Physics 2130: Lab 1

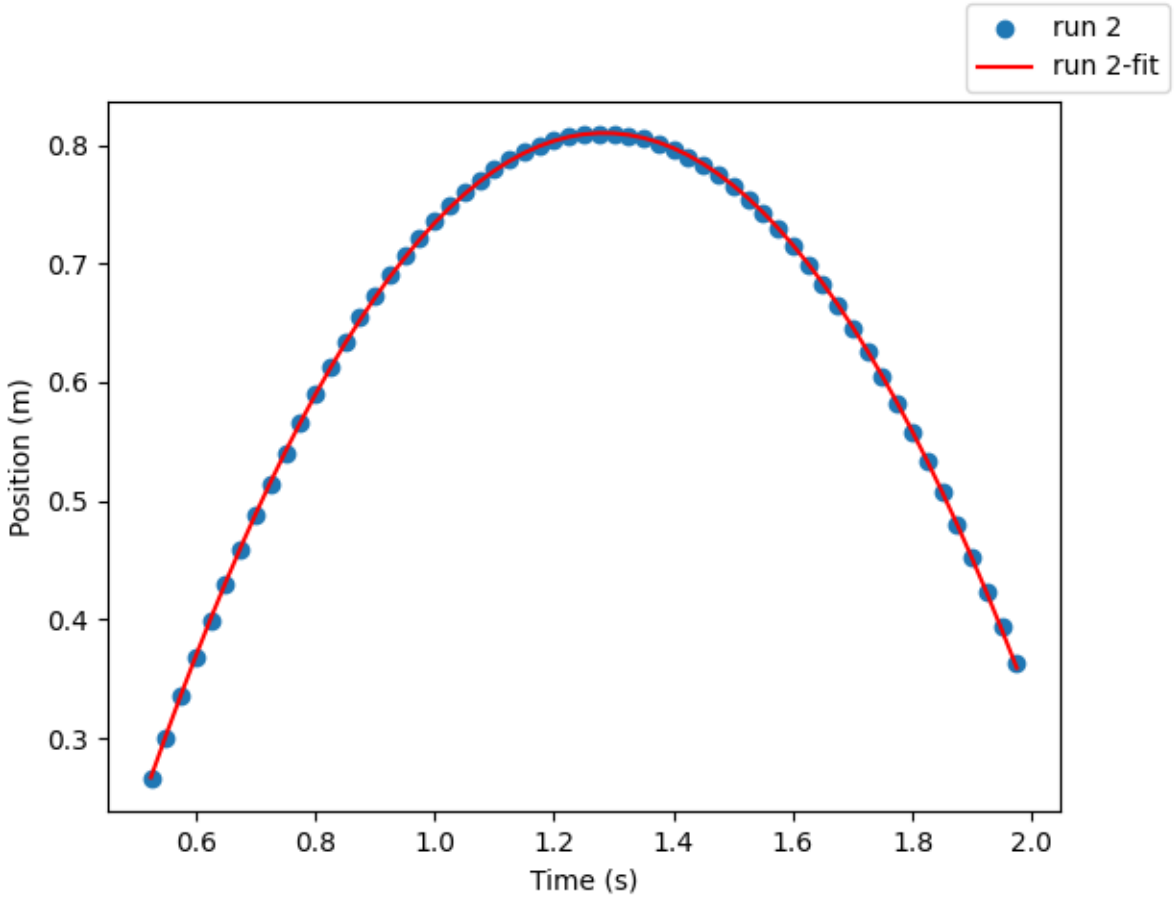
Jeremy Favro

October 11, 2024

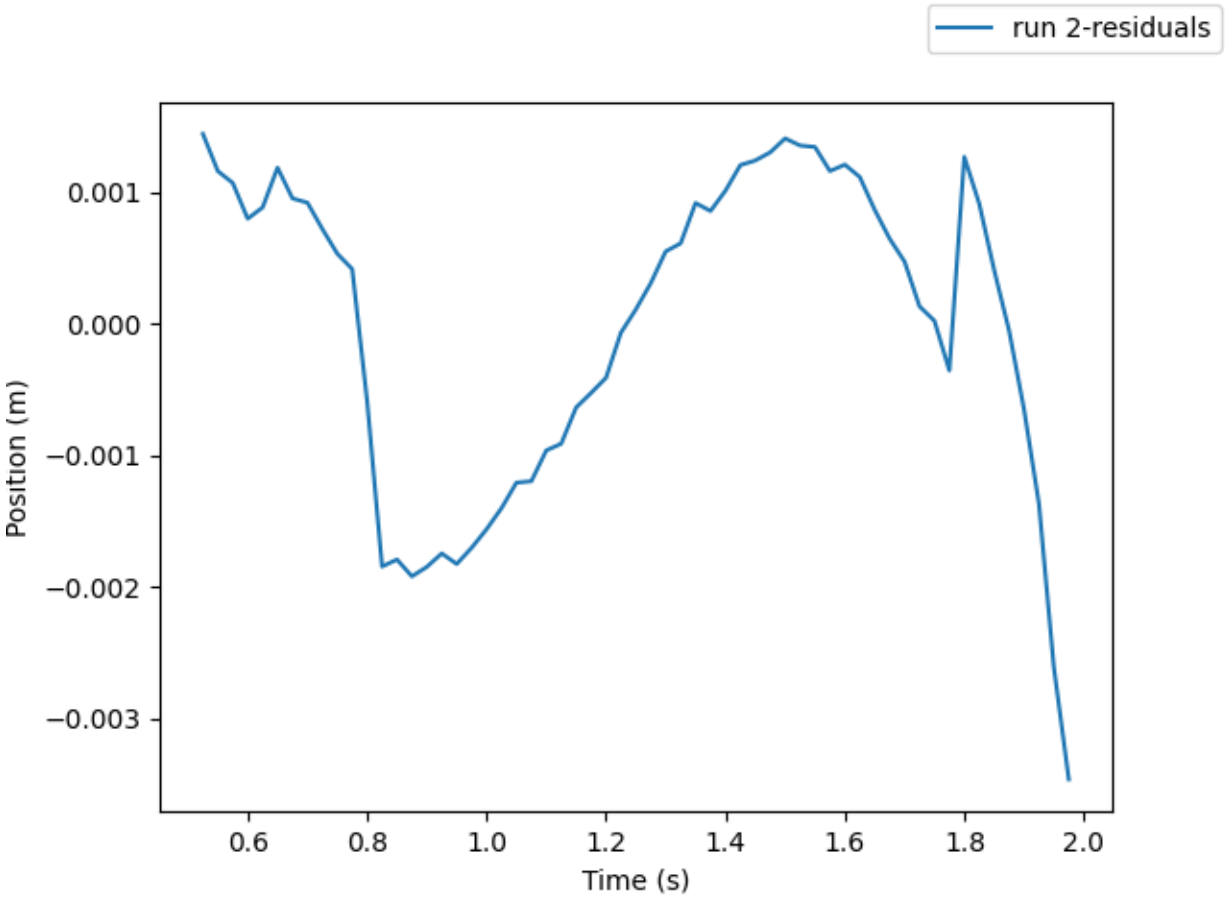
```
import matplotlib.pyplot as pyplot
import numpy as np

def plot_trimmed(path: str, label: str, mintime: float, minpos: float, maxtime: float) -> None:
    data = np.loadtxt(path, delimiter=",", skiprows=1) # does this path syntax work on windows? who knows
    data = data[ # not sure if there's a "smarter" way of doing this
    (data[:, 0] > mintime) &
    (data[:, 1] > minpos) &
    (data[:, 0] < maxtime)
    ]

    times = data[:, 0]
    positions = data[:, 1]
    pyplot.figure(1)
    pyplot.scatter(times, positions, label=label)

    fit, cov = np.polyfit(times, positions, 2, cov=True)

    # Without friction
    fit_data = []
    for t in times:
        fit_data.append(fit[0]*t**2 + fit[1]*t + fit[2])

    pyplot.plot(times, fit_data, label=label + "-fit", color="red")
    pyplot.figlegend()
    pyplot.xlabel("Time (s)")
    pyplot.ylabel("Position (m)")
    residual_data = []

    for t in range(len(times)):
        residual_data.append(fit_data[t] - positions[t])
    pyplot.figure(2)
    pyplot.plot(times, residual_data, label=label + "-residuals")
    pyplot.figlegend()
    pyplot.xlabel("Time (s)")
    pyplot.ylabel("Position (m)")

    g = 9.8

    a = 2*fit[0]
    a_u = np.sqrt(2*cov[0,0])

    print(f"{np.round(a, 3)}+-{np.round(a_u, 3)}")

    # With friction - up
    pyplot.figure(3)
```

```
    max_val_index = np.argmax(positions)
    positions_up = []
    times_up = []
    for i in range(0, max_val_index):
        positions_up.append(positions[i])
        times_up.append(times[i])

