Tensile Testing Data Reduction

This document serves as the data reduction process and data analysis of the Tensile Testing lab data for group A01-Y.

In [4]:
```
# Module imports
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

# Loading the data
aluminum_data = pd.read_csv("../Data/Aluminium_Monatonic.csv")
mystery_data = pd.read_csv("../Data/Mystery_Monatonic.csv")

# Dropping the '1' column
del aluminum_data['1']
del mystery_data['1']

# Dropping the units row
aluminum_data = aluminum_data.drop(0)
mystery_data = mystery_data.drop(0)

# Converting the entire dataset into floating point numbers
aluminum_data = aluminum_data.astype(float)
mystery_data = mystery_data.astype(float)

# Renaming the 'Strain 2' column
aluminum_data.rename(columns={'Strain 2': 'Extensometer Strain'}, inplace=True)
mystery_data.rename(columns={'Strain 2': 'Extensometer Strain'}, inplace=True)

# Calculating all relevant test sample dimension data
aluminum_initial_area = 12.68 / 1000 * 2.07 / 1000
aluminum_fracture_area = 11.76 / 1000 * 1.80 / 1000
aluminum_uniform_area = 12.02 / 1000 * 1.91 / 1000

aluminum_initial_length = 67.16 / 1000
aluminum_final_length = 74.94 / 1000

mystery_initial_area = 12.81 / 1000 * 2.03 / 1000
mystery_fracture_area = 9.18 / 1000 * 1.52 / 1000
mystery_uniform_area = 10.90 / 1000 * 1.81 / 1000

mystery_initial_length = 64.94 / 1000
mystery_final_length = 76.37 / 1000

# Placing this data in relevant table
table_1_data = {
    'Aluminum Specimen (m)': [
        67.16 / 1000,
        12.68 / 1000,
        2.07 / 1000,
        11.76 / 1000,
```

```
        1.80 / 1000,
        12.02 / 1000,
        1.91 / 1000,
        74.94 / 1000,
        2.00 * 2.54 / 1000
    ],
    'Mystery Specimen (m)': [
        64.94 / 1000,
        12.81 / 1000,
        2.03 / 1000,
        9.18 / 1000,
        1.52 / 1000,
        10.90 / 1000,
        1.81 / 1000,
        76.37 / 1000,
        2.00 * 2.54 / 1000
    ]
}

table_1_rows = [
    'Reduced Section Length',
    'Reduced Section Width',
    'Reduced Section Thickness',
    'Fractured Section Width',
    'Fractured Section Thickness',
    'Uniform Section Width',
    'Uniform Section Thickness',
    'Final Reduced Length',
    'Extensometer Gauge Length'
]

table_1 = pd.DataFrame(table_1_data, index=table_1_rows)

# Unit conversions for SI units across the data
aluminum_data['Displacement'] = aluminum_data['Displacement'].values / 1000 # mm to
mystery_data['Displacement'] = mystery_data['Displacement'].values / 1000 # mm to m

aluminum_data['Force'] = aluminum_data['Force'].values * 1000 # kN to N
mystery_data['Force'] = mystery_data['Force'].values * 1000 # kN to N

aluminum_data['Extensometer Strain'] = aluminum_data['Extensometer Strain'].values
mystery_data['Extensometer Strain'] = mystery_data['Extensometer Strain'].values /

# Data display test
# print(aluminum_data)
# print(mystery_data)

# Plotting just for vibes
%config InlineBackend.figure_format = 'svg'
plt.ioff()
# plt.scatter(aluminum_data['Strain 2'], aluminum_data['Force'])
# plt.xticks([])
# plt.yticks([])
# plt.show()
```

Out[4]: <contextlib.ExitStack at 0x1d05e8467e0>

Data Reduction

Now that the data has been loaded, sanitized, and converted into SI units, it's time to complete the data reduction steps in the lab manual.

