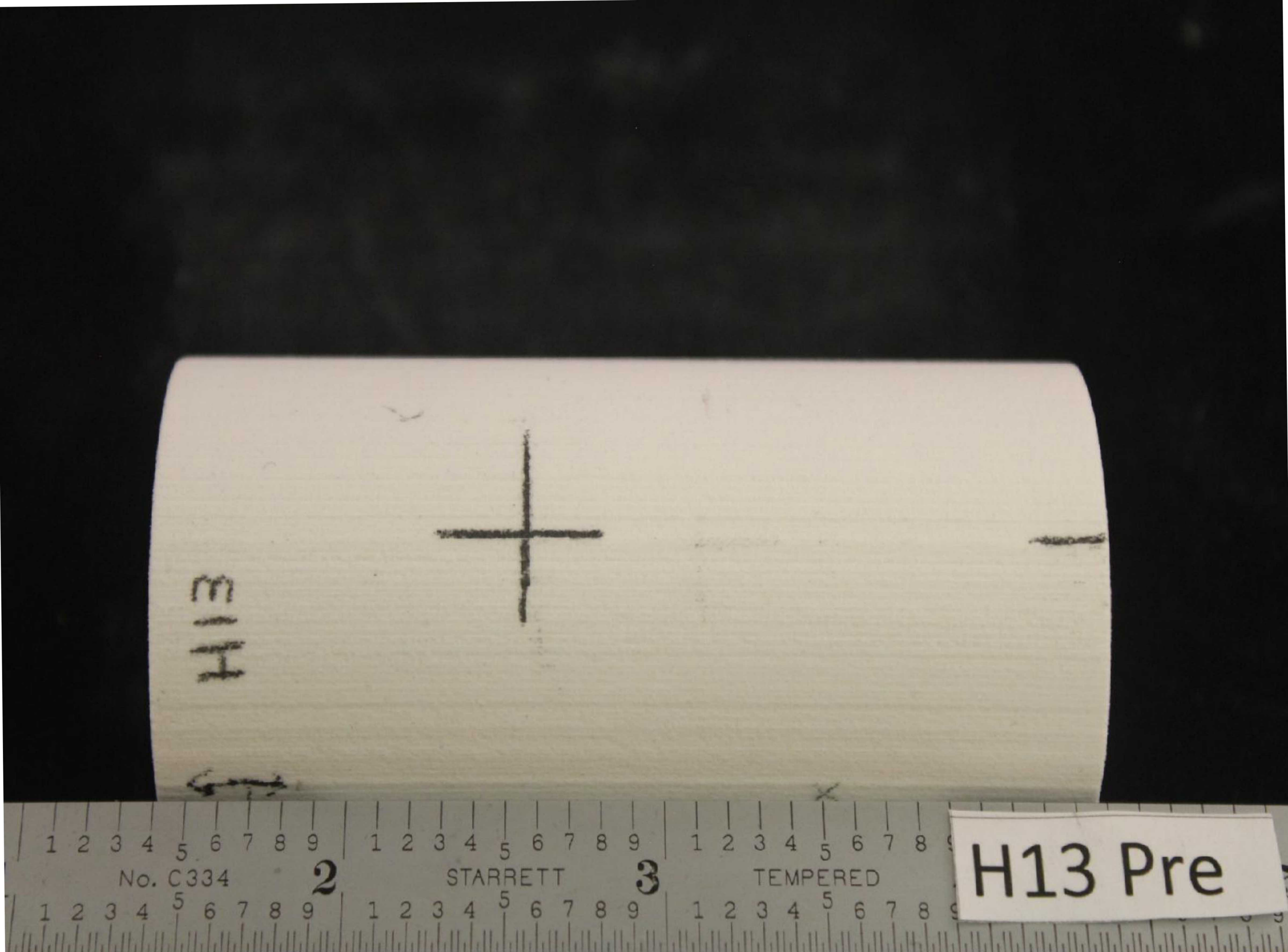
SAND2019-6661R

Michelle Williams
University of Arizona

MNE520 Data Analysis Final Project
3D Printed Rock Analysis using Python

May 2019

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525


No. C334
STARRETT
TEMPERED
H13 Pre

H13 Post

H13_Axial_Parallel Velocity

amplitude (V)

