```
#IMPORT MODULES
import numpy as np
import pandas as pd
from math import factorial
import matplotlib.pyplot as plt

r"""
PARAMETERS
window_size : Window lenght. Must be an odd integer number.
order : Polynomial order used in the filtering. Must be less then `window_size`
- 1.
deriv: Derivative order to compute (default = 0 means only smoothing)
"""

#READING DATA
def read_data():
 data = np.array(pd.read_csv(r'C:\Users\edwincaballero\Desktop\data.csv')) #
Cambiar a csv con 3 espectros
 return data

#SAVING DATA
def write_sg_output(sg_data):
 sg_output_writer = r'C:\Users\edwincaballero\Desktop\sg_results.csv'
 pd.DataFrame(sg_data).to_csv(sg_output_writer, index=False)

#APPLY SNV ON DATA
def sg(input_data, window_size, order, deriv, rate=1):
 #CONVERT WINDOW SIZE IN INTEGER AND AS A POSITIVE VALUE
 window_size = np.abs(np.int(window_size))
 #CONVERT ORDER IN INTEGER AND AS A POSITIVE VALUE
 order = np.abs(np.int(order))
 #INPUT VALUES
 order_range = range(order+1)

 half_window = (window_size -1) // 2
 #PRECOMPUTE COEFFICIENTS
 output_data = np.zeros_like(input_data)

 for i in range(input_data.shape[0]):
 #INTERPRET 'b' AS A MATRIX
 b = np.mat([[k**i for i in order_range] for k in range(-
half_window, half_window+1)])
 #COMPUTING MOORE-PENROSE PSEUDO-INVERSE OF A MATRIX BY ITS SVD
 m = np.linalg.pinv(b).A[deriv] * rate**deriv * factorial(deriv)
 #Pad the signal at the extremes with values taken from the signal itsel
f
 firstvals = input_data[0] - np.abs(input_data[1:half_window+1][::-
1] - input_data[0])
 lastvals = input_data[-1] + np.abs(input_data[-half_window-1:-1][::-
1] - input_data[-1])
 #JOIN SEQUENCY OF ARRAYS ALONG AN EXISTING AXIS
 input_data = np.concatenate((firstvals, input_data, lastvals))

 #RETURN DISCRETE, LINEAR CONVOLUTION OF TWO 1-D SEQUENCES
 output_data = np.convolve(m[::-1], input_data, mode='valid')

 return output_data

def sg_grupal(input_data_grupal):
 output_grupal = []
 for input_data in input_data_grupal:
 out_put = sg(input_data, window_size, order, deriv, rate=1)
 output_grupal.append(out_put)
 return output_grupal

def plot(input_data, output_data):
 plt.plot(t, y, label='Noisy signal')
 plt.plot(t, np.exp(-t**2), 'k', lw=1.5, label='Original signal')
 plt.plot(t, ysg, 'r', label='Filtered signal')
 plt.legend()
 plt.show()

input_data = read_data()
window_size = 7
order = 2
deriv = 2

sg_data = sg_grupal(input_data)
write_sg_output(sg_data)
count = len(sg_data)
print()
print('SUCCESS')
print(str(count)+ ' spectra transformed with Savitzky-Golay algorithm')
```