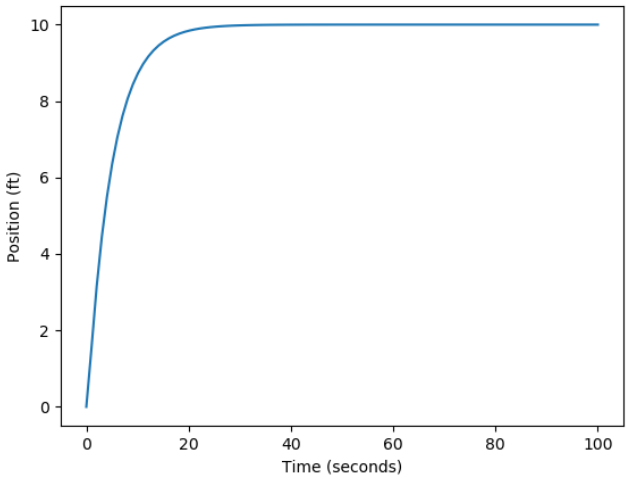

Figure 5: Automatic Parking Graph of Position vs Time

4 Conclusion:

One of the most important types of these stabilities is the stability of the solution near an equilibrium point. Lyapunov theory of stability is used for determining stability of a dynamical system around an equilibrium point. Lyapunov's direct method is shown to determine stability for both the cruise control and automatic parking designs for a hypothetical vehicle. The hypothetical vehicle is shown to be both stable for the cruise control and automatic parking systems. The cruise control system had all negative values for the derivative of the Lyapunov function and all positive values for the Lyapunov function proving stability. The automatic parking system had no steady state error or overshoot and was critically damped and stable.

5 Acknowledgements:

We would like to acknowledge Dr. Carlos Montalvo, James Helton, and Slater Dozier. Dr. Carlos Montalvo helped with the making of the python code used in this lab. James Helton and Slater Dozier helped to answer some calculation and coding questions that came up during the lab.

6 Appendix:

6.1 Code for Lyapunov Method for Automatic Parking and Cruise Control:

```
import matplotlib.pyplot as plt

import numpy as np

import scipy.integrate as integrate

from mpl_toolkits.mplot3d import Axes3D

g = 32.2

mu = 0.1
```

```
A = mu*g
Cd = 0.2
S = 19.3
rho = 0.00238
mass = 2000./g
B = 0.5*rho*S*Cd/mass
r = 1.0
C = 1/(r*mass)
def sat(xdot):
    eps = 0.1
    if xdot > eps:
        return 1
    elif xdot < eps:
        return - 1
    else:
        slope = 2/eps
        return slope*xdot
def Harry_Potter_Magic_Controller(state):
    x = state[0]
    xdot = state[1]
    k = 5.
    beta = -k*xdot
    dt = (1/(C*available_shaft_torque(xdot)))*(B*xdot**2 + A*sat(xdot) - (x-10) + beta)
    return dt
def Derivatives(state,t):
    x = state[0]
    xdot = state[1]
    dt = Harry_Potter_Magic_Controller(state)
    xddot = -A*sat(xdot) - B*xdot**2 + C*dt*available_shaft_torque(xdot)
    return [xdot,xddot]
```

```
def available_shaft_torque(xdot):
    tau_min = 100.0
    tau_max = 175.0
    sig = 0.1
    xhat = xdot - 50.0
    tau = (tau_max-tau_min)*(np.exp(sig*xhat)/(np.exp(sig*xhat)+1)-0.5) + (tau_max+tau_min)/2
    return tau
##Plot the maximum shaft torque for debugging
xdot = np.linspace(0,117.33,1000) #117.33 ft/s is 80 mph
tau = available_shaft_torque(xdot)
plt.figure()
plt.plot(xdot,tau)
plt.xlabel('Speed (ft/s)')
plt.ylabel('Available Shaft Torque (ft-lbs)')
plt.grid()
###Plot V
x = np.linspace(0,20,100)
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
xx, xxdot = np.meshgrid(x, xdot, sparse=False, indexing='ij')
V = 0.5*(xx-10)**2 + 0.5*xxdot**2
ax.plot_surface(xx,xxdot,V)
###Plot Vdot
Vdot = 0*V
rowctr = -1
colctr = 0
for xi in x:
    rowctr += 1
    colctr = 0
    for xdoti in xdot:
```

```
        state = [xi,xdoti]
        statedot  = Derivatives(state,0)
        xddoti = statedot[1]
        Vdot[rowctr][colctr] = (xi-10)*xdoti + xdoti*xddoti
        colctr += 1
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.plot_surface(xx,xxdot,Vdot)
##Simulate with alternating initial conditions
x0 = 0
xdot0 = 0
state0 = [x0,xdot0]
tout = np.linspace(0,100,100)
stateout = integrate.odeint(Derivatives,state0,tout)
xout = stateout[:,0]
plt.figure()
plt.plot(tout,xout)
plt.show()
```

6.2 References:

[1] Franklin, G. F., Powell, J. D., and Emami-Naeini, A., *Feedback control of dynamic systems*, Upper Saddle River, NJ: Pearson, 2020.

[2] Slotine, J.-J. E., and Li, W., *Applied nonlinear control*, Taipei: Prentice Education Taiwan Ltd., 2005.