    fit, cov = np.polyfit(times_up, positions_up, 2, cov=True)

    up_fit_data = []
    for t in times_up:
        up_fit_data.append(fit[0]*t**2 + fit[1]*t + fit[2])

    a_p = 2*fit[0]
    a_p_u = np.sqrt(2*cov[0,0])
    pyplot.plot(times_up, up_fit_data, label=label+"-friction-fit-up", color="green")
    pyplot.scatter(times_up, positions_up, label=label+"-friction-up")
    pyplot.xlabel("Time (s)")
    pyplot.ylabel("Position (m)")

    # With friction - down
    positions_down = []
    times_down = []
    for i in range(max_val_index - 1, len(positions)):
        positions_down.append(positions[i])
        times_down.append(times[i])

    fit, cov = np.polyfit(times_down, positions_down, 2, cov=True)

    down_fit_data = []
    for t in times_down:
        down_fit_data.append(fit[0]*t**2 + fit[1]*t + fit[2])

    a_m = 2*fit[0]
    a_m_u = np.sqrt(2*cov[0,0])
    pyplot.plot(times_down, down_fit_data, label=label+"-friction-fit-down", color="purple")
    pyplot.scatter(times_down, positions_down, label=label+"-friction-down")
    pyplot.figlegend()
    pyplot.xlabel("Time (s)")
    pyplot.ylabel("Position (m)")

    print(f"Accel on down = {np.round(a_m, 3)}+-{np.round(a_m_u, 3)}")
    print(f"Accel on up = {np.round(a_p, 3)}+-{np.round(a_p_u, 3)}")
    theta = np.rad2deg(np.arcsin((a_m+a_p)/(-2*g)))
    mu = (a_m-a_p)/(2*g*np.cos(theta))

    print(f"theta = {theta}")
    print(f"coeff of friction = {mu}")

# plot_trimmed("./data/run1.csv", "run 1", .45, .1, 2.5)
plot_trimmed("./data/run2.csv", "run 2", .5, .1, 2)
# plot_trimmed("./data/run3.csv", "run 3", 1, .1, 3)
pyplot.figlegend()
pyplot.show()
```

