Lab 5: Validation Testing of a Suspension System

Jay Parmar

Spring 2025
ME 344L/ECE 382L Control Systems

1 Spring Memo

1.1 Step Response Criteria: Overshoot and Settling Time

In analyzing the springs, the first criteria to consider are overshoot and settling time. As shown in Figure 1, with a voltage input of 1V, Springs A, B, and C overshot by 17%, 23%, and 33%, respectively. Regarding settling time, Figure 1 shows that A, B, and C had settling times of about 1.15s, 1.2s, and 1.1s. These are important criteria as the car cannot have too high of an overshoot after hitting a bump but also cannot continue shaking for a while after the bump has ben passed. After observation and calculation of overshoot, spring C was ruled out as a viable option.

1.2 Frequency Response

Figures 2 and 3 show the response of springs A and B to a sinusoidal input at a range of frequencies. All frequencies have similar amplitudes of response, with varying periods of oscillation. To decide which of A and B was viable, we took a look at the Bode plots of the frequency response for each spring. In analyzing both A and B's frequency response (Figure 4), we can see that A reacts entirely within the 1- 7 Hz range with a max of 15 dB at 1.5 Hz. Spring B's range of reactive frequencies extends beyond the 7 Hz testing range, with a peak of 20 dB at 6 Hz.

When choosing between these two, though both work for our purposes, we would likely choose Spring A because we can model its entire behavior within the 1 - 7 Hz range, meaning there is no uncertainty. In addition, it has a lower max response at a lower frequency, making it less volatile.

1.3 Conclusion

To maximize comfort, we chose to work with Spring A. It has a low overshoot and moderate settling time when the car hits a bump. In addition, it has a predictable frequency that has the lowest response among the three springs. In order to maximize comfort, we would recommend using spring A.

