Data science for Materials Science & Engineering

Linear Regression in materials science – Young's Modulus

In this module

- Introduction to linear regression with materials examples
- Hands-on tutorials using nanoHUB: Young's modulus and yield stress (this file)
- Hands-on tutorials using nanoHUB: correlating materials properties
- Homework assignments

Michael Sakano and Alejandro Strachan
msakano@purdue.edu || strachan@purdue.edu
School of Materials Engineering & Network for Computational Nanotechnology
Purdue University
West Lafayette, Indiana USA

Objective and prerequisites

After completing this activity you will be able to:

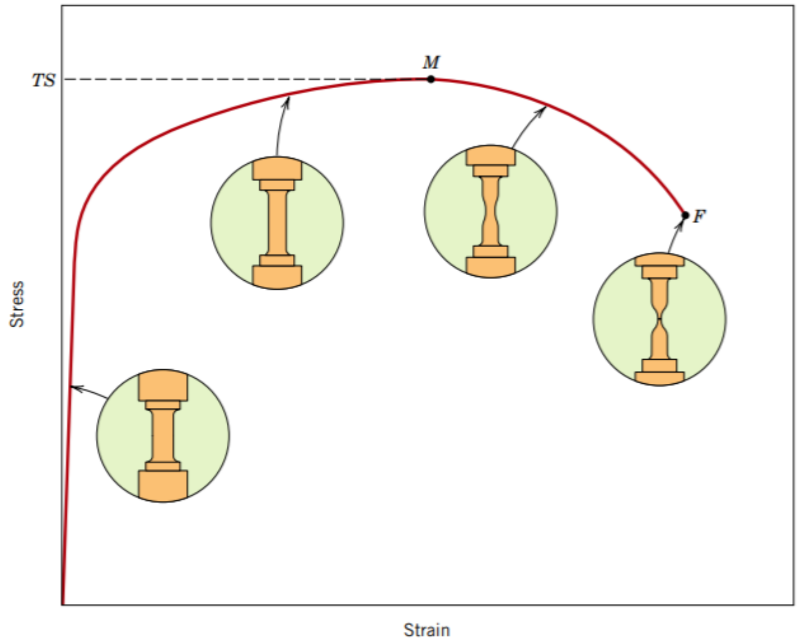
1. Use Pandas data structures to organize your data
2. Use linear regression to obtain Young's modulus and yield stress from stress-strain data
3. Explore how the fit parameters affect the results

Pre-requisites:

- Basic Python programming. See Juan C. Verduzco https://nanohub.org/resources/33266

Linear Regression for mechanical properties

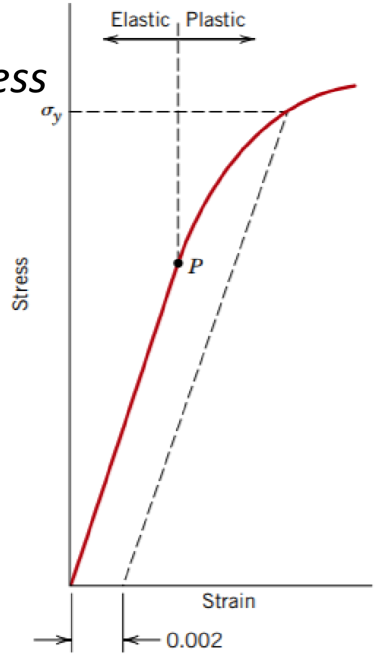
From *Materials Science and Engineering: An Introduction, William D. Callister Jr. and David G. Rethwisch, 8th edition*