2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5

0.0
0.2
0.4
0.6
0.8
1.0
1e−4

time (sec)

H13_Axial_Perpendicular Velocity

amplitude (V)

3
2
1
0
−1
−2

0.0
0.2
0.4
0.6
0.8
1.0
1e−4

time (sec)

H13_Radial_Parallel Velocity

amplitude (V)

3
2
1
0
−1
−2

0.0
0.2
0.4
0.6
0.8
1.0
1e−4

time (sec)

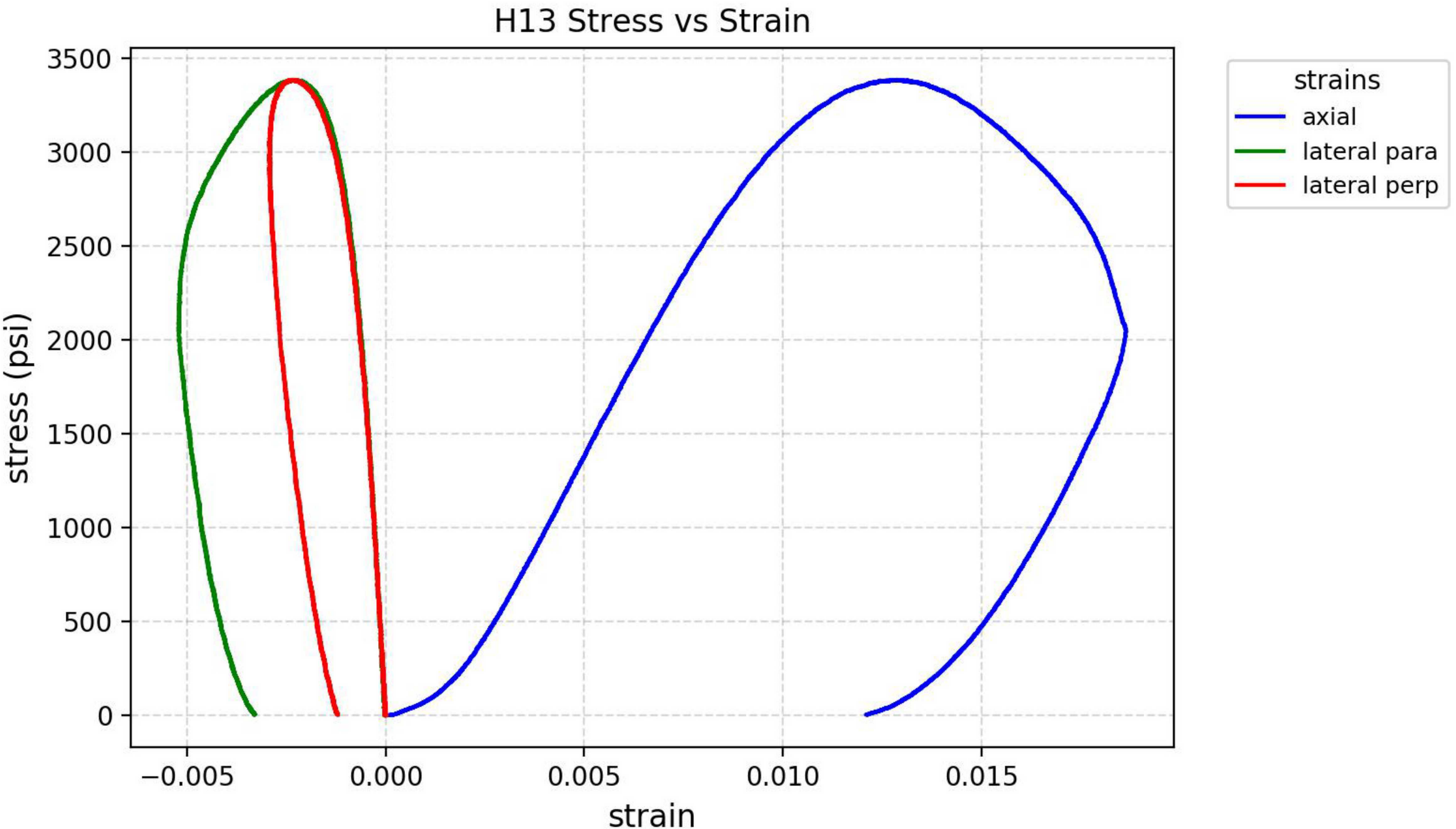
H13_Radial_Perpendicular Velocity

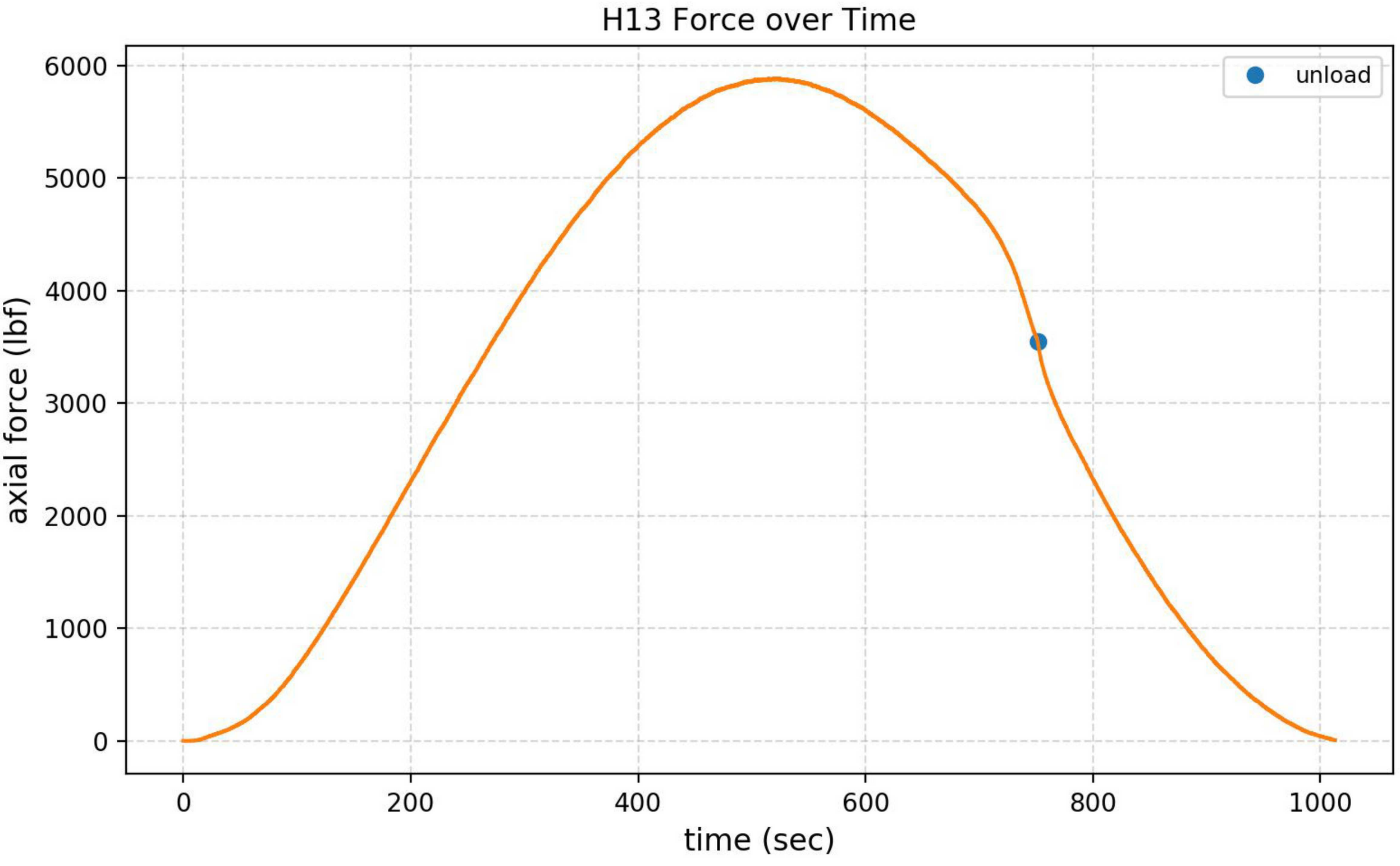
amplitude (V)

