Roderick Landreth
April 8, 2019
MER-322: Dynamics of Physical Systems
Professor Ramasubramanian

Automobile Suspension Modelling

Introduction

The purpose of this experiment is confirming that the method of state propagation yields the same result as modeling a given dynamic system using Simulink. This model is a suspension system, similar to that of a car. Both methods analyze how successfully this suspension system reduces the vertical acceleration and jerk of the driver by as much as possible.

Materials and Methods

The system being modeled is a mass suspended by a spring and damper, connected from a mass that contains a wheel running over obstacles on the ground. The system input is the ground terrain elevation over time, and the output is the movement of the sprung mass. We derived state equations from the forces acting on each mass (M_s and M_u), relating state derivatives to the spring constants (K) (modeling the wheel as a spring), the damper constant (B), and the vertical position and velocity of masses and the ground (x and $\dot{x}$). The vertical position of the ground is a single flat step that lasts one second, with a magnitude of 0.05m.

$$\ddot{x_3} = \frac{1}{Ms}\Big[K_s(x_2 - x_3) + B_s(\dot{x_2} - \dot{x_3})\Big] \tag{1}$$

$$\ddot{x_2} = \frac{1}{Mu}\Big[K_t(x_2 - x_3) - K_s(x_2 - x_3) - B_s(\dot{x_2} - \dot{x_3})\Big] \tag{2}$$

Simulink solved the differential State Equations 1 and 2 discretely within the program using a time step of $\Delta t = 0.001s$. State propagation in Python used Euler's method, iterating over small time steps to produce values of position and velocity for each time, which can then be used to find values of acceleration. If the time steps are too large, the estimate is too rough and over-corrects during subsequent time steps, eventually becoming an unstable and inaccurate solution.

Results

Figure 1 displays both solutions graphed on the same plot, confirming that state propagation and Simulink yielded the same results. The shape of the acceleration is as expected, including a positive large peak at t=1 as the wheel climbs the step, and a negative peak at t=2 as the wheel dismounts the peak. The acceleration's oscillation after mounting and dismounting the step is desirable, applying more acceleration and jerk to the passenger than the ideal suspension system, which would produce low maximum acceleration without subsequent oscillation.

Discussion

The benefits of Simulink include its visual representation of the system, quick production time and integration with Matlab. The State Propagation using Euler's method can be created without specific software in different programming languages (this case uses Python), meaning it is more easily accessible.

This model assumes the wheel is small, mounting the step quickly rather than a large diameter wheel slowly ascending. If there were no damper, the initial jump in acceleration would be much higher over a shorter time interval, causing more discomfort to the passenger. If the wheel were large in comparison to the bump, the increased time to mount the step would lower the maximum acceleration and increase comfort.