$$Stress = \frac{Force}{Area}$$

$$Strain = \frac{Change\ in\ length}{Original\ length}$$

Yield stress



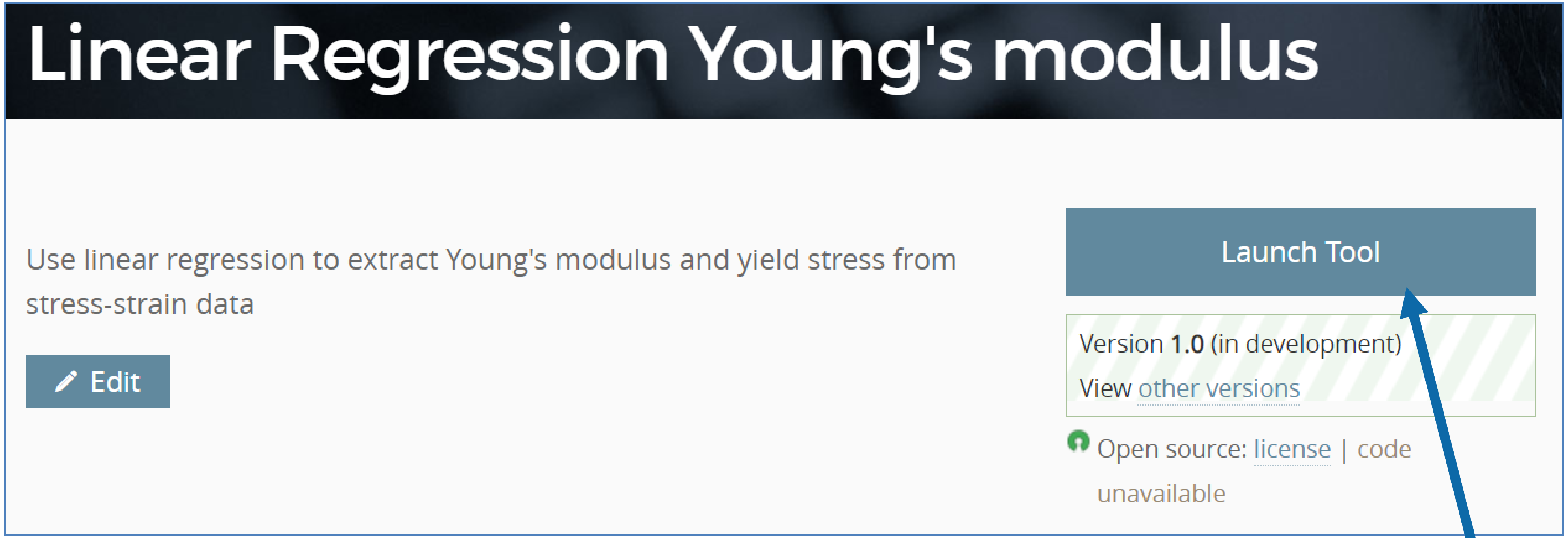
Yield stress
– amount of stress required to permanently deform a material
– typically use 0.2% offset rule to calculate (same Young's modulus value)
– important concept as the value depends on materials processing

- Stress vs. Strain have a linear relationship for small deformations
- Slope is known as Young's modulus (one of the elastic constants that describe the small deformation of materials)
- Yield stress measures where the material response deviates from linearity

Step 1: Launching a Jupyter tool in nanoHUB

Machine Learning for Materials Science: Part 1

From your browser go to link: https://nanohub.org/tools/youngsmod



Click on Launch Tool to begin

Step 2: Landing Page – Notebook: Linear Regression

Navigate to the first link in the landing page to access the notebook we will be working on during this exercise

Linear regression to obtain mechanical properties of metals

Michael Sakano, Saaketh Desai and Alejandro Strachan (strachan@purdue.edu)

School of Materials Engineering, Purdue University

1. Overview

The notebook liked below uses linear regression to extract materials properties from stress-strain data. Users will use Pandas dataframes to organize their data, us

2. Learning objectives

After going over this notebook you will be able to:

1. Use Pandas data structures to organize your data
2. Use linear regression to obtain Young's modulus and yield stress from stress-strain data
3. Explore how the fit parameters affect the results

3. Pre-requisites:

Basic Python programming: see Introduction to Jupyter Notebooks, Data Organization and Plotting by Juan C. Verduzco

4. Get started, the link below will take you to the notebook

Extracting Young's Modulus from stress-strain data using linear regression

This workflow uses linear regression from the Scikit-learn Python library to calculate the Young's modulus from a stress strain curves. Users can i) import their own *Stress-Strain Curves of a 70-30 Brass.* (1944).

