Vehicle Vibration Analysis with C++

Courtney Armstrong

MEC 510, Final Project

Outline

- Motivation
- Background
- Problem Description
- Code Structure
  - Overview
  - External Libraries
  - Classes and Functions
- Demonstration
- Results

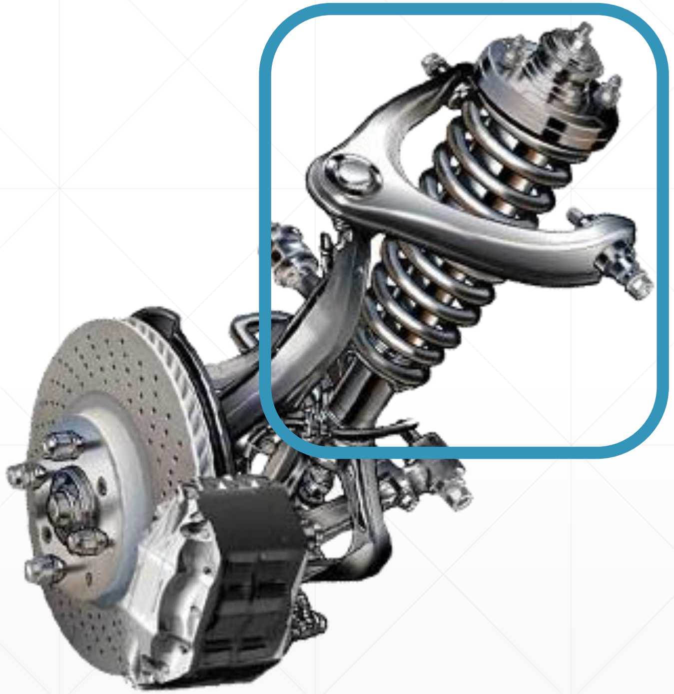
Problem Background & Description

Motivation

- MATLAB can be used to successfully analyze a vehicle system
- C++ analysis is much faster than MATLAB
  - MATLAB must run on CPU, C++ can be run using GPU for complex calculations via CUDA
- C++ also allows for direct comparison of multiple vehicle/road object combinations by simply defining more objects

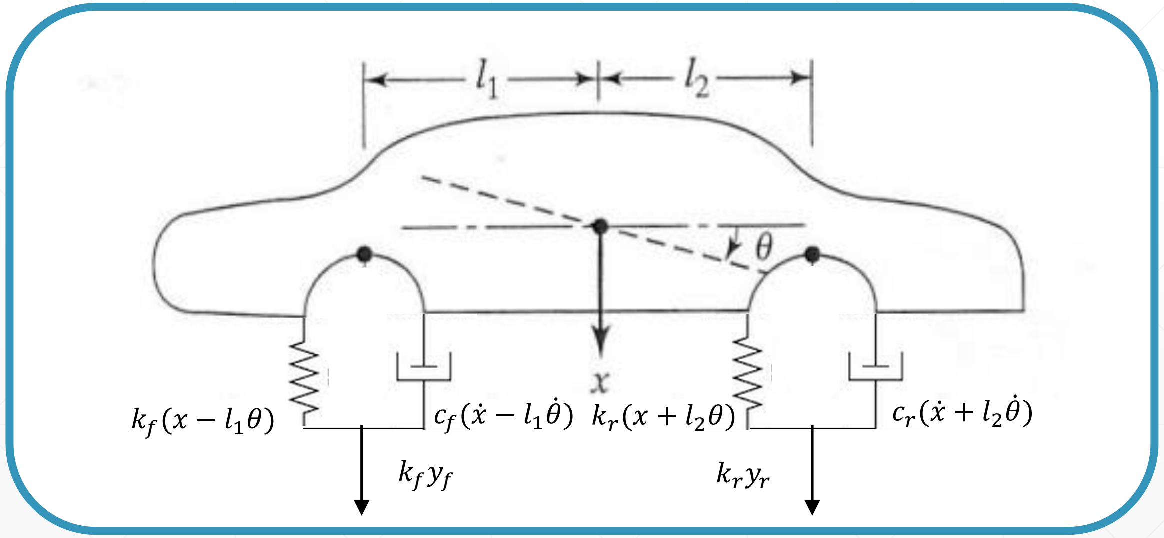
Vehicle Vibrations

- Suspension systems in vehicles are designed to reduce uncomfortable motions felt by passengers while driving
  - Utilizes springs and shock absorbers to control vibration transmitted through vehicle
- Focused on reducing two kinds of motion
  - Bounce (up and down motion)
  - Pitch (front to back rotation)



Vehicle Model

- Bounce (x) and pitch (θ) are independent of one another
  - Results in a two degree of freedom system with two coupled differential equations



