Quinn Dougherty
Dr. Zekavat's 140 Lab
Week 4

Summary

This week we were given measurements of distance as a carriage fell to the ground. The measurements were taken by some mysterious device that has something to do with hertz. The task was to calculate average acceleration by first calculating all the pointwise accelerations, by first calculating all the pointwise velocities. We took our average acceleration and calculated the percent error with an accepted value for average acceleration, 980 cm/s/s. Then we took a slope of the velocity vs. time graph and found that it was about 10 cm/s/s off from our average acceleration.

Data Sheet

Experiment M-5 **Data Sheet**

Mass of carriage (m): 1237.7 g **Frequency of spark (f):** 30 Hz

Time per interval (Δt): 0.033 s $\frac{\Delta s}{0.033}$ $\frac{\Delta v}{0.033}$

Spot #	t	s	Δ s	v	Δ v	a
	sec	cm	cm	cm/s	cm/s	cm/s²
0	0	$S_0 = 0$				
1	0.033	$S_1 = 1.35$	$S_1 - S_0 = 1.35$	$V_1 = 40.9$	$V_2 - V_1 = 33.3$	$a_1 = 1009$
2	0.066	$S_2 = 3.8$	$S_2 - S_1 = 2.45$	$V_2 = 74.2$	28.788	872.36
3	0.099	$S_3 = 7.2$	3.4	103.03	36.364	1101.93
4	0.133	$S_4 = 11.8$	4.6	139.39	30.303	918.27
5	0.166	$S_5 = 17.4$	5.6	169.697	33.333	1010.10
6	0.199	$S_6 = 24.1$	6.7	203.03	33.333	1010.10
7	0.232	$S_7 = 31.9$	7.8	236.36	30.303	918.27
8	0.265	$S_8 = 40.7$	8.8	266.667	33.333	1010.10
9	0.298	$S_9 = 50.6$	9.9	300		

Average acceleration: 981.405 cm/s² **Percent error:** 0.14336 %

$\frac{\text{add all } a\text{'s}}{8}$ $100 \times \frac{980 - \text{average acceleration}}{980}$

Calculations

```python
from typing import Tuple
import pandas as pd
import numpy as np
import altair as alt

delta_t = 0.033

def mkdf(delta_t: float) -> Tuple[pd.DataFrame, pd.DataFrame, pd.DataFrame]:
  positions = pd.DataFrame({
      "t": [0, 0.033, 0.066, 0.099, 0.133, 0.166, 0.199, 0.232, 0.265, 0.298],
      "s": [0, 1.35,  3.8,   7.2,   11.8,  17.4,  24.1,  31.9,  40.7,  50.6]
  })

  velocity = pd.DataFrame({
      "delta_s": [position.s[k+1] - position.s[k] for k in range(position.shape[0] - 1)]
  })
  velocity = velocity.assign(v=velocity.delta_s / delta_t)

  acceleration = pd.DataFrame({
      "delta_v": [velocity.v[k+1] - velocity.v[k] for k in range(velocity.shape[0] - 1)]
  })
  acceleration = acceleration.assign(a=acceleration.delta_v / delta_t)

  return position, velocity, acceleration
```

```python
[39]
avg_accel = acceleration.a.sum() / acceleration.shape[0]
percent_error = abs(100 * (980 - avg_accel) / 980)
```

```python
[40]
avg_accel
```

```
981.4049586776857
```

```python
[44]
percent_error
```

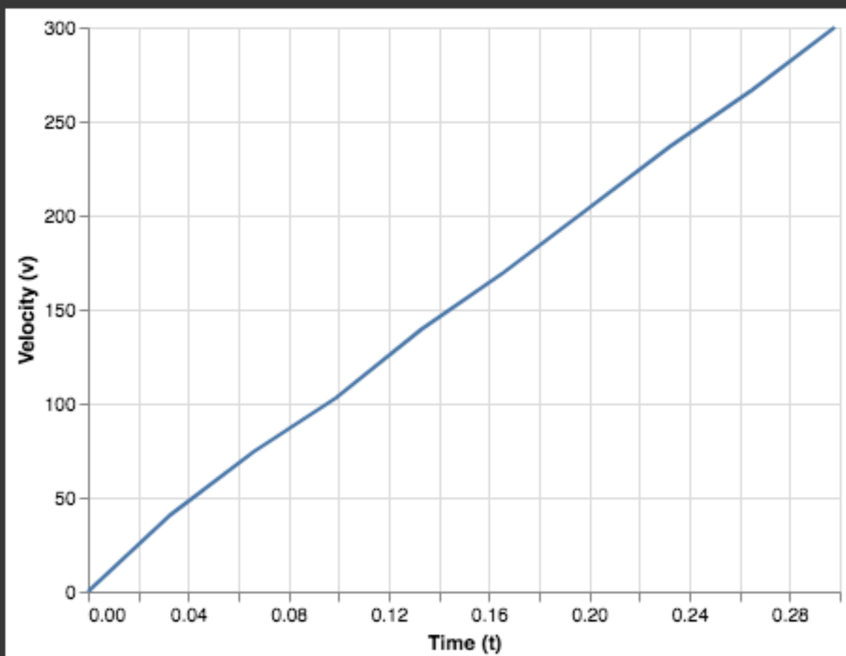
```
0.14336313037609458
```

Analysis & Conclusion

It was fun getting a sense of how velocity and acceleration is calculated under my fingernails. It is disappointing to do it during Covid because I would have liked to see the measuring instrument in action. I was surprised that the slope of the line of best fit wasn't closer to the average acceleration. In conclusion, position, velocity, and acceleration are mathematically related by a chain of derivatives. We can approximate this chain of derivatives by taking pointwise differences and dividing by delta t. Using a linear regression, the slope of the line of best fit of the velocity points is within 1% of the pointwise-calculated average acceleration.

Velocity vs. Time Graph

```python
velocity_prime = pd.DataFrame({"delta_s": [np.nan], "v": [0]})
alt.Chart(
    position.join(pd.concat((velocity_prime, velocity)).reset_index())
).mark_line(
).encode(
    x=alt.X("t", title="Time (t)"), y=alt.Y("v", title="Velocity (v)")
)
```



[70]

```python
from sklearn.linear_model import LinearRegression
model = LinearRegression()

model.fit(v_vs_t.t.values.reshape(-1, 1), v_vs_t.v)

print(f"Slope of line of best fit: {model.coef_[0]}")
```

```
Slope of line of best fit: 991.2646120415806
```