- Use Pandas data structures to organize your data
- Apply regression techniques to create a linear model
- Develop a best fit curve from a dataset and obtain meaningful correlations

Important: To exit individual notebooks and return to this page, use File -> Close and Halt. "Terminate Session" (top right) will kill your entire Jupyter session.

Import several libraries we will use

```
import pandas as pd # This library is for developing data structures in the form of tables
import numpy as np # This library is for scientific operations and data manipulation in matrices
from sklearn import datasets, linear_model # This library helps to develop the linear model
from sklearn.metrics import mean_squared_error, r2_score # This library adds statistics to our model

import matplotlib.pyplot as plt # This library is for visualizing the curves
import plotly #This library is for visualizing advanced graphics
import plotly.graph_objs as go # This library is the graphical object for plotly
from plotly.offline import iplot # This library is necessary to run Plotly in Jupyter Notebooks, but not in a dedicated environment

import hublib.ui # This library is required for file read-in
from IPython.display import display # This library is required to show things like widgets

import os # This library provides access to the operating system commands. It is used to looking through datafiles in the current directory
```

Import scikit learn for linear regression

Documentation:

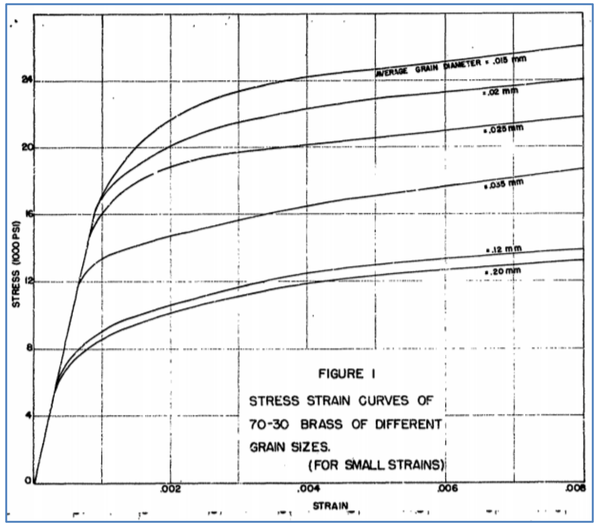
pandas: data analysis tool for creating data structures. See https://pandas.pydata.org/

numpy: scientific computing package with more common Python functionalities. See https://numpy.org/

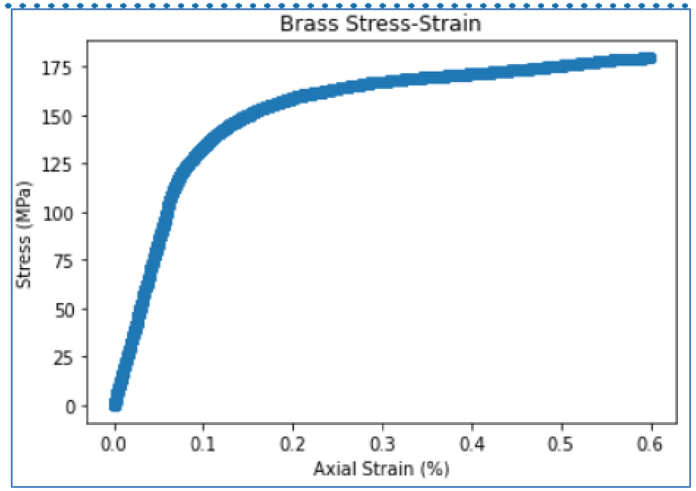
scikit-learn: tool package for predictive analysis. See https://scikit-learn.org/stable/

plotly: Graphical library for interactive plotting. See https://plotly.com/python/

