University of South-Eastern Norway

www.usn.no

IIA1420 2022

Assignment 1: Railroad predictive maintenance

Lars Rikard Rådstoga

Faculty of Technology, Natural Sciences and Maritime Sciences

Campus Porsgrunn

Contents

1 Problem 1: Load data

Data from the attached .csv file was read using a python library also called Csv. Furthermore, the data was plotted as per instructions given by the assignment, see figure 1.1. The upper figure displays the measured acceleration data in red and the simulated data in green. The measured signal is full of noise, but the simulated data looks to follow the general trend, or a moving average, of the measured data. The second plot is the difference of the data, measured minus simulated. The expected behavior of this plot, that being if the measured data reflected an ideal train travelling on an ideal set of train tracks, would be evenly distributed noise. Observing this plot as a whole, it is clear that this is not the case. Some parts of the plot, e.g. between 200 and 380 seconds looks to contain this even noise. But there are amplitude spikes at multiple places, most notably at ca. 190, 380 and 460 seconds. These spikes are likely due to turbulence.

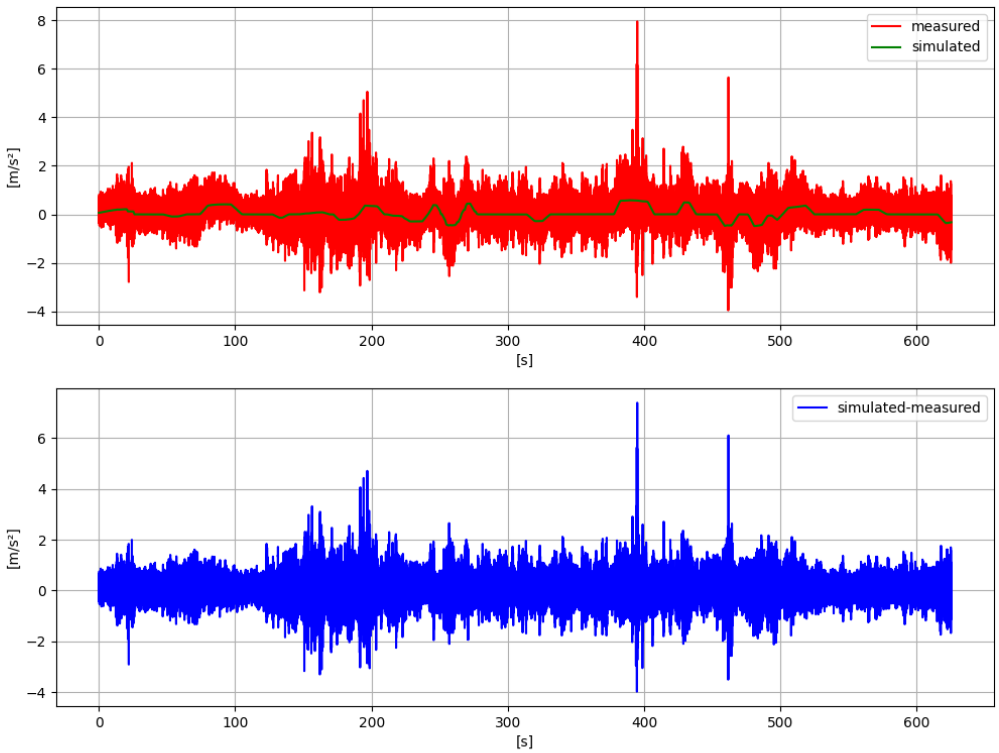

Figure 1.1: A plot of the data given in the assignment. The first plot shows both the measured and the simulated acceleration data. And the second shows the difference between the two.