Attachment A: Plotting Model Solution

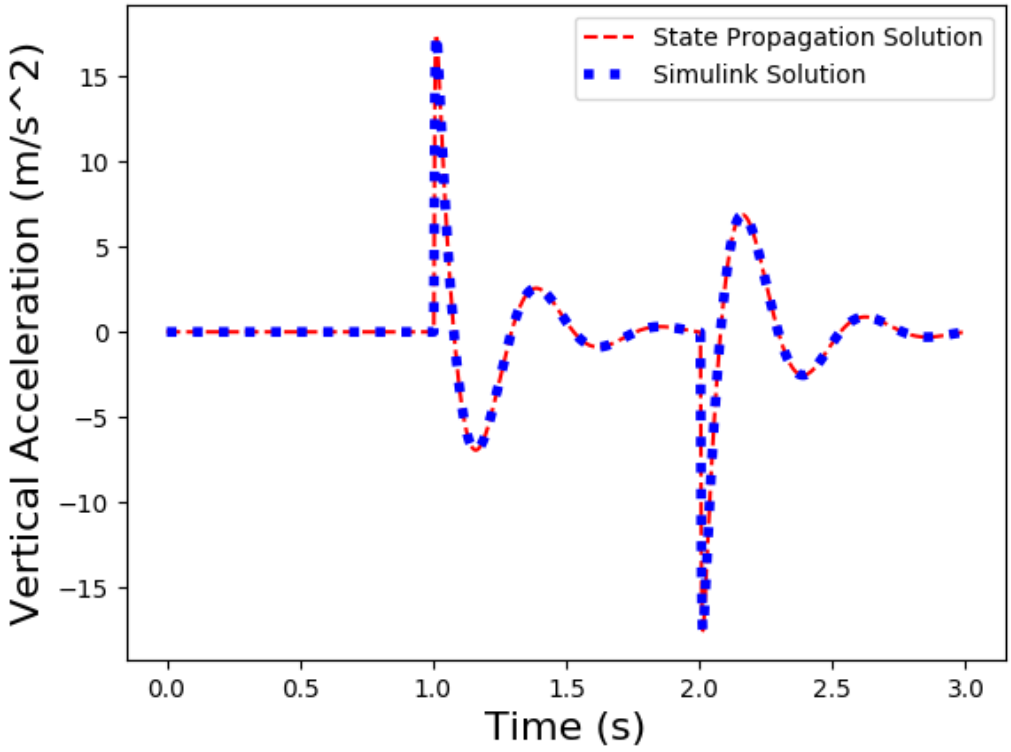

Figure 1: A Comparison of the Suspension System's Sprung Mass Vertical Acceleration Solutions generated by Euler's method of state propagation, and with Simulink. Solutions use Δt=0.001, a sprung mass of 300kg, an unsprung mass of 30kg, a spring with spring constant $K_s = 6000N/m$, a tire with stiffness $K_t = 12000N/m$, and a shock absorbed(damper) constant of $b_s = 6000Ns/m$

Attachment B: State Propagation Python Code

```
# State Propagation using Python
# Author: Roderick Landreth
# Date: April 9, 2019

import matplotlib.pyplot as plt
from io import BytesIO
from PIL import Image

""" Given system parameters, propagate the states,
returns the acceleration of the sprung mass."""
def propagate_suspention(ms,mu,ks,bs,kt,T,dt):
    # Initial Conditions
    x2=[0]
    x3=[0]
    v2=[0]
    v3=[0]
    x1=[0]
    a2=[0]
    a3=[0]
    t=[0]
    T=3
    dt=0.001
    numsteps = T / dt

    # Iterates over the Timestep
    for i in range(0,int(numsteps)):

        # This is the input, x1, the terrain the wheel passes over
        if (t[i] < 1):
            x1.append(0)
        elif(t[i] > 2):
            x1.append(0)
        else:
            x1.append(0.05)

        x2.append( x2[i] + v2[i] * dt)
        x3.append( x3[i] + v3[i] * dt)

        v2.append( v2[i] + a2[i] * dt)
        v3.append( v3[i] + a3[i] * dt)

        a2.append(1 / mu * (-ks * (x2[i] - x3[i]) - bs * (v2[i] - v3[i]) + kt * (x1[i] - x2[i])))
        a3.append((1 / ms) * (ks * (x2[i] - x3[i]) + bs * (v2[i] - v3[i])))

        t.append( t[i] + dt)
    return (a3,t)


'''This is to export the file for graphing in MatLab, because my desktop
```

```
is having troubles with installing external modules like matplotlib'''
def export_txt(lst,name):
    threePositionSolution = open("{}.txt".format(name), "w")
    threePositionSolution.write(str(lst))
    threePositionSolution.close()

'''Plot a list and name the axes'''
def sol_plot(x1,y1,label1,param_dict1,x2,y2,label2,param_dict2,x_axis,y_axis):
    fig, ax = plt.subplots()
    axis_font = {'size':16}
    ax.plot(x1,y1,**param_dict1)
    ax.plot(x2,y2,**param_dict2)
    ax.set_xlabel("{}".format(x_axis),**axis_font)
    ax.set_ylabel("{}".format(y_axis),**axis_font)
    plt.legend([label1,label2])
    plt.show()

    png1 = BytesIO()
    fig.savefig(png1, format='png')
    png2 = Image.open(png1)
    png2.save('State Propagation.tiff')
    png1.close()


# Converting the Simulink solution from a stripped ascii file to a float array
get_a3_Simulink = open("a3.txt","r")
a3_Simulink=[float(x) for x in get_a3_Simulink.read().split(',')]
get_a3_Simulink.close()
print(a3_Simulink)

State_Propagation_Solution = propagate_suspention(300,30,60000,6000,120000,3,0.001)
export_txt(State_Propagation_Solution,"State Propagation Solution")
sol_plot(State_Propagation_Solution[1],State_Propagation_Solution[0],
         "State Propagation Solution", {'color' : 'red', 'linestyle' : '--'},
         State_Propagation_Solution[1],a3_Simulink,"Simulink Solution",
         {'color' : 'blue', 'linestyle' : ':', 'linewidth':4},"Time (s)",
         "Vertical Acceleration (m/s^2)")
```