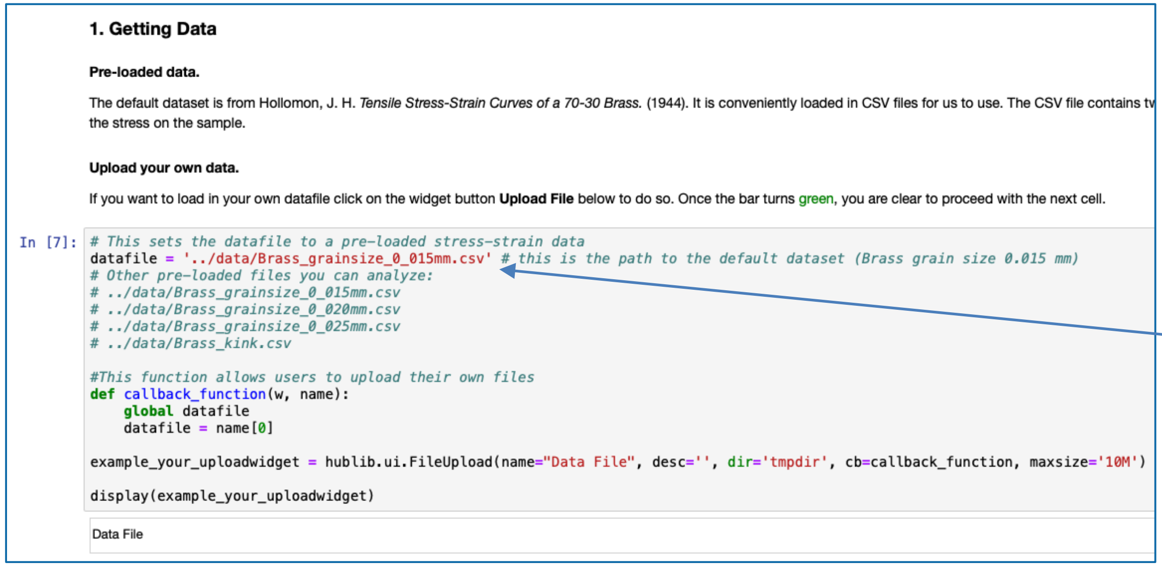
Step 1. Getting Data - Preloaded



Digitize



Hollomon, J. H. Tensile Stress-Strain Curves of a 70-30 Brass. (1944)



1. Getting Data

Pre-loaded data.

The default dataset is from Hollomon, J. H. *Tensile Stress-Strain Curves of a 70-30 Brass.* (1944). It is conveniently loaded in CSV files for us to use. The CSV file contains tv
the stress on the sample.

Upload your own data.

If you want to load in your own datafile click on the widget button **Upload File** below to do so. Once the bar turns green, you are clear to proceed with the next cell.

In [7]:

```
# This sets the datafile to a pre-loaded stress-strain data
datafile = '../data/Brass_grainsize_0_015mm.csv' # this is the path to the default dataset (Brass grain size 0.015 mm)
# Other pre-loaded files you can analyze:
# ../data/Brass_grainsize_0_015mm.csv
# ../data/Brass_grainsize_0_020mm.csv
# ../data/Brass_grainsize_0_025mm.csv
# ../data/Brass_kink.csv

#This function allows users to upload their own files
def callback_function(w, name):
    global datafile
    datafile = name[0]

example_your_uploadwidget = hublib.ui.FileUpload(name="Data File", desc='', dir='tmpdir', cb=callback_function, maxsize='10M')

display(example_your_uploadwidget)
```

Data File

Experimental data was digitized and pre-loaded

Set the pre-loaded data to be analyzed

Step 1. Getting Data – Upload your own data

```python
datafile = '../data/Brass_grainsize_0_015mm.csv' # this is the path to the default dataset (Brass grain size 0.015 mm)

def callback_function(w, name):
    global datafile
    datafile = name[0]

example_your_uploadwidget = hublib.ui.FileUpload(name="Data File", desc='', dir='tmpdir', cb=callback_function, maxsize='10M')

display(example_your_uploadwidget)
```