2 Problem 2: FFT

Both the simulated and the measured data was transformed using the FFT algorithm with the python library Numpy. The plots in figure 2.1 display the results. The transform of the simulated data show very small amplitudes, and it is all close to 0 Hz. The measured signal, on the other hand, show both an even spread of amplitudes and bigger spikes as well. The even spread should represent the expected noise. The spikes, especially around 3 and 38 Hz, should tell us something about non-stochastic disturbances on the system. These could be related to imperfections on the tracks, wheels, engine, and more. My initial guess would be that the 38 Hz amplitude could be related to the wheels, according to [1] the average speed of trains in Norway is 80 km/h. Quick google searches don't supply much information of wheel radii of trains in Norway. However, another quick search says that a train moving at 72 km/h with a wheel radius of 1.2 m would have wheels with an RPM of about 160 [2]. This means the wheels could be rotating at 4 times the frequency compared to this disturbance, perhaps it is still related closer to the engine, due to gears. The 3 Hz disturbance could be related to different things, e.g. track joints in relation to lengths of the train cars. However, more information about the train and the track are needed to say for sure.

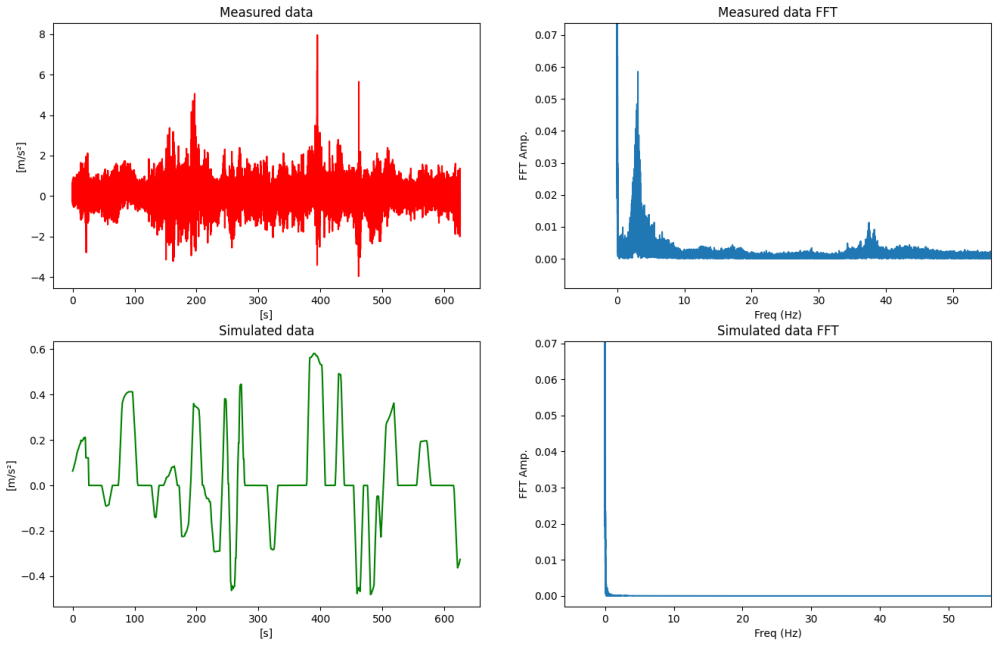

Figure 2.1: A plot of both the original data, measured and simulated data on the left and their respective FFT plots in the frequency domain on the right.

3 Problem 3: Comparing model with measurements

The difference between the measured and simulated data was transformed using the FFT algorithm and plotted in figure 3.1 together with the non-transformed data. There isn't any noticeable difference that I can point out.

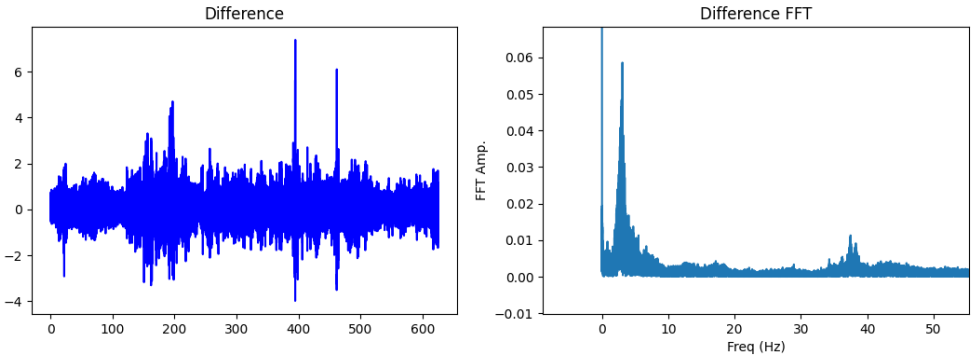

Figure 3.1: Plot of the difference in data: measured - simulated. Original data on the left and transformed using the FFT algorithm on the right.

