Abby Ortego
W0716476
CMPS 473 – 01

Program 1.5 Data Pre-Processing

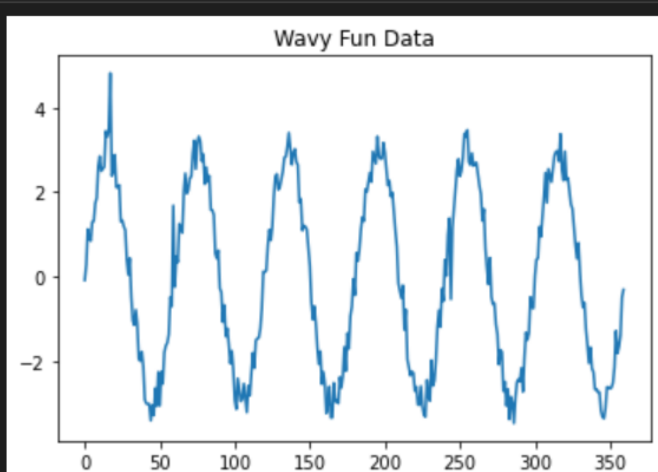
Imports, Reading Data, and Plotting Function for Set-Up

```
Run Cell | Run Below | Debug Cell | Go to [38]
# %% IMPORTS
import matplotlib.pyplot as plt
import numpy as np
import copy

Run Cell | Run Above | Debug Cell | Go to [39]
# %% READ DATA
with open("wavy fun.txt") as myFile:
    wavyFunData = [float(line.strip('\n')) for line in myFile]
#

Run Cell | Run Above | Debug Cell | Go to [40]
# %% FUNCTIONS
def plot_data(data,title):
    plt.plot(data)
    plt.title(title)
    plt.show()
# plot_data

```

(1) Plot the Data & Output

```
Run Cell | Run Above | Debug Cell | Go to [4]
# %% (1) PLOT DATA
plot_data(wavyFunData, 'Wavy Fun Data')

```

```
✓ plot_data(wavyFunData, 'Wavy Fun Data') ···
```



Abby Ortego

W0716476

CMPS 473 – 01

Program 1.5 Data Pre-Processing

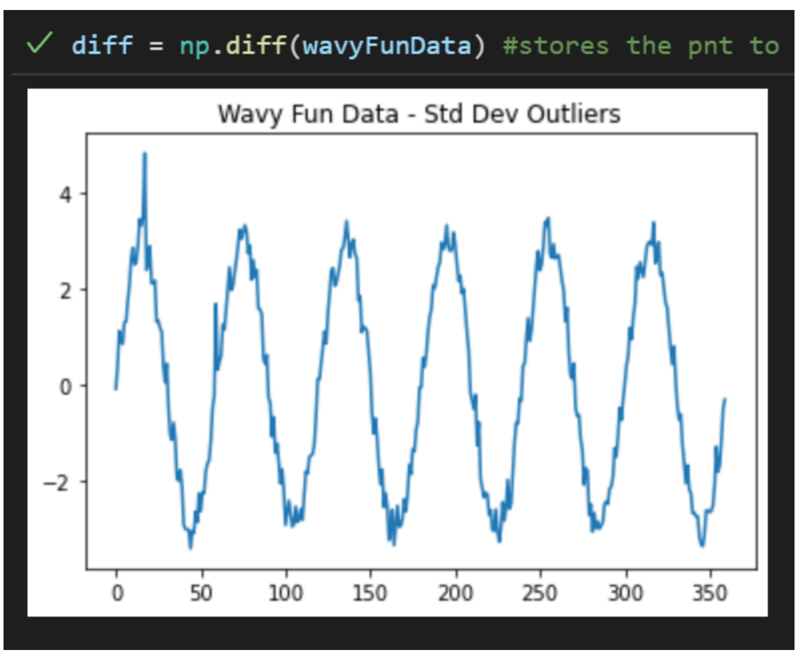
(2) Finding Outliers via Observation

There is a lot of noise over the span of the graph but particularly large outliers in the 25-75 and 225 – 275 ranges.

(3) Smoothing via Standard Deviation

Run Cell | Run Above | Debug Cell

```
# %% (3) FIND OUTLIERS VIA STD. DEV.
diff = np.diff(wavyFunData) #stores the pnt to pnt differences
diff_dev = diff.std() #stores the std dev for those differences
wavyFunData_stdDev = copy.deepcopy(wavyFunData) #deep copy so that wavyFunData stays the same

for i in range(len(diff)):
    #if the difference is greater than the allowed std dev...
    if((diff[i] > diff_dev)):
        #...find its 6 closest wavyFunData neighbors (3 on the left and 3 on the right)...
        neighbors = [wavyFunData[pnt] for pnt in range(i-2,i+3)]
        #...get their average and replace that point at wavyFunData with it.
        wavyFunData_stdDev[i] = np.average(neighbors)
    #
#

#plot for reference / comparison
plot_data(wavyFunData_stdDev, 'Wavy Fun Data - Std Dev Outliers')