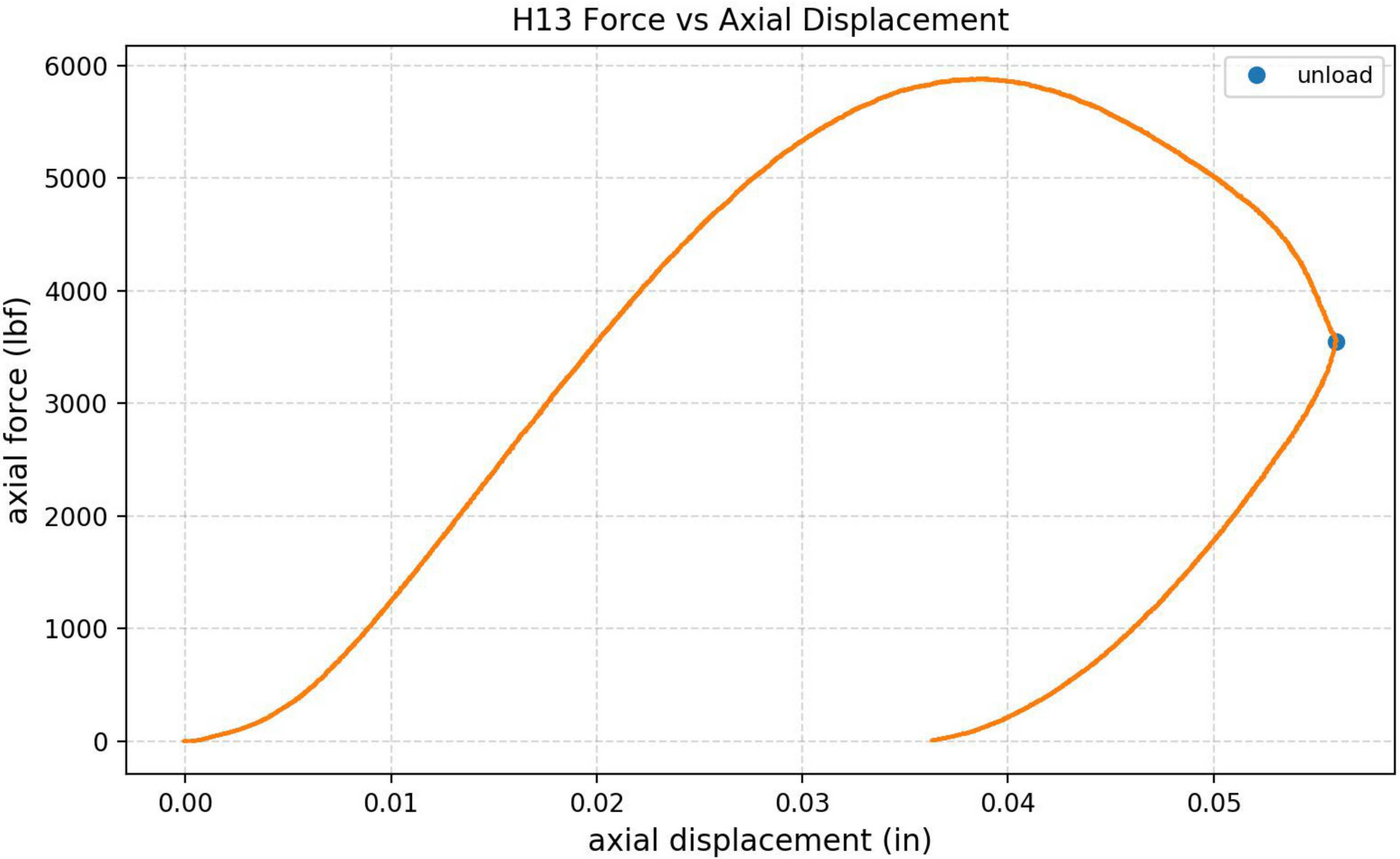
3
2
1
0
−1
−2

0.0
0.2
0.4
0.6
0.8
1.0
1e−4

time (sec)