References

[1] Wikipedia contributors, *High-speed rail in norway — Wikipedia, the free encyclopedia*, [Online; accessed 11-September-2022], 2022. [Online]. Available: https://en.wikipedia.org/w/index.php?title=High-speed_rail_in_Norway&oldid=1105760953.

[2] *Train speed 72 km / h how many revolutions per minute does a carriage wheel with a radius of 1.2 m? – univerkov*, https://www.univerkov.com/train-speed-72-km-h-how-many-revolutions-per-minute-does-a-carriage-wheel-with-a-radius-of-1-2-m/, (Accessed on 09/11/2022).

Appendix A

Source code

The source code, Jupyter notebook, used to load, transform and plot data.

In [ ]:
```
import matplotlib.pyplot as plt
%matplotlib widget
import numpy as np
import os
import pandas as pd
import scipy.fft
import csv
```

In [ ]:
```
# Load data
a_meas = []
a_sim = []
a_sim_minus_meas = []
path = 'data.csv'
with open(path, 'r') as f:
    reader = csv.reader(f, delimiter=';')
    headers = next(reader)
    for row in reader:
        a_meas.append(float(row[0]))
        a_sim.append(float(row[1]))
        a_sim_minus_meas.append(float(row[0])-float(row[1]))
```

In [ ]:
```
sample_rate = 500
dt = 1/sample_rate
sample_amount = len(a_meas)
time_array = np.arange(0, sample_amount/sample_rate,dt)
print(time_array[0])
print(time_array[-1])

sum = 0;
for i in range(len(a_meas)):
    sum += a_meas[i]*dt
print(sum)
```

In [ ]:
```
sample_rate = 500
dt = 1/sample_rate
sample_amount = len(a_meas)
time_array = np.arange(0, sample_amount/sample_rate,dt)
print(time_array[0])
print(time_array[-1])

sum = 0;
for i in range(len(a_meas)):
    sum += a_meas[i]*dt
print(sum)
```

```
0.0
625.394
84.26727999999062
```

In [ ]:
```
plt.close('all')
plt.figure(1, figsize=(12, 9))

plt.subplot(2, 1, 1)
plt.plot(time_array, a_meas, 'r', label='measured')
plt.plot(time_array, a_sim, 'g', label='simulated')
plt.legend()
plt.grid()
plt.xlabel('[s]')
```

```
plt.ylabel('[m/s²]')

""" plt.subplot(3, 1, 2)
plt.plot(time_array, a_sim, 'g', label='simulated')
plt.legend()
plt.grid()
plt.xlabel('[s]')
plt.ylabel('[m/s²]') """

plt.subplot(2, 1, 2)
plt.plot(time_array, a_sim_minus_meas, 'b', label='simulated-measured')
plt.legend()
plt.grid()
plt.xlabel('[s]')
plt.ylabel('[m/s²]')

"""
When we look at the first plot where the simulated data is drawn on top of the measu
we can see that the simulated data paints a good picture of the trend or moving aver
This means that the simulation probably paints the correct picture when the train tr
When subtracting the measured data from the simulated, we should end up with mostly
Since the result of the substraction is not random noise, but still contians some re
"""
```

Out[ ]:
```
' \nWhen we look at the first plot where the simulated data is drawn on top of the m
easured, \nwe can see that the simulated data paints a good picture of the trend or
moving average data from the measured.\nThis means that the simulation probably pain
ts the correct picture when the train tracks are in optimal conditions.\nWhen subtra
cting the measured data from the simulated, we should end up with mostly random nois
e if the simulation is correct.\nSince the result of the substraction is not random
noise, but still contians some residuals, it is possible these residuals are indicat
ors of non-optimal track conditions.\n'
```