2 Appendix

2.1 Collected Data

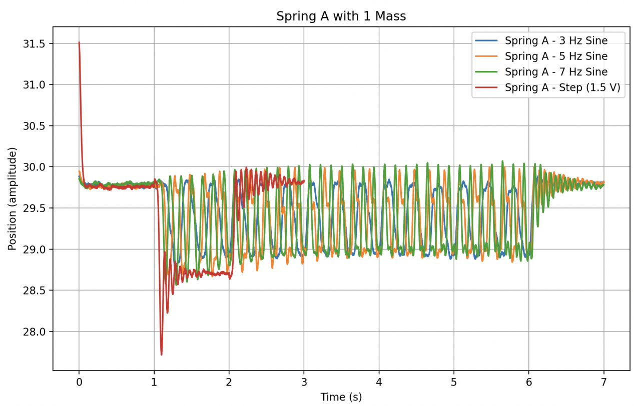

Figure 1: Spring A Results

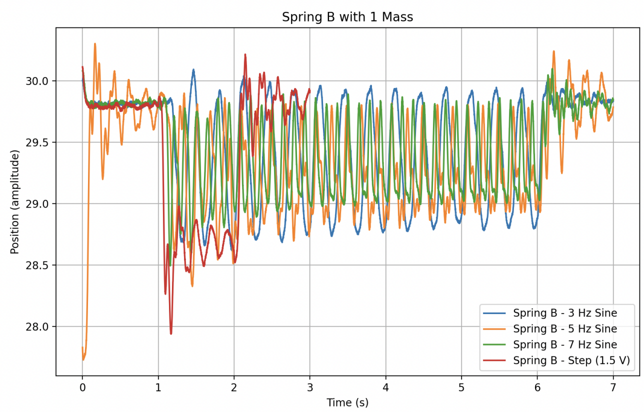

Figure 2: Spring B Results

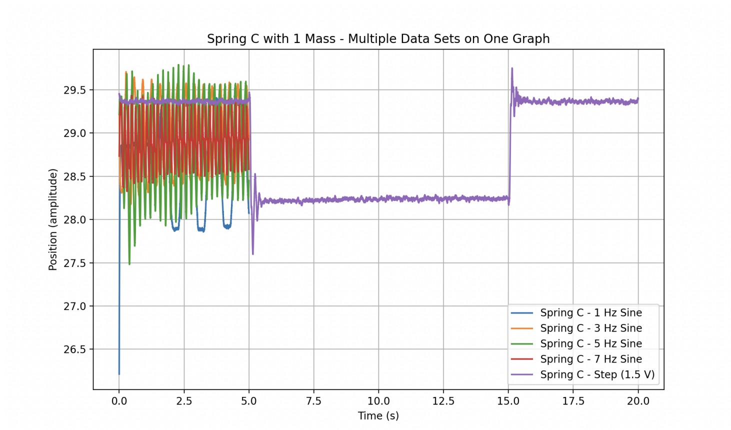

Figure 3: Spring C Results

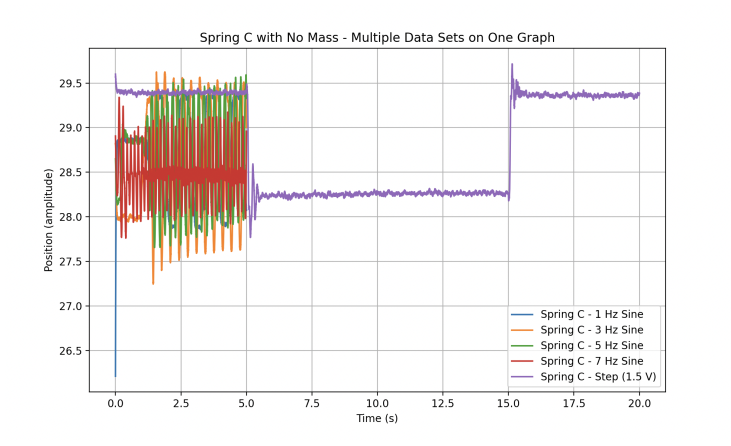

Figure 4: Spring C No Mass Results

2.2 Spring Responses: Step Input

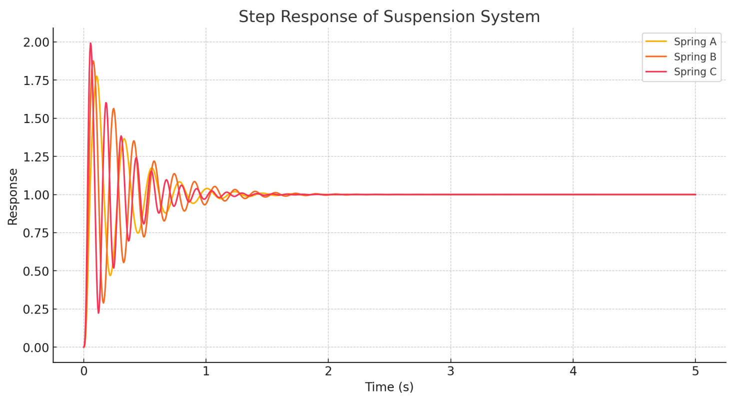

Figure 5: Spring Step Responses

2.3 Spring Responses: Sinusoidal Input

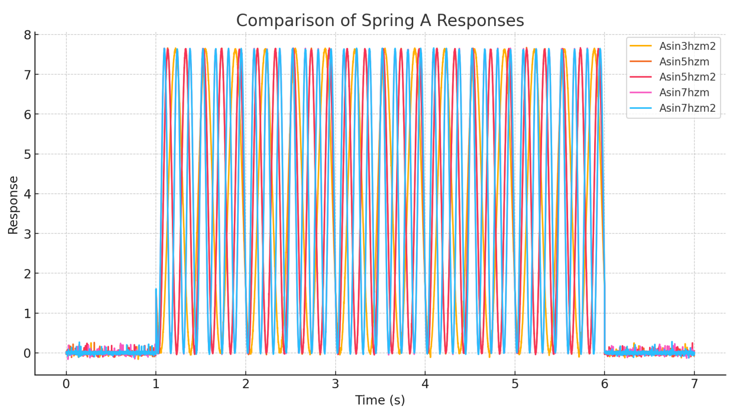

Figure 6: Spring A Sin Responses

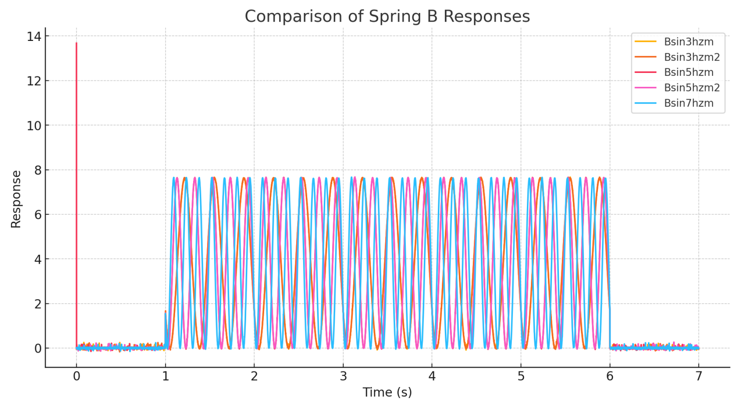

Figure 7: Spring B Sin Responses

2.4 Frequency Change Responses

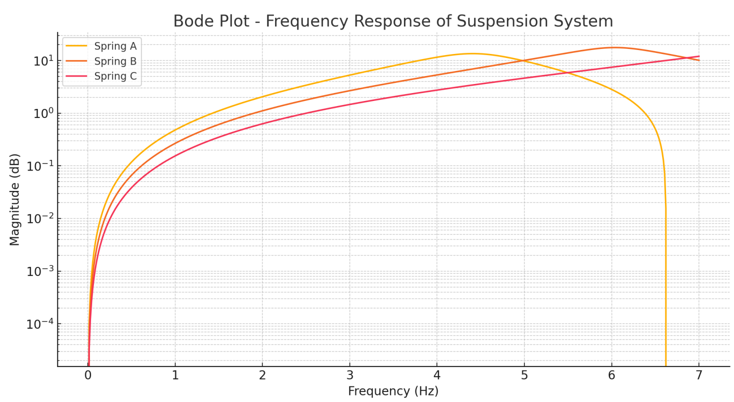

Figure 8: Suspension System Frequency Response

2.5 Bode Plot Code:

```python
import scipy.signal as signal
import numpy as np
import matplotlib.pyplot as plt

m1 = 0.9
m2 = 0.875
k2 = 8870
b2 = 21.4

springs = {
    "Spring A": {"k1": 770, "b1": 6.9},
    "Spring B": {"k1": 1576, "b1": 6.80},
    "Spring C": {"k1": 3642, "b1": 11.62},
}

freqs_hz = np.linspace(0, 7, 500)
freqs_rad = 2 * np.pi * freqs_hz

plt.figure(figsize=(12, 6))

for spring, params in springs.items():
    k1 = params["k1"]
    b1 = params["b1"]


    num = [b1 * b2, k1 * b2 + k2 * b1, k1 * k2]
    den = [
        m1 * m2,
        m1 * b1 + m1 * b2 + m2 * b1,