In [5]:
```
# Computing the engineering stress based on initial area
aluminum_data['Engineering Stress'] = aluminum_data['Force'].values / aluminum_init
mystery_data['Engineering Stress'] = mystery_data['Force'].values / mystery_initial

# Estimating region for Young's Modulus (Aluminum)
derivative_test = aluminum_data['Engineering Stress'].diff() # When the derivative

high_slope = derivative_test.where(derivative_test > 1 * 10**6).dropna() # if there
# print(high_slope)
# print(high_slope.describe()) # while this cutoff value was rather random, low var

aluminum_region_start = 206
aluminum_region_end = 421
# print(aluminum_data['Engineering Stress'].iat[aluminum_region_start])
# print(aluminum_data['Engineering Stress'].iat[aluminum_region_end])

# Estimating region for Young's Modulus (Mystery)
derivative_test = mystery_data['Engineering Stress'].diff() # When the derivative i

high_slope = derivative_test.where(derivative_test > 0.8 * 10**6).dropna() # if the
# print(high_slope)
# print(high_slope.describe()) # while this cutoff value was rather random, low var

mystery_region_start = 57
mystery_region_end = 361
# print(mystery_data['Engineering Stress'].iat[mystery_region_start])
# print(mystery_data['Engineering Stress'].iat[mystery_region_end])

# Computing Young's Modulus over the region
aluminum_delta_stress = aluminum_data['Engineering Stress'].iloc[aluminum_region_st
# print(aluminum_delta_stress)
aluminum_delta_strain = aluminum_data['Extensometer Strain'].iloc[aluminum_region_s
# print(aluminum_delta_strain)
aluminum_youngs_modulus = aluminum_delta_stress / aluminum_delta_strain
# print(aluminum_youngs_modulus / (10**9))

mystery_delta_stress = mystery_data['Engineering Stress'].iloc[mystery_region_start
# print(mystery_delta_stress)
mystery_delta_strain = mystery_data['Extensometer Strain'].iloc[mystery_region_star
# print(mystery_delta_strain)
mystery_youngs_modulus = mystery_delta_stress / mystery_delta_strain
# print(mystery_youngs_modulus / (10**9))

# Finding true values to add to the data
aluminum_data['Engineering Strain'] = aluminum_data['Displacement'].values / alumin
mystery_data['Engineering Strain'] = mystery_data['Displacement'].values / mystery_
```

```
aluminum_data['1 + Epsilon'] = 1 + aluminum_data['Engineering Strain'].values
aluminum_data['True Stress'] = aluminum_data['Engineering Stress'].values * aluminu
aluminum_data['True Strain'] = np.log(aluminum_data['1 + Epsilon'])

mystery_data['1 + Epsilon'] = 1 + mystery_data['Engineering Strain'].values
mystery_data['True Stress'] = mystery_data['Engineering Stress'].values * mystery_d
mystery_data['True Strain'] = np.log(mystery_data['1 + Epsilon'])

# Data reduction step 3 (aluminum)
aluminum_uts = aluminum_data['Engineering Stress'].max()