H13 Stress vs Strain
stress (psi)
strain
3500
3000
2500
2000
1500
1000
500
0
−0.005
0.000
0.005
0.010
0.015
strains
axial
lateral para
lateral perp


H13 Force over Time
unload
6000
5000
4000
3000
2000
1000
0
axial force (lbf)
0
200
400
600
800
1000
time (sec)


H13 Force vs Axial Displacement
unload
6000
5000
4000
3000
2000
1000
0
axial force (lbf)
0.00
0.01
0.02
0.03
0.04
0.05
axial displacement (in)

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Sat Apr 20 17:15:59 2019

@author: mwilli9
"""
from math import pi
import pandas as pd
import matplotlib.pyplot as plt
from PIL import Image
from numpy import diff

#close all open plots
plt.close('all')

# ---------------------------------------------
#Define functions for importing, formating, and ploting data
# ---------------------------------------------

#function to calculate sample area (circle)
def sample_area(specs):
    # area = pi*((lateral paral.length + laterial perpendic. length)/2)^2)/4
    return pi*(((specs.iloc[1]['Value'] + specs.iloc[2]['Value'])/2.)**2)/4.

#function to calculate stress
def stress(test_data, area):
    #stress = axial force/area (in2)
    return test_data['Axial Force'].values/area

#function to calculate axial strain
def strain(test_data, specs):
    #get sample height from specs excel sheet
    sample_height = specs[specs['Dimension'].str.contains('Axial Length')]['Value'][0]
    #axial strain = axial displacement/sample height
    return test_data['Axial Displacement'].values/sample_height

#function to calculate laterial strains
def lateral_strain(test_data, specs, lvdts):
    #laterial strain 1 = lvdt/lat parallel length
    lstrain1 = test_data[lvdts[0]].values/specs.iloc[1]['Value']
    #laterial strain 2 = lvdt/lat perpendicular length
    lstrain2 = test_data[lvdts[1]].values/specs.iloc[2]['Value']
    return lstrain1, lstrain2
```

```
#function for importing multiple velocity data files
def import_velocity(filepath):
   #importing data into data frame
   vdata = pd.read_csv(filepath, usecols = [3,5])
   vunit = vdata.loc[0]
   #removing unit row from data frame
   vdata.drop(0, inplace=True)
   #change dataframe from object to numeric format
   vdata = vdata.transform(pd.to_numeric, errors='coerce')
   return vdata, vunit

#function for selecting start of p wave
def p_wave_select(dataframe):
   #should be centered at zero
   #cutoff initial time
   init_index = dataframe['Source'] > 0.5E-5
   #determine bias in measurement baseline
   inst_bias = (dataframe['CH2'][init_index]).median()
   #typically deviates below zero first
   #determine first time the signcal deviates below bias for more than 1 meas.
   p_wave_start = two_meas_below(dataframe[init_index], inst_bias)
   return dataframe[init_index].values[p_wave_start]


def two_meas_below(dataframe, bias):
   for i in range(len(dataframe.values)):
      #find first time signal is consistently above or below bias
      if ((dataframe['CH2'].values[i:i+7] < bias).all() or
         (dataframe['CH2'].values[i:i+7] > bias).all()):
         return i

def s_wave_select(dataframe):
   #cutoff initial time
   init_index = dataframe['Source'] > 0.5E-5
   #should be increasing or decreasing for many timesteps
   for i in range(len(dataframe[init_index].values)):
      dx = diff(dataframe['CH2'][init_index].values[i:i+12:2 ])
      if ((dx > 0).all() or (dx < 0).all()):
         return dataframe[init_index].values[i]


#function for generating velocity plot
def plot_velocity(dataframe, title, results_list):
```