In [ ]:
```
fourier_transform_meas = np.fft.fft(a_meas)
fourier_transform_sim = np.fft.fft(a_sim)
fourier_transform_sim_minus_meas = np.fft.fft(a_sim_minus_meas)
# Counting array [0,1,2,...,312 697]
transform_length = np.arange(len(fourier_transform_meas))
sampling_period = len(fourier_transform_meas)/(sample_rate)
fourier_frequency = transform_length/sampling_period

plt.close('all')
plt.figure(figsize=(16, 10))

# measured
plt.subplot(2, 2, 1)
plt.plot(time_array, a_meas, 'r', label='measured')
plt.xlabel('[s]')
plt.ylabel('[m/s²]')
plt.title('Measured data')
plt.grid()

plt.subplot(2, 2, 2)
plt.plot(fourier_frequency[:len(fourier_transform_meas)//2+1],
         2.0/sample_amount*np.abs(fourier_transform_meas[:len(fourier_transform_meas
plt.xlabel('Freq (Hz)')
plt.ylabel('FFT Amp.')
plt.title('Measured data FFT')
plt.grid()

# simulated
plt.subplot(2, 2, 3)
plt.plot(time_array, a_sim, 'g', label='simulated')
```

```
plt.xlabel('[s]')
plt.ylabel('[m/s²]')
plt.title('Simulated data')
plt.grid()

plt.subplot(2, 2, 4)
plt.plot(fourier_frequency[:len(fourier_transform_meas)//2+1],
         2.0/sample_amount*np.abs(fourier_transform_sim[:len(fourier_transform_meas)
plt.xlabel('Freq (Hz)')
plt.ylabel('FFT Amp.')
plt.title('Simulated data FFT')
plt.grid()

""" # difference
plt.subplot(3, 2, 5)
plt.plot(time_array, a_sim_minus_meas, 'b', label='simulated-measured')
plt.title('Difference')

plt.subplot(3, 2, 6)
plt.plot(fourier_frequency[:len(fourier_transform_sim_minus_meas)//2+1],
         2.0/sample_amount*np.abs(fourier_transform_sim_minus_meas[:len(fourier_tran
plt.xlabel('Freq (Hz)')
plt.ylabel('FFT Amp.')
plt.title('Difference FFT') """

plt.show()
```

In [ ]:

```
plt.close('all')
# measured
plt.subplot(3, 2, 1)
plt.plot(a_meas, 'r')

plt.subplot(3, 2, 2)
plt.plot(fourier_frequency[:len(fourier_transform_meas)//2+1],
         np.abs(fourier_transform_meas[:len(fourier_transform_meas)//2+1]))
plt.xlabel('Period (minute)')
plt.ylabel('FFT Amp.')

#Leakage effect?
# simulated
plt.subplot(3, 2, 3)
plt.plot(a_sim, 'g')

plt.subplot(3, 2, 4)
plt.plot(fourier_frequency[:len(fourier_transform_sim)//2+1],
         np.abs(fourier_transform_sim[:len(fourier_transform_sim)//2+1]))
plt.xlabel('Freq (Hz)')
plt.ylabel('FFT Amp.')
plt.show()
```

In [ ]:

```
plt.close('all')
plt.figure(figsize = (13, 4))
# difference
plt.subplot(1, 2, 1)
plt.plot(time_array, a_sim_minus_meas, 'b', label='simulated-measured')
plt.title('Difference')
plt.grid()

plt.subplot(1, 2, 2)
```

```
plt.plot(fourier_frequency[:len(fourier_transform_sim_minus_meas)//2+1],
         2.0/sample_amount*np.abs(fourier_transform_sim_minus_meas[:len(fourier_tran
plt.xlabel('Freq (Hz)')
plt.ylabel('FFT Amp.')
plt.title('Difference FFT')
plt.grid()
plt.show()

"""
comparing modeled and simulated data by subtraction, and using FFT on the difference
see a flat frequency spectrum, i.e., where the energy and therefore information cont
evenly distributed across the frequencies.  """
```

Out[ ]:
```
' \ncomparing modeled and simulated data by subtraction, and using FFT on the differ
ence, we would expect to \nsee a flat frequency spectrum, i.e., where the energy and
therefore information content is \nevenly distributed across the frequencies.  '
```