Data File Upload File

- You can upload your own data (csv format) - click on **Upload File**
- Find the data in your desktop
- Once the bar turns green, you are clear to proceed with the next cell.

```python
datafile = '../data/Brass_grainsize_0_015mm.csv' # this is the path to the default dataset (Brass grain size 0.015 mm)

def callback_function(w, name):
    global datafile
    datafile = name[0]

example_your_uploadwidget = hublib.ui.FileUpload(name="Data File", desc='', dir='tmpdir', cb=callback_function, maxsize='10M')

display(example_your_uploadwidget)
```

Data File Brass1.csv

Brass1.csv:

Step 2: Processing and Organizing the data

```
print(pd.read_csv(datafile)[:10])
```

```
         Strain  Stress (MPa)
0  0.000000e+00      0.000000
1  7.600000e-07      0.086268
2  1.520000e-06      0.172536
3  2.280000e-06      0.258803
4  3.040000e-06      0.345071
5  3.800000e-06      0.431339
6  4.560000e-06      0.517607
7  5.320000e-06      0.603875
8  6.080000e-06      0.690143
9  6.840000e-06      0.776410
```

```
# If you are uploading your own data, modify the column numbers if necessary.
# Otherwise for the default dataset, leave these variables alone

strain_column = 0 # the default strain column is 0
stress_column = 1 # the default stress column is 1

# In these sets of commands, we are using the pandas library to load the strain and stress columns of data into a dataframe
# We then convert each of them into separate arrays

df_strain_brass = pd.read_csv(datafile, usecols=[strain_column])
arr_strain_brass = np.array(df_strain_brass)*100

df_stress_brass = pd.read_csv(datafile, usecols=[stress_column])
arr_stress_brass = np.array(df_stress_brass)
```

Note the columns of data

Organize data into numpy arrays

```
strain_percentage = 0.05 # units of percent

# We define where in the list of strain datapoints we want to only process the linear (elastic) region. The default value is 0.1% strain.
end = np.where(arr_strain_brass > strain_percentage)[0][0]
arr_stress_brass_linear = arr_stress_brass[:end]
arr_strain_brass_linear = arr_strain_brass[:end]

# We pass the datapoints within the linear region as X and y training pairs
X_train = arr_strain_brass_linear
y_train = arr_stress_brass_linear
```

Define linear (elastic) region and push data into new numpy arrays

Step 4: Create linear model and train model

3. Creating a linear model

```python
def regression(x_train, y_train):

    # Define the model and train it
    model = linear_model.LinearRegression()
    model.fit(x_train, y_train)

    # Use the model to predict the entire set of data
    predictions = model.predict(x_train) # Make it for all values

    # Print model and mean squared error and variance score
    print("Linear Equation: %.4e X + (%.4e)"%(model.coef_, model.intercept_))
    print("Mean squared error: %.4e" % (mean_squared_error(y_train, predictions)))
    print('Variance score: %.4f' % r2_score(y_train, predictions))

    return predictions
```

Use scikit-learn to fit a linear model

Scikit-learn: https://scikit-learn.org/stable/

Print model parameters

5. Train models and plot results

```python
predictions = regression(X_train, y_train) # This line calls the Regression model implemented in the function

# This line plots the results from that model
plot(X_train, arr_strain_brass, y_train, arr_stress_brass, "Axial Strain (%)", "Stress (MPa)", predictions)

# NOTE: If the cell does not show the plot within 10-15 seconds, Rerun the cell !!!
```