aluminum_tuts = (aluminum_data['True Stress'].values * aluminum_initial_area / alum
aluminum_tfs = (aluminum_data['True Stress'].values * aluminum_initial_area / alumi
aluminum_efs = aluminum_data['Engineering Stress'].values[-1]

uts_index = aluminum_data['True Stress'].idxmax()
aluminum_tutsn = np.log(aluminum_data['Extensometer Strain'].iloc[uts_index] + 1)

aluminum_efsni = aluminum_data['Engineering Strain'].values[-1]
aluminum_efsnf = aluminum_data['Engineering Strain'].values[-1] * aluminum_initial_

aluminum_tfsni = aluminum_data['True Strain'].values[-1]
aluminum_tfsnf = aluminum_data['True Strain'].values[-1] * aluminum_initial_length

aluminum_tfsne = np.log(aluminum_data['Extensometer Strain'].values[-1] + 1)

# Data reduction step 3 (mystery)
mystery_uts = mystery_data['Engineering Stress'].max()
mystery_tuts = (mystery_data['True Stress'].values * mystery_initial_area / mystery
mystery_tfs = (mystery_data['True Stress'].values * mystery_initial_area / mystery_
mystery_efs = mystery_data['Engineering Stress'].values[-1]

uts_index = mystery_data['True Stress'].idxmax()
mystery_tutsn = np.log(mystery_data['Extensometer Strain'].iloc[uts_index] + 1)

mystery_efsni = mystery_data['Engineering Strain'].values[-1]
mystery_efsnf = mystery_data['Engineering Strain'].values[-1] * mystery_initial_len

mystery_tfsni = mystery_data['True Strain'].values[-1]
mystery_tfsnf = mystery_data['True Strain'].values[-1] * mystery_initial_length / m

mystery_tfsne = np.log(mystery_data['Extensometer Strain'].values[-1] + 1)

# Strain hardening index and strength coefficient (aluminum)
# safe points well within the strain hardening region are time steps 1000 and 1500
x1 = np.log(aluminum_data['True Strain'].values[1000])
x2 = np.log(aluminum_data['True Strain'].values[1500])
y1 = np.log(aluminum_data['True Stress'].values[1000])
y2 = np.log(aluminum_data['True Stress'].values[1500])

aluminum_n = (y2 - y1) / (x2 - x1)
ln_K = y2 - aluminum_n * x2
aluminum_K = np.exp(ln_K)
# print(aluminum_n)
# print(aluminum_K / (10**6))
```

```
# plt.plot(np.log(np.log(aluminum_data['Extensometer Strain'].values + 1)), np.log(
# plt.show()

# Strain hardening index and strength coefficient (mystery)
# safe points well within the strain hardening region are time steps 1000 and 1500
x1 = np.log(mystery_data['True Strain'].values[1000])
x2 = np.log(mystery_data['True Strain'].values[1500])
y1 = np.log(mystery_data['True Stress'].values[1000])
y2 = np.log(mystery_data['True Stress'].values[1500])

mystery_n = (y2 - y1) / (x2 - x1)
ln_K = y2 - mystery_n * x2
mystery_K = np.exp(ln_K)
# print(mystery_n)
# print(mystery_K / (10**6))

# plt.plot(np.log(np.log(mystery_data['Extensometer Strain'].values + 1)), np.log(m
# plt.show()
```