Vehicle Model

- Mathematical equations governing bounce and pitch response derived from diagram

$$m\ddot{x} + (c_f + c_r)\dot{x} + (k_f + k_r)x + (c_r l_2 - c_f l_1)\dot{\theta} + (k_r l_2 - k_f l_1)\theta = k_f y_f(t) + k_r y_r(t)$$

$$J\ddot{\theta} + (c_f l_1^2 + c_r l_2^2)\dot{\theta} + (k_f l_1^2 + k_r l_2^2)\theta + (c_r l_2 - c_f l_1)\dot{x} + (k_r l_2 - k_f l_1) = k_r l_2 y_r(t) - k_f l_1 y_f(t)$$

- $y_f(t)$ and $y_r(t)$ are the forcing equations due to bumps on the road and vehicle speed

$$y_f = A sin\left(\frac{2\pi V}{L} t\right) \qquad y_r = A sin\left(\frac{2\pi V}{L} t - \frac{2\pi(l_1 + l_2)}{L}\right)$$

Project Description

- Understanding the vibration of the vehicle requires solving the equations of motion to determine x and θ
  - Only x, θ, dx, dθ and t are variable (all other variables in equation are constants)
- Using C++, along with external libraries, the equations can be solved based on user input of parameters

C++ Code Structure

Functionality Overview

1. User inputs car parameters or selects a vehicle class (based on SAE standards)
   - Mass, damping, inertia, etc.
2. User inputs road parameters
   - "Wavelength", speed, and amplitude
3. User inputs analysis parameters
   - Start time, end time, time step and initial values
4. Script outputs natural frequencies, mass and stiffness matrices and writes integration values to a text file

External Libraries

- Determining the solution for the coupled equations requires the use of linear algebra and differential equations
  - Outside the scope of what MSVS libraries can do natively
- Eigen
  - High level, template library for matrix and vector linear algebra
- Odeint
  - High level, template library for solving differential equations
  - Peer reviewed by Boost for accuracy

Vehicle Class

- Stores all required constants for left hand side of motion equation (physical parameters of car)
- Includes the functions required to determine:
  - Natural frequency (Eigen)
- Also includes functions to allow user to either define the vehicle parameters or pick pre-set parameters (SAE vehicle class)

Road Class

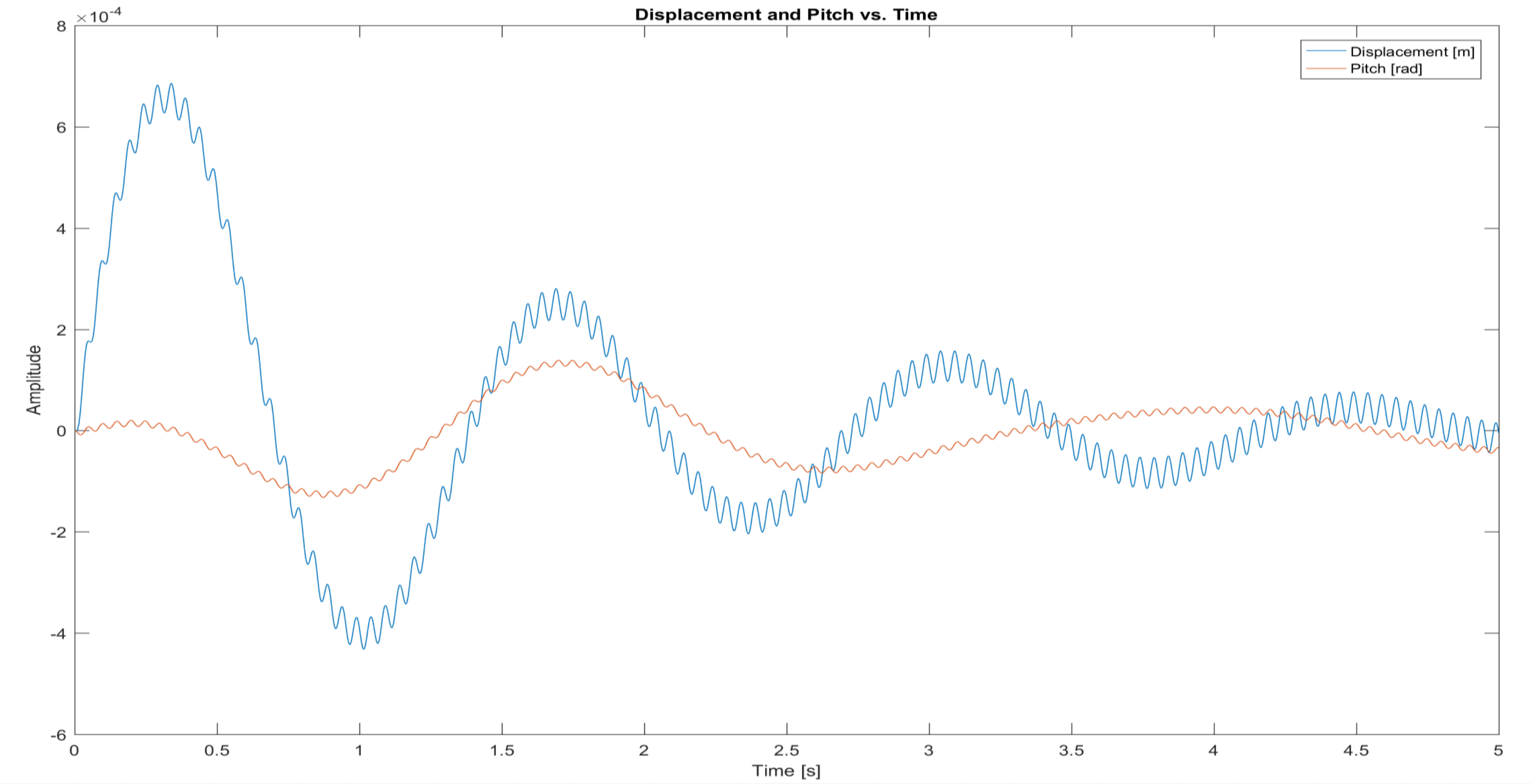
- Stores all required constants for forcing functions