```



Abby Ortego
W0716476
CMPS 473 – 01

Program 1.5 Data Pre-Processing

(4) Smoothing via Sliding Windows – Simple & Weighted Averages

```python
# %% (4.1) SMOOTH OUTLIERS VIA SLIDING WINDOW - SIMPLE AVG
'''5 POINT SIMPLE AVG'''
j = 2 #start at 2 for a 5 point simple avg
wavyFunData_window5_simple = copy.deepcopy(wavyFunData)
#while loop prevents array out of bound error
while(j >= 2 and j <= (len(wavyFunData)-3)):
    #window is 2 pts before and after j - the target pt to be replaced
    window_5_simple = [wavyFunData[pnt] for pnt in range(j-2, j+3)]
    wavyFunData_window5_simple[j] = np.average(window_5_simple)
    j = j + 1
#
#plot for reference / comparison
plot_data(wavyFunData_window5_simple, 'Wavy Fun Data - Sliding Window Simple Avg (5 points)')


'''7 POINT SIMPLE AVG'''
j = 3 #start at 3 for a 7 point simple avg
wavyFunData_window7_simple = copy.deepcopy(wavyFunData)
#while loop prevents array out of bound error
while(j >= 3 and j <= (len(wavyFunData)-4)):
    #window is 3 pts before and after j - the target pt to be replaced
    window_7_simple = [wavyFunData[pnt] for pnt in range(j-3, j+4)]
    wavyFunData_window7_simple[j] = np.average(window_7_simple)
    j = j + 1
#
#plot for reference / comparison
plot_data(wavyFunData_window7_simple, 'Wavy Fun Data - Sliding Window Simple Avg (7 points)')

'''9 POINT SIMPLE AVG'''
j = 4 #start at 4 for a 9 point simple avg
wavyFunData_window9_simple = copy.deepcopy(wavyFunData)
#while loop prevents array out of bound error
while(j >= 4 and j <= (len(wavyFunData)-5)):
    #window is 4 pts before and after j - the target pt to be replaced
    window_9_simple = [wavyFunData[pnt] for pnt in range(j-4, j+5)]
    wavyFunData_window9_simple[j] = np.average(window_9_simple)
    j = j + 1
#
#plot for reference / comparison
plot_data(wavyFunData_window9_simple, 'Wavy Fun Data - Sliding Window Simple Avg (9 points)')
```

