from collections import Counter

from itertools import islice

from argparse import ArgumentParser

import numpy as np

import matplotlib.pyplot as plt

def extract\_data\_points(fname):

# return a dict called 'data', such that

# data[s, i, j] = (d, f),

# where s is the series, i, j are the point coordinates;

# d is a numpy array containing measured distances,

# f is a numpy array containing measured forces.

data = dict()

with open(fname, 'rt') as ftext:

pass # TODO: extract s, i, j, distance (d) and force (f) measurements from the data

data[s, i, j] = (d, f) # this may have to be within some loop...

return data

def raw\_plot(point, curve, save=None, show=True):

"""plot one raw distance-force curve"""

# point is the triple (s, i, j) with series s, iIndex i, jIndex j

# curve is the pair (d, f) of two numpy arrays with distances and forces

d, f = curve

plt.figure(figsize=[9, 6])

pass # TODO: do an actually nice plot here with title, axis labels, legend, etc

plt.grid()

if save is not None:

plt.savefig(save, dpi=200, bbox\_inches='tight')

if show:

plt.show()

plt.close()

def do\_raw\_plots(data, show, plotprefix):

for point, curve in data.items():

s, i, j = point

print(f"plotting curve at {point}")

fname = f'{plotprefix}-{s:01d}-{i:03d}-{j:03d}.png' if plotprefix is not None else None

raw\_plot(point, curve, show=show, save=fname)

def main(args):

fname = args.textfile

print(f"parsing {fname}...")

full\_data = extract\_data\_points(fname)

if args.first is not None:

data = dict((k, v) for k, v in islice(full\_data.items(), args.first))

else:

data = full\_data

do\_raw\_plots(data, args.show, args.plotprefix)

def get\_argument\_parser():

p = ArgumentParser()

p.add\_argument("--textfile", "-t", required=True,

help="name of the data file containing AFM curves for many points")

p.add\_argument("--first", type=int,

help="number of curves to extract and plot")

p.add\_argument("--plotprefix", default="curve",

help="non-empty path prefix of plot files (PNGs); do not save plots if not given")

p.add\_argument("--show", action="store\_true",

help="show each plot")

return p

if \_\_name\_\_ == "\_\_main\_\_":

p = get\_argument\_parser()

args = p.parse\_args()

main(args)