```
    #use function to find start of p wave
    p_start = p_wave_select(dataframe)
    #use function to find start of s wave
    s_start = s_wave_select(dataframe)
    print(s_start)

    dataframe.plot(x='Source', y='CH2', kind='line', figsize=(8, 5),
                   legend=False)

    plt.plot(p_start[0], p_start[1], 'r*')
    plt.plot(s_start[0], s_start[1], 'g*')

    plt.ticklabel_format(axis='x', style='sci', scilimits=(-2,2))
    plt.locator_params(axis='x', nbins=6)
    plt.xlim(0, 1E-4)
    plt.grid(color='gray', linestyle='--', alpha=0.3)
    plt.xlabel('time (sec)', fontsize=12)
    plt.ylabel('amplitude (V)', fontsize=12)
    plt.title(title + ' Velocity')
    plt.tight_layout()
    file_name = 'Results/' + title + '.png'
    plt.savefig(file_name, dpi=200)
    results_list += [Image.open(file_name)]

#function for generating stress strain plot
def plot_stress_strain(dataframe, title, results_list):
    plt.figure(figsize=(8, 5))
    plt.subplots_adjust(bottom=0.135, right=0.8, left=.105)
    plt.plot(dataframe['Axial Strain'].values, dataframe['Axial Stress'], 'b-',
             label='axial')
    plt.plot(dataframe['Lstrain1'].values, dataframe['Axial Stress'], 'g-',
             label='lateral para')
    plt.plot(dataframe['Lstrain2'].values, dataframe['Axial Stress'], 'r-',
             label='lateral perp')
    plt.xlabel('strain', fontsize=12)
    plt.ylabel('stress (psi)', fontsize=12)
    plt.grid(color='gray', linestyle='--', alpha=0.3)
    plt.legend(bbox_to_anchor=(1.04, 1), loc='upper left', fontsize=9,
               title='strains')
    plt.title(title + ' Stress vs Strain')
    file_name = 'Results/' + title + '.png'
    plt.savefig(file_name, dpi=200)
    results_list += [Image.open(file_name)]
```

```
#function for generating force over time plot
def plot_force_time(dataframe, title, results_list):
   #determine unload time from maximum displacement
   unload = dataframe['Axial Displacement'].argmax()

   plt.figure(figsize=(8, 5))
   plt.plot(dataframe['Running Time'].values[unload],
         dataframe['Axial Force'].values[unload], 'o')
   plt.plot(dataframe['Running Time'], dataframe['Axial Force'])
   plt.ylabel('axial force (lbf)', fontsize=12)
   plt.xlabel('time (sec)', fontsize=12)
   plt.grid(color='gray', linestyle='--', alpha=0.3)
   plt.legend(['unload'], loc=0, fontsize=9)
   plt.title(title + ' Force over Time')
   plt.tight_layout()
   file_name = 'Results/' + title + 'force_time.png'
   plt.savefig(file_name, dpi=200)
   results_list += [Image.open(file_name)]


#function for generating force versus displacement plot
def plot_force_displacement(dataframe, title, results_list):
   #determine unload time from maximum displacement
   unload = dataframe['Axial Displacement'].argmax()

   plt.figure(figsize=(8, 5))
   plt.plot(dataframe['Axial Displacement'].values[unload],
         dataframe['Axial Force'].values[unload], 'o')
   plt.plot(dataframe['Axial Displacement'], dataframe['Axial Force'])
   plt.ylabel('axial force (lbf)', fontsize=12)
   plt.xlabel('axial displacement (in)', fontsize=12)
   plt.grid(color='gray', linestyle='--', alpha=0.3)
   plt.legend(['unload'], loc=0, fontsize=9)
   plt.title(title + ' Force vs Axial Displacement')
   plt.tight_layout()
   file_name = 'Results/' + title + 'force_displacement.png'
   plt.savefig(file_name, dpi=200)
   results_list += [Image.open(file_name)]

# ----------------------------------------------
#Import, format, and format data
# ----------------------------------------------
```

```
#defining test specifications filepath
specsdata = 'H13 Dimensions.xlsx'
specs = pd.read_excel(specsdata, sheetname='dimensions')