        m1 * k1 + m1 * k2 + k1 * m2 + b1 * b2,
        b1 * k2 + k1 * b2,
        k1 * k2
    ]

    system = signal.TransferFunction(num, den)

    w, mag, phase = signal.bode(system, freqs_rad)

    plt.semilogy(w / (2 * np.pi), mag, label=spring)

plt.xlabel("Frequency (Hz)")
plt.ylabel("Magnitude (dB)")
plt.title("Bode Plot - Frequency Response of Suspension System")
plt.legend()
plt.grid(which="both", linestyle="--")
plt.show()
```

2.6 Step Response Code:

```
import scipy.signal as signal
import numpy as np
import matplotlib.pyplot as plt

m1 = 0.9
m2 = 0.875
k2 = 8870
b2 = 21.4

springs = {
    "Spring A": {"k1": 770, "b1": 6.9},
    "Spring B": {"k1": 1576, "b1": 6.80},
    "Spring C": {"k1": 3642, "b1": 11.62},
}

t = np.linspace(0, 5, 1000)
plt.figure(figsize=(12, 6))

for spring, params in springs.items():
    k1 = params["k1"]
    b1 = params["b1"]

    num = [b1 * b2, k1 * b2 + k2 * b1, k1 * k2]
    den = [
        m1 * m2,
        m1 * b1 + m1 * b2 + m2 * b1,
        m1 * k1 + m1 * k2 + k1 * m2 + b1 * b2,
        b1 * k2 + k1 * b2,
        k1 * k2
    ]

    system = signal.TransferFunction(num, den)
    t_out, response = signal.step(system, T=t)
    plt.plot(t_out, response, label=spring)

plt.xlabel("Time (s)")
plt.ylabel("Response")
plt.title("Step Response of Suspension System")
plt.legend()
plt.grid(True)
plt.show()
```

