-2: The Volatile Cargo – Semi-Active Suspension Control

Name: sandeep Kumar dhaka
Roll number: 25119046
Year : 1st Year
College: IIT Roorkee

Problem Description

The objective of this project is to design a semi-active suspension controller for a quarter-car model. The model represents the vertical dynamics of a vehicle by considering two masses: the sprung mass, which corresponds to the vehicle body, and the unsprung mass, which represents the wheel assembly. Different road profiles introduce disturbances that cause vibrations in the vehicle body, negatively affecting ride comfort and cargo stability.

The main goal of the controller is to reduce the displacement of the sprung mass and to minimize jerk, which is the rate of change of acceleration. By controlling these quantities, the vehicle motion becomes smoother and passenger comfort is improved. The controller operates in real time using only permitted system signals and adjusts the damping characteristics accordingly

Model and Controller Description

A quarter-car suspension model is used in this project, consisting of a sprung mass and an unsprung mass connected through a spring and a damper. The unsprung mass is also connected to the road through a tire stiffness element, allowing the model to capture the effects of road irregularities on vehicle dynamics.

The controller functions as a semi-active damper, where the damping coefficient C(t) is varied over time. The controller uses sprung mass acceleration and unsprung mass acceleration as inputs, which are computed from the system dynamics during simulation. The damping coefficient is constrained within the limits Cmin and Cmax to ensure physical feasibility. Additionally, a four time-step actuator delay is incorporated into the system to model realistic actuator behavior.

7:31

kaggle

SANDEEP KUMAR DHAKA 25119046 · 1H AGO · 20 VIEWS

notebookb517c345c6

Edit

0

Notebook Input Output Logs

Competition Notebook

The Volatile Cargo - Syn...

Public Score	Best Score
45.96638	45.96638 V1

In [1]:

```
# This Python 3 environment comes with many helpful analytics libraries installed
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/docker-python
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will l
```

7:32

kaggle

‹ ut Logs Comments (0) Setti ›

```
ll list all files under the input di
rectory

import os
for dirname, _, filenames in os.wal
k('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname,
filename))

# You can write up to 20GB to the cu
rrent directory (/kaggle/working/) t
hat gets preserved as output when yo
u create a version using "Save & Run
All"
# You can also write temporary files
to /kaggle/temp/, but they won't be
saved outside of the current session
```

7:32

kaggle

ut Logs Comments (0) Setti

In [2]:

```
# ======================================
===
# PS-2 FINAL WORKING CODE (NO PATH E
RROR)
# ======================================
===

import numpy as np
import pandas as pd
import os

# -------------------------------
# AUTO FIND road_profiles.csv
# -------------------------------
csv_path = None
for root, dirs, files in os.walk("/
kaggle/input"):
    if "road_profiles.csv" in file
```

```python
print("FOUND DATASET AT:", csv_path)

# ----------------------------
# LOAD DATA
# ----------------------------
data = pd.read_csv(csv_path)
time = data["t"].values

# ----------------------------
# PARAMETERS
# ----------------------------
m_s = 290.0
m_u = 59.0
k_s = 16000.0
k_t = 190000.0

c_min = 300.0
```

7:52 m

kaggle

Notebook Input Output Logs

```
# ----------------------------
# UTILITY
# ----------------------------
def RMS(x):
    return np.sqrt(np.mean(x ** 2))

# ----------------------------
# CONTROLLER (LEGAL)
# ----------------------------
def controller(a_s, a_u):
    c = 900 * abs(a_s) + 250 * abs(a_u)
    return np.clip(c, c_min, c_max)

# ----------------------------
# SIMULATION
# ----------------------------
def simulate(road):
    z_s = np.zeros(N)
```