#defining test data filepath
ucstestdata = 'Test Data/' + pd.read_excel(specsdata,
                                sheetname='test data')['file'][0]
#defining names of lvdt
lvdt1 = pd.read_excel(specsdata, sheetname='test data')['lvdt 1'][0]
lvdt2 = pd.read_excel(specsdata, sheetname='test data')['lvdt 2'][0]
#store as together as a tuple
lvdts = lvdt1, lvdt2

#importing data into data frame
tdata = pd.read_csv(ucstestdata, header=1)
#remove whitespace from column headers
tdata.columns = tdata.columns.str.strip()
#create series of test data units
tunit = tdata.loc[0]

#removing unit row from data frame
tdata.drop(0, inplace=True)
#change dataframe from object to numeric format
tdata = tdata.transform(pd.to_numeric, errors='coerce')

#defining velcoity data filepath
velocitydata1 = 'Velocity/' + pd.read_excel(specsdata,
                                sheetname='velocity data')['file'][0]
velocitydata2 = 'Velocity/' + pd.read_excel(specsdata,
                                sheetname='velocity data')['file'][1]
velocitydata3 = 'Velocity/' + pd.read_excel(specsdata,
                                sheetname='velocity data')['file'][2]
velocitydata4 = 'Velocity/' + pd.read_excel(specsdata,
                                sheetname='velocity data')['file'][3]

#importing velocity datasets into organized data frames
vdata1, vunit1 = import_velocity(velocitydata1)
vdata2, vunit2 = import_velocity(velocitydata2)
vdata3, vunit3 = import_velocity(velocitydata3)
vdata4, vunit4 = import_velocity(velocitydata4)

#calculate sample area
sample_area = sample_area(specs)
#calculate axial stress
```

```
tdata['Axial Stress'] = stress(tdata, sample_area)
#calculate strain
tdata['Axial Strain'] = strain(tdata, specs)
#calculate laterial strains
lstrain1, lstrain2 = lateral_strain(tdata, specs, lvdts)
tdata['Lstrain1'] = lstrain1
tdata['Lstrain2'] = lstrain2

#importing pre photo
prephoto = 'Photos/' + pd.read_excel(specsdata, sheetname='photos')['pre'][0]
#open pre photo jpg file
preImage = Image.open(prephoto)
#change image size
preImage = preImage.resize((1600, 1200))
#save pre photo image as pdf
preImage.save('Results/' + prephoto[7:-4] + '.pdf', "PDF", Quality = 100)

#importing post photo
postphoto = 'Photos/' + pd.read_excel(specsdata, sheetname='photos')['post'][0]
#open post photo jpg file
postImage = Image.open(postphoto)
#change image size
postImage = postImage.resize((1600, 1200))
#save post photo image as pdf
postImage.save('Results/' + postphoto[7:-4] + '.pdf', "PDF", Quality = 100)
#add image to results list
results_list = [postImage]

# ----------------------------------------------
#Plot data
# ----------------------------------------------

#plot veloity data
plot_velocity(dataframe=vdata1, title=velocitydata1[9:-4],
        results_list=results_list)
plot_velocity(dataframe=vdata2, title=velocitydata2[9:-4],
        results_list=results_list)
plot_velocity(dataframe=vdata3, title=velocitydata3[9:-4],
        results_list=results_list)
plot_velocity(dataframe=vdata4, title=velocitydata4[9:-4],
        results_list=results_list)

#plot stress strain data
plot_stress_strain(dataframe=tdata, title=ucstestdata[10:-25],
```

```
            results_list=results_list)

#plot force vs time
plot_force_time(dataframe=tdata, title=ucstestdata[10:-25],
            results_list=results_list)

#plot force vs displacement
plot_force_displacement(dataframe=tdata, title=ucstestdata[10:-25],
            results_list=results_list)

# ----------------------------------------------
#Create PDF containing all results
# ----------------------------------------------

#name of results file
results_file = 'Results/AllResults.pdf'
#add images to pdf
preImage.save(results_file, 'PDF', resolution=100, save_all=True,
          append_images=results_list)
```