2.7 Step Response and Overshoot Analysis Code:

```
import numpy as np
import pandas as pd
```

```python
import os
import scipy.signal as signal
import matplotlib.pyplot as plt

m1 = 0.9
m2 = 0.875
k2 = 8870
b2 = 21.4

springs = {
    "Spring A": {"k1": 770, "b1": 6.9},
    "Spring B": {"k1": 1576, "b1": 6.80},
    "Spring C": {"k1": 3642, "b1": 11.62},
}

def analyze_experimental_response(time, response):
    steady_state_value = np.median(response[-50:])  # Approximate steady-state using median
    peak_value = np.max(response)
    overshoot = ((peak_value - steady_state_value) / steady_state_value) * 100 if steady_state_value != 0 else np.nan
    upper_bound, lower_bound = steady_state_value * 1.05, steady_state_value * 0.95
    within_bounds = np.where((response > lower_bound) & (response < upper_bound))[0]
    settling_time = time[within_bounds[-1]] if len(within_bounds) > 0 else np.nan
    return overshoot, settling_time

def filter_time_range(time, response):
    mask = (time > 1) & (time < 6)  # Keep only data between 1s and 6s
    return time[mask], response[mask]

extract_path = "./Fw_lab_5_extracted"
extracted_files = os.listdir(extract_path)
experimental_results = []

for file in extracted_files:
    file_path = os.path.join(extract_path, file)
    df = pd.read_csv(file_path, header=None)
    time, response = df[0], df[1]
    time_filtered, response_filtered = filter_time_range(time, response)
    overshoot, settling_time = analyze_experimental_response(time_filtered, response_filtered)
    experimental_results.append({"File": file, "Overshoot (%)": overshoot, "Settling Time (s)": settling_time})

```

```python
df_experimental_results = pd.DataFrame(experimental_results)

spring_experimental_performance = {"Spring A": [], "Spring B": [], "Spring C": []}
for _, row in df_experimental_results.iterrows():
    for spring in spring_experimental_performance:
        if row["File"].startswith(spring[-1]):  # Match last letter A/B/C
            spring_experimental_performance[spring].append((row["Overshoot (%)"], row["Settling Time (s)"]))

spring_experimental_summary = []
for spring, data in spring_experimental_performance.items():
    if data:
        avg_overshoot = np.nanmean([d[0] for d in data])
        avg_settling_time = np.nanmean([d[1] for d in data])
        spring_experimental_summary.append({"Spring": spring, "Avg Overshoot (%)": avg_overshoot, "Avg Settling Time (s)": avg_settling_time})

df_experimental_summary = pd.DataFrame(spring_experimental_summary)

t = np.linspace(0, 5, 1000)
plt.figure(figsize=(12, 6))

for spring, params in springs.items():
    k1, b1 = params["k1"], params["b1"]
    num = [b1 * b2, k1 * b2 + k2 * b1, k1 * k2]
    den = [m1 * m2, m1 * b1 + m1 * b2 + m2 * b1, m1 * k1 + m1 * k2 + k1 * m2 + b1 * b2, b1 * k2 + k1 * b2, k1 * k2]
    system = signal.TransferFunction(num, den)
    t_out, response = signal.step(system, T=t)
    plt.plot(t_out, response, label=spring)

plt.xlabel("Time (s)")
plt.ylabel("Response")
plt.title("Step Response of Suspension System")
plt.legend()
plt.grid(True)
plt.show()

print("Experimental Performance Summary:\n", df_experimental_summary)
```

3 Authorship

All team members helped collect data. Christian Carbeau wrote all the code and plotted the data. Winslow Griffen formatted the document and wrote the memo. Jay Parmar edited the document and completed the experimental procedure section.

4 Acknowledgments

We would like to acknowledge Dr. Michael Gustafson for providing the template used in this lab. Likewise, we want to acknowledge Pat McGuire for his assistance during this lab and all group members for their participation in these experiments.