Results For Report

Creation of plots and tables for the lab report.

In [6]:
```
# Save table one
table_1.to_excel('../Documents/figure1.xlsx')

# Graph of engineering stress vs strain (2-3)
plt.scatter(aluminum_data['Engineering Strain'].values * (10**6), aluminum_data['En
plt.ylabel('Engineering Stress (MPa)')
plt.xlabel('Engineering Strain (microstrain)')
plt.grid()
plt.savefig('../Documents/figure2.png', dpi=300)
plt.close()

plt.scatter(mystery_data['Engineering Strain'].values * (10**6), mystery_data['Engi
plt.ylabel('Engineering Stress (MPa)')
plt.xlabel('Engineering Strain (microstrain)')
plt.grid()
plt.savefig('../Documents/figure3.png', dpi=300)
plt.close()

# Young's Modulus table (4)
mpa_data = np.array([aluminum_youngs_modulus, mystery_youngs_modulus, 73.1 * (10**9
mpa_to_ksi = 0.1450377377 # source: https://www.unitconverters.net/pressure/megapas
table_4_data = {
    'MPa': mpa_data,
    'ksi': mpa_data * mpa_to_ksi
}
table_4_rows = ["Young's Modulus, Aluminum", "Young's Modulus, Mystery", "YM, Publi

table_4 = pd.DataFrame(table_4_data, index=table_4_rows)
table_4.to_excel('../Documents/figure4.xlsx')

# Tables with data reduction points (5)
```

```
"""
For future reference, requiring this data to be in two distinct tables is mathemati
unless completed in the way below, which is a complete waste of space. I've also no
locations in the lab manual that are clearly out of date (such as the objective men
testing). I HIGHLY RECCOMEND for future semesters of this class revising the lab ma
errors or updates relevant for future years. Thank you!
"""
table_5_rows = [
    'Engineering ultimate tensile stress, MPa',
    'True ultimate tensile stress, MPa',
    'Engineering fracture stress, MPa',
    'True fracture stress, MPa',

    'Engineering ultimate tensile stress, ksi',
    'True ultimate tensile stress, ksi',
    'Engineering fracture stress, ksi',
    'True fracture stress, ksi',

    'True ultimate tensile strain from extensometer, m/m',
    'True fracture strain from extensometer, m/m',
    'True fracture strain from initial dimensions, m/m',
    'True fracture strain from fracture dimensions, m/m',
    'Engineering fracture strain from initial dimensions, m/m',
    'Engineering fracture strain from fracture dimensions, m/m'
]

t5al_data = {'Aluminum': [
    aluminum_uts / (10**6),
    aluminum_tuts / (10**6),
    aluminum_efs / (10**6),
    aluminum_tfs / (10**6),

    aluminum_uts / (10**6) * mpa_to_ksi,
    aluminum_tuts / (10**6) * mpa_to_ksi,
    aluminum_efs / (10**6) * mpa_to_ksi,
    aluminum_tfs / (10**6) * mpa_to_ksi,

    aluminum_tutsn,
    aluminum_tfsne,
    aluminum_tfsni,
    aluminum_tfsnf,
    aluminum_efsni,
    aluminum_efsnf
]}

table_5_aluminum = pd.DataFrame(t5al_data, index=table_5_rows)
table_5_aluminum.to_excel('../Documents/figure5-aluminum.xlsx')

t5my_data = {'Mystery': [
    mystery_uts / (10**6),
    mystery_tuts / (10**6),
    mystery_efs / (10**6),
    mystery_tfs / (10**6),

    mystery_uts / (10**6) * mpa_to_ksi,
    mystery_tuts / (10**6) * mpa_to_ksi,
```

```
    mystery_efs / (10**6) * mpa_to_ksi,
    mystery_tfs / (10**6) * mpa_to_ksi,

    mystery_tutsn,
    mystery_tfsne,
    mystery_tfsni,
    mystery_tfsnf,
    mystery_efsni,
    mystery_efsnf
]}

table_5_mystery = pd.DataFrame(t5my_data, index=table_5_rows)
table_5_mystery.to_excel('../Documents/figure5-mystery.xlsx')

# Stress vs extensometer strain graphs
main = plt.scatter(aluminum_data['Extensometer Strain'].values * (10**6), aluminum_
plt.ylabel('Engineering Stress (MPa)')
plt.xlabel('Extensometer Strain (microstrain)')

tfs = plt.scatter(aluminum_tfsne * (10**6), aluminum_tfs / (10**6), marker='x')
tfs.set_label('True Fracture Stress')
tuts = plt.scatter(aluminum_tutsn * (10**6), aluminum_tuts / (10**6), marker='s')