Verification

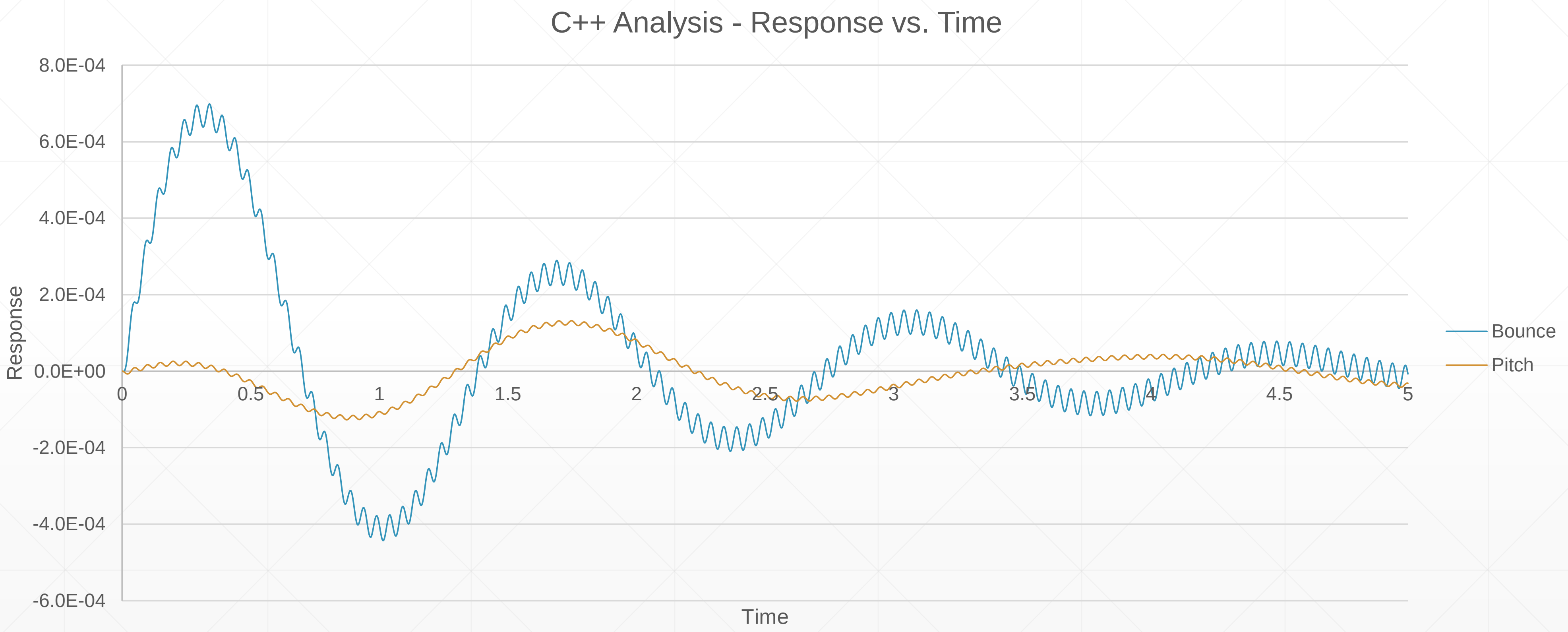
Verification Setup

- Vehicle Parameters
  - Class 4 SAE Vehicle (Luxury vehicle, Jaguar XJ)
- Road Parameters (moderately bumpy road)
  - 10 m/s velocity (approx. 22.4 mph)
  - 0.025 m peak amplitude (approx. 1”)
  - 0.5 m wavelength (approx. 20”)
- Time period
  - 0-5 seconds

MATLAB Results (ode45)



C++ Results (Bulirsch–Stoer algorithm)



Numerical Comparison

Maximum Response Values		
	Bounce (m)	*Pitch (rad)*
C++ Analysis	6.99×10^{-4}	1.32×10^{-4}
MATLAB Analysis	6.86×10^{-4}	1.39×10^{-4}

$$\text{Percent Error} \begin{cases} Bounce & -\ 0.18\% \\ Pitch & -\ 0.54\% \end{cases}$$

Error is within acceptable range, verification is successful!

Questions?