import csv  
from numpy import \*  
from numpy import linalg as LA  
import numpy as np  
import os  
  
  
def file\_read(csv\_file\_name):  
 # file\_read reads file and enters it into an array  
 dir\_name = os.path.dirname(\_\_file\_\_)  
 file\_name = os.path.join(dir\_name, csv\_file\_name)  
 reader = csv.reader(open(file\_name), delimiter=",")  
 x\_matrix\_raw = array(list(reader)).astype("str")  
 return x\_matrix\_raw  
  
  
def p\_extraction\_lagrange\_multiplier(x\_matrix, x\_matrix\_transpose):  
 # p\_extraction\_lagrange\_multiplier returns p\_vector,  
 # the solution to the optimization of p(T)X(T)Xp such that norm(p) = 1  
 x\_transpose\_x = x\_matrix\_transpose @ x\_matrix # Matrix multiplication to determine XTransposeX matrix  
 eig\_values, eig\_vectors = LA.eig(x\_transpose\_x) # Eigvenvalues and Eigenvectors of XTX  
 max\_eig\_value = eig\_values.max()  
 eig\_values = list(eig\_values.real)  
 max\_eigen\_value\_index = eig\_values.index(max\_eig\_value)  
 p\_vector = np.array((eig\_vectors[:, max\_eigen\_value\_index]).real) # eigenvector of XTX with largest eigenvalue  
 return p\_vector, x\_matrix, x\_matrix\_transpose  
  
  
csvFileName = 'TEPdataProc1dataCSV.csv'  
XMatrixRaw = file\_read(csvFileName)  
XMatrix = np.array(delete(XMatrixRaw, 0, 0)) # removes headers of the data  
XMatrix = np.array(delete(XMatrix, XMatrix.shape[1] - 1, 1)) # removes last column with sample time data  
rows, cols = XMatrix.shape[0], XMatrix.shape[1]  
XMatrix = XMatrix.astype(np.float)  
XMatrixTranspose = np.array(XMatrix.transpose())  
tol = 1e-12 # set tolerance for relative error to make sure norm(p) == 1  
PCArray = []  
pVector, xMatrix, xMatrixTranspose = p\_extraction\_lagrange\_multiplier(XMatrix, XMatrixTranspose) # loading vector p, eigenvector for PC1  
tVector = xMatrix @ pVector # Matrix multiplication to determine latent score vector t  
normPVector = linalg.norm(pVector)  
relativeError = abs(1 - normPVector)  
count = 0  
  
while len(PCArray) < cols + 2:  
 new\_relative\_error = abs(1 - normPVector) #determines convergence criteria if the p vector gives the largest veriance  
 pVector = pVector.reshape(cols, 1)  
 pVectorTranspose = pVector.transpose()  
 tVector = np.resize(tVector, (rows, 1))  
 xMatrix = xMatrix - np.asarray(tVector @ pVectorTranspose) #Deflate xMatrix to extract next loading vector  
 xMatrixTranspose = xMatrix.transpose()  
 if count < 2:  
 PCArray.append(pVector.transpose())  
 count += 1  
 continue  
 else:  
 if new\_relative\_error < tol:  
 PCArray.append(pVector.transpose())  
 pVector, xMatrix, xMatrixTranspose = p\_extraction\_lagrange\_multiplier(xMatrix, XMatrixTranspose)  
 else:  
 print("Something went wrong, please check your data set again.")  
PCArray.pop(0)  
PCArray.pop(0)  
with open('PrincipleComponents.csv', 'w') as f:  
 for item in PCArray:  
 f.write(str(item) + ", \n\n")