Abby Ortego

W0716476

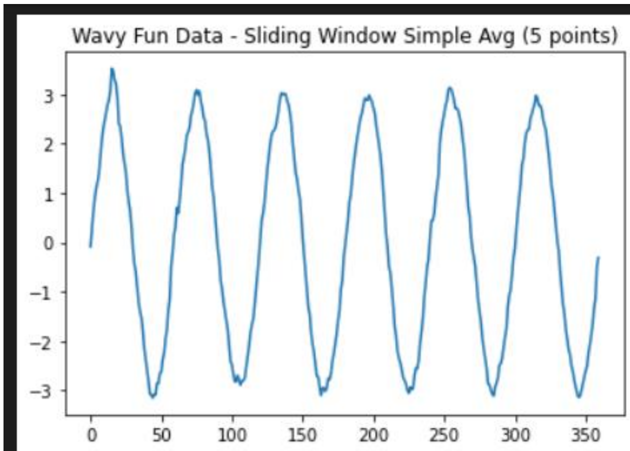
CMPS 473 – 01

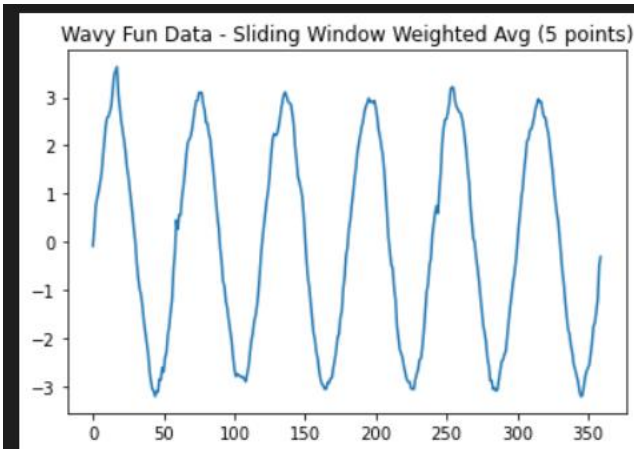
Program 1.5 Data Pre-Processing

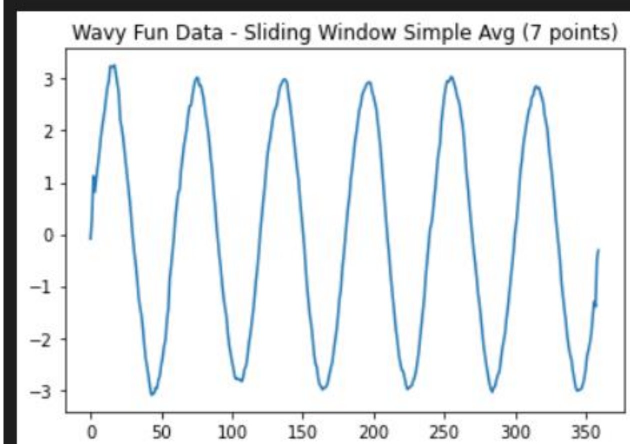
Run Cell | Run Above | Debug Cell

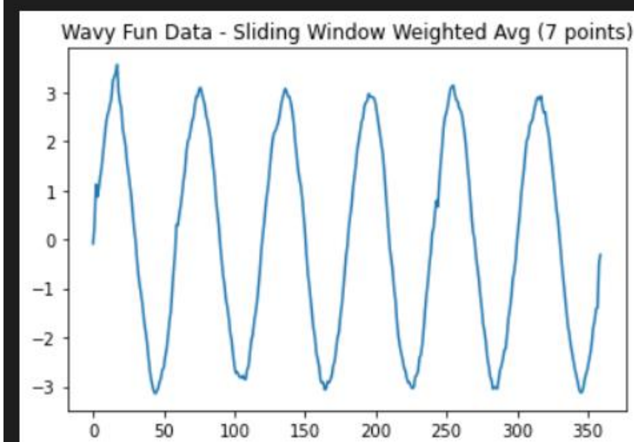
```python
# %% (4.2) SMOOTH OUTLIERS VIA SLIDING WINDOW - WEIGHTED AVG
'''5 POINT WEIGHTED AVG'''
j = 2 #start at 2 for a 5 point weighted avg
weights_5 = [1/3, 1/2, 1, 1/2, 1/3]
wavyFunData_window5_weighted = copy.deepcopy(wavyFunData)
#while loop prevents array out of bound error
while(j >= 2 and j <= (len(wavyFunData)-3)):
    #window is 2 pts before and after j - the target pt to be replaced
    window_5_weighted = [wavyFunData[pnt] for pnt in range(j-2, j+3)]
    wavyFunData_window5_weighted[j] = np.average(window_5_weighted, weights = weights_5)
    j = j + 1
#
#plot for reference / comparison
plot_data(wavyFunData_window5_weighted, 'Wavy Fun Data - Sliding Window Weighted Avg (5 points)')

'''7 POINT WEIGHTED AVG'''
j = 3 #start at 3 for a 7 point weighted avg
weights_7 = [1/4, 1/3, 1/2, 1, 1/2, 1/3, 1/4]
wavyFunData_window7_weighted = copy.deepcopy(wavyFunData)
#while loop prevents array out of bound error
while(j >= 3 and j <= (len(wavyFunData)-4)):
    #window is 3 pts before and after j - the target pt to be replaced
    window_7_weighted = [wavyFunData[pnt] for pnt in range(j-3, j+4)]
    wavyFunData_window7_weighted[j] = np.average(window_7_weighted, weights = weights_7)
    j = j + 1
#
#plot for reference / comparison
plot_data(wavyFunData_window7_weighted, 'Wavy Fun Data - Sliding Window Weighted Avg (7 points)')

'''9 POINT WEIGHTED AVG'''
j = 4 #start at 4 for a 9 point weighted avg
weights_9 = [1/5, 1/4, 1/3, 1/2, 1, 1/2, 1/3, 1/4, 1/5]
wavyFunData_window9_weighted = copy.deepcopy(wavyFunData)
#while loop prevents array out of bound error
while(j >= 4 and j <= (len(wavyFunData)-5)):
    #window is 4 pts before and after j - the target pt to be replaced
    window_9_weighted = [wavyFunData[pnt] for pnt in range(j-4, j+5)]
    wavyFunData_window9_weighted[j] = np.average(window_9_weighted, weights = weights_9)
    j = j + 1
#
#plot for reference / comparison
plot_data(wavyFunData_window9_weighted, 'Wavy Fun Data - Sliding Window Weighted Avg (9 points)')
```