tuts.set_label('True Ultimate Tensile Stress')

mid_true_stress = aluminum_data['Engineering Stress'].iloc[1000] * (1 + aluminum_da
mid_true_strain = np.log(1 + aluminum_data['Extensometer Strain'].iloc[1000])
mid = plt.scatter(mid_true_strain * (10**6), mid_true_stress / (10**6), marker='*')
mid.set_label('True Stress-Strain Point')

# yield point is approximately at 330 MPa
yield_estimate = 330 * (10**6)
yield_value = aluminum_data['Engineering Stress'].iloc[np.searchsorted(aluminum_dat
yield_index = aluminum_data.loc[aluminum_data['Engineering Stress'] == yield_value]
yield_strain = yield_index['Extensometer Strain'].values[0]
yieldpt = plt.scatter(yield_strain * (10**6), yield_value / (10**6), marker='D')
yieldpt.set_label('Yield Stress')

# creating an array to roughly imitate a true stress-strain curve
x_points = np.array([0, yield_strain, mid_true_strain, aluminum_tutsn, aluminum_tfs
y_points = np.array([0, yield_value, mid_true_stress, aluminum_tuts, aluminum_tfs])
plt.plot(x_points * (10**6), y_points / (10**6), c=[0, 0, 0], label='True Stress-St

plt.grid()
plt.legend()
plt.savefig('../Documents/figure6-aluminum.png', dpi=300)
# plt.show()
plt.close()

main = plt.scatter(mystery_data['Extensometer Strain'].values * (10**6), mystery_da
plt.ylabel('Engineering Stress (MPa)')
plt.xlabel('Extensometer Strain (microstrain)')

tfs = plt.scatter(mystery_tfsne * (10**6), mystery_tfs / (10**6), marker='x')
tfs.set_label('True Fracture Stress')
tuts = plt.scatter(mystery_tutsn * (10**6), mystery_tuts / (10**6), marker='s')
tuts.set_label('True Ultimate Tensile Stress')
```

```
mid_true_stress = mystery_data['Engineering Stress'].iloc[2500] * (1 + mystery_data
mid_true_strain = np.log(1 + mystery_data['Extensometer Strain'].iloc[2500])
mid = plt.scatter(mid_true_strain * (10**6), mid_true_stress / (10**6), marker='*')
mid.set_label('True Stress-Strain Point')

# yield point is approximately at 290 MPa
yield_estimate = 290 * (10**6)
yield_value = mystery_data['Engineering Stress'].iloc[np.searchsorted(mystery_data[
yield_index = mystery_data.loc[mystery_data['Engineering Stress'] == yield_value]
yield_strain = yield_index['Extensometer Strain'].values[0]
yieldpt = plt.scatter(yield_strain * (10**6), yield_value / (10**6), marker='D')
yieldpt.set_label('Yield Stress')

# creating an array to roughly imitate a true stress-strain curve
x_points = np.array([0, yield_strain, mid_true_strain, mystery_tutsn, mystery_tfsne
y_points = np.array([0, yield_value, mid_true_stress, mystery_tuts, mystery_tfs])
plt.plot(x_points * (10**6), y_points / (10**6), c=[0, 0, 0], label='True Stress-St

# source for true stress-strain imitation: https://mechanicalc.com/reference/mechan

plt.grid()
plt.legend()
plt.savefig('../Documents/figure6-mystery.png', dpi=300)
# plt.show()
plt.close()

# Strain hardening index and strength coefficient (7)
table_7_data = {
    'Aluminum': [aluminum_K, aluminum_n],
    'Mystery': [mystery_K, mystery_n]
}
table_7_rows = ['K, Pa', 'n']

table_7 = pd.DataFrame(table_7_data, index=table_7_rows)
table_7.to_excel('../Documents/figure7.xlsx')

# Mystery material selection (8)
"""
Properties of choice:
 - E
 - color
As the mystery material provided was a golden color, many materials can be immediat
These include:
 - 4130 Steel
 - 220 Nickel
 - 316 Series SS
 - Aluminum 7075
 - CRES PH 17-4
 - Inconel 625
"""
table_8_rows = ['Mystery', '145 Copper', '110 Copper', '510 Bronze', '260 Brass', '
table_8_data = {
    'Young\'s Modulus': [f'{round(mystery_youngs_modulus / (10**9), 1)} GPa', '120.
}
print(table_8_data)
```

```
table_8 = pd.DataFrame(table_8_data, index=table_8_rows)
table_8.to_excel('../Documents/figure8.xlsx') # Most likely 510 Bronze
```

```
{"Young's Modulus": ['69.4 GPa', '120.7 GPa', '110 GPa', '41.4 GPa', '110 GPa', '10
4.8 GPa']}
```