Plot results

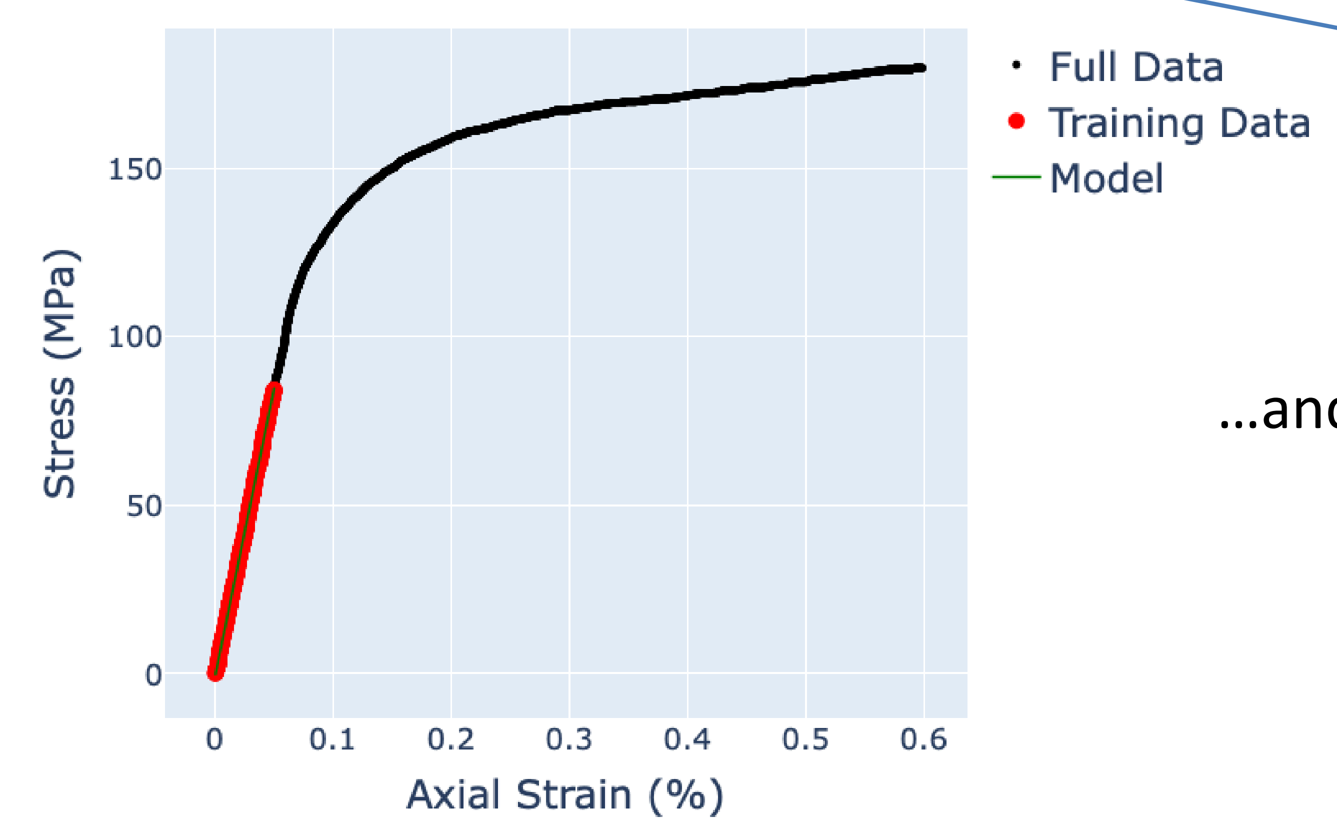
Analyze and plot results

```
In [15]: predictions = regression(X_train, y_train) # This line calls the Regression model implemented in the function

         # This line plots the results from that model
         plot(X_train, arr_strain_brass, y_train, arr_stress_brass, "Axial Strain (%)", "Stress (MPa)", predictions)

         # NOTE: If the cell does not show the plot within 10-15 seconds, Rerun the cell !!!

         Linear Equation: 1.7088e+03 X + (-5.0309e-01)
         Mean squared error: 2.0474e-01
         Variance score: 0.9997
```



You can extract the Young's modulus from the fit

...and visually assess your regression

Next steps

Homework assignment: to reinforce concepts and help students modify the workflow and adapt it for their needs

Including the calculation of yield stress using the 0.2% offset method