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.  '
```