This graph shows the recorded position-time data of the cart in blue and the result of polyfit in red. The calculated acceleration is $(-1.888 \pm 0.001)\, ms^{-1}$ which is comparable to the acceleration found in part back of $(-1.9 \pm 0.1)\, ms^{-1}$.



This graph shows the residuals in the case where zero friction is assumed. The graph looks non-random (though there is some randomness in the overall curve) and demonstrates that systematic error is a greater contributor to uncertainty than random error.



This graph shows the position-time data split between travelling up the ramp and down the ramp, and the fits for each case. The down/up $a_- = -1.852 \pm 0.003$ and $a_+ = -1.923 \pm 0.004$ which make sense compared to the "ideal" acceleration found earlier as we'd expect $a_- < a$ as friction would slow it down and for the inverse reason we'd expect $a_+ > a$ because friction is "helping". Using these acceleration values we can derive equations for μ and θ

$$\mu = \frac{a_- - a_+}{2g\cos\theta} = 0.0334$$
$$\theta = \arcsin\left(\frac{a_- + a_+}{-2g}\right) = 11.104\,\text{deg}$$

Both of which are reasonable as we don't expect friction to have an enormous impact given the difference between our "real" and "ideal" acceleration values is minimal and we expect the angle to be small as the acceleration parallel to the ramp is low compared to g.