7:53 m

kaggle

lotebook Input Output Logs (

```
    for k in range(1, N):
        c_cmd = controller(a_s[k-1], a_u[k-1])
        buffer.append(c_cmd)
        c = buffer.pop(0)

        a_s[k] = (-k_s * (z_s[k-1] - z_u[k-1])
                  - c * (v_s[k-1] - v_u[k-1])) / m_s

        a_u[k] = ( k_s * (z_s[k-1] - z_u[k-1])
                  + c * (v_s[k-1] - v_u[k-1])
                  - k_t * (z_u[k-1] - road[k])) / m_u

        v_s[k] = v_s[k-1] + a_s[k] *
```

7:53 m

kaggle

Notebook Input Output Logs

```
    return z_s, a_s

# ------------------------------
# METRICS
# ------------------------------
def compute_metrics(z_s, a_s):
    z_rel = z_s - z_s[0]

    rms_zs = RMS(z_rel)
    max_zs = np.max(np.abs(z_rel))

    jerk = np.zeros_like(a_s)
    jerk[1:] = (a_s[1:] - a_s[:-1]) /
dt

    rms_jerk = RMS(jerk)
    jerk_max = np.max(np.abs(jerk))

    comfort = (
```

7:53 m

kaggle

lotebook Input Output Logs

```
# ------------------------------
# RUN ALL PROFILES
# ------------------------------
rows = []

for i in range(1, 6):
    road = data[f"profile_{i}"].value
s
    z_s, a_s = simulate(road)

    rms_zs, max_zs, rms_jerk, comfort
= compute_metrics(z_s, a_s)

    rows.append([
        f"profile_{i}",
        rms_zs,
        max_zs,
        rms_jerk,
        comfort
```

```
# ----------------------------
submission = pd.DataFrame(
    rows,
    columns=["profile", "rms_zs", "ma
x_zs", "rms_jerk", "comfort_score"]
)

submission.to_csv("submission.csv", i
ndex=False)

print("submission.csv GENERATED SUCCE
SSFULLY")
submission
```

```
FOUND DATASET AT: /kaggle/input/the-v
olatile-cargo-synapse-drive-ps-2/dat
a/road_profiles.csv
submission.csv GENERATED SUCCESSFULLY
```

```
# ----------------------------
submission = pd.DataFrame(
    rows,
    columns=["profile", "rms_zs", "ma
x_zs", "rms_jerk", "comfort_score"]
)

submission.to_csv("submission.csv", i
ndex=False)

print("submission.csv GENERATED SUCCE
SSFULLY")
submission
```

```
FOUND DATASET AT: /kaggle/input/the-v
olatile-cargo-synapse-drive-ps-2/dat
a/road_profiles.csv
submission.csv GENERATED SUCCESSFULLY
```

```
import matplotlib.pyplot as plt

# time vector (CSV se)
time = data["t"].values

for i in range(1, 6):
    road = data[f"profile_{i}"].values

    # simulate function se sprung mass displacement
    z_s, a_s = simulate(road)

    plt.figure(figsize=(8, 4))
    plt.plot(time, z_s)
    plt.xlabel("Time (seconds)")
    plt.ylabel("Displacement (meters)")
    plt.title(f"profile_{i}: z_s(t)")
    plt.grid(True)
    plt.tight_layout()
    plt.show()import matplotlib.pyplot as plt

# time vector (CSV se)
time = data["t"].values

for i in range(1, 6):
    road = data[f"profile_{i}"].values

    # simulate function se sprung mass displacement
    z_s, a_s = simulate(road)

    plt.figure(figsize=(8, 4))
    plt.plot(time, z_s)
    plt.xlabel("Time (seconds)")
    plt.ylabel("Displacement (meters)")
    plt.title(f"profile_{i}: z_s(t)")
    plt.grid(True)
    plt.tight_layout()
    plt.show()
```

Results and Conclusion

The simulation results show that the designed semi-active suspension controller effectively reduces the sprung mass displacement for all given road profiles. The vehicle body motion remains within acceptable limits, indicating improved ride comfort and stability.
The controller also successfully controls the jerk, resulting in smoother acceleration changes and reduced vibrations. Overall, the performance of the controller is stable and consistent across all road profiles, demonstrating its effectiveness in handling different road disturbances.