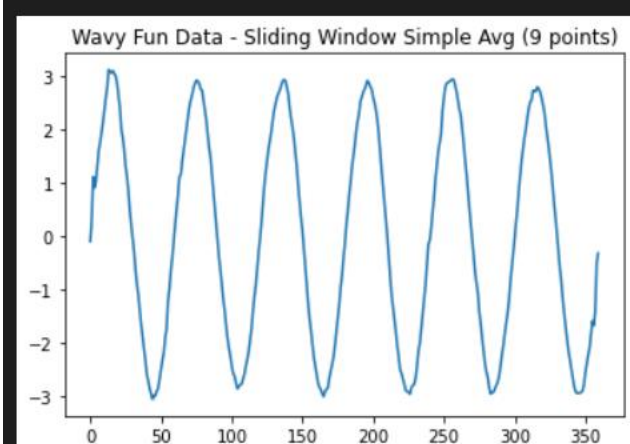
Abby Ortego
W0716476
CMPS 473 – 01
Program 1.5 Data Pre-Processing

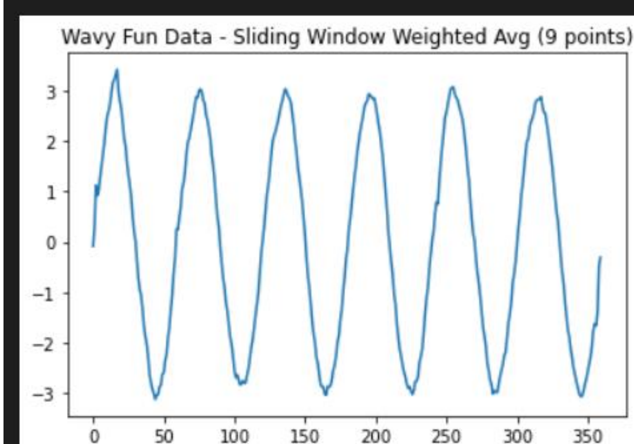












Abby Ortego

W0716476

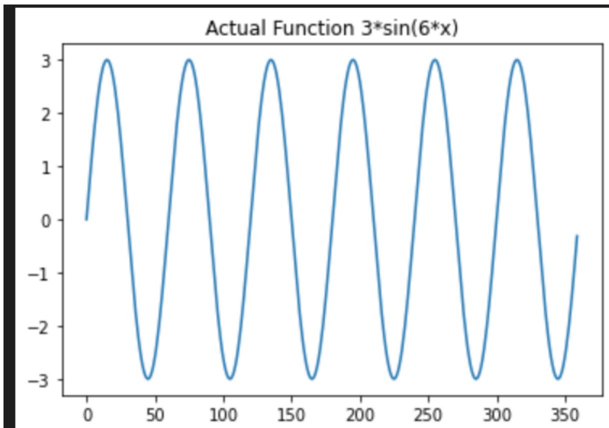
CMPS 473 – 01

Program 1.5 Data Pre-Processing

(5) Error for Sliding Windows & Original Function

Run Cell | Run Above | Debug Cell

```
# %% (5) ERROR FOR SLIDING WINDOWS
''' Formula 3*sin(6*x) '''
actual_fun = [3*np.sin(6*(x*(np.pi / 180))) for x in range(0,360)] #where (np.pi / 180) is the sampling rate
plot_data(actual_fun, 'Actual Function 3*sin(6*x)')

print("\nAverage Error for smoothing via Sliding Windows & Simple Avg...")
print("\tWindow Size 5: ", (abs(np.subtract(wavyFunData_window5_simple, actual_fun)).mean()) )
print("\tWindow Size 7: ", (abs(np.subtract(wavyFunData_window7_simple, actual_fun)).mean()) )
print("\tWindow Size 9: ", (abs(np.subtract(wavyFunData_window9_simple, actual_fun)).mean()) )

print("\nAverage Error for smoothing via Sliding Windows & Weighted Avg...")
print("\tWindow Size 5: ", (abs(np.subtract(wavyFunData_window5_weighted, actual_fun)).mean()) )
print("\tWindow Size 7: ", (abs(np.subtract(wavyFunData_window7_weighted, actual_fun)).mean()) )
print("\tWindow Size 9: ", (abs(np.subtract(wavyFunData_window9_weighted, actual_fun)).mean()) )
```




```
Average Error for smoothing via Sliding Windows & Simple Avg...
        Window Size 5:  0.10577260316091909
        Window Size 7:  0.09619764428510691
        Window Size 9:  0.10094913474637504

Average Error for smoothing via Sliding Windows & Weighted Avg...
        Window Size 5:  0.11508751317949202
        Window Size 7:  0.10224798763903677
        Window Size 9:  0.09916010492557675
```

According to the error calculations, using more neighboring points allows for not only a smoother function but also less error. However, using more neighboring points does have its limitations. The simple average seems to have less error on average when the window size remains around 7 points versus 9 points. Using weighted average, the opposite seems to be true where using 9 neighboring points lessens the error on average while still smoothing the function. **The best technique for this data set in terms of error and overall smoothness would be the sliding window with window size 7 using simple average.**