jupyter Assignment-I-Darsh Last Checkpoint: a few seconds ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

Least Square based approach for Signal Denoising

Presented By: Darshana Subhash

One approach to denoise the signal is based on **least squares weighted regularization**. The smoothness of a signal can be measured by the energy of its derivative (second order derivative(**D**)). The smoother a signal is, the smaller the energy of its derivative is.

Objective is to denoise the signal without loosing the orignal signal charecteristics.

$$D = \begin{bmatrix} 1 & -2 & 1 & & & \\ & 1 & -2 & 1 & & \\ & & \ddots & & \ddots & \\ & & & 1 & -2 & 1 \end{bmatrix}$$

D is the second order difference matrix with **diagonal loadings**.Therefore, solution is obtained using fast solvers for banded systems.

$$\arg\min_x \|y - x\|_2^2 + \lambda \|Dx\|_2^2$$

$$\frac{df(x,y)}{dx} = -y + (I + \lambda D^T D)x = 0$$

$$x = (I + \lambda D^T D)^{-1} y$$

Signal to Noise Ratio: Computed as mentioned below

$$\mathrm{SNR_{dB}} = 10 \log_{10}\left(\frac{P_{\mathrm{signal}}}{P_{\mathrm{noise}}}\right)$$

$$\mathrm{SNR_{dB}} = 10 \log_{10}\left(\frac{rms(AVG_{\mathrm{signal}})}{rms(AVG_{\mathrm{noise}})}\right)$$

Peak signal to noise ratio: Computed as mentioned below

$$PSNR = 10 * \log_{10}\left(\frac{MAX_I^2}{MSE}\right)$$

$$10 * \log_{10}\left(\frac{MAX_I^2}{\sqrt{MSE}}\right)$$

$$20 * \log_{10}(MAX_I) - 20 * \log_{10}\left(\sqrt{MSE}\right)$$

Q1. In the least square based signal denoising, change the control parameter from 0 to 1 with the step size =0.1 and evaluate the performance of the algorithm using PSNR and SNR. Plot the variation of PSNR and SNR with the control parameter.

Justification:

- Effectiveness of Denoised signal to be smooth and similar to Orignal /reference signal is traded with the varying lamda(λ) value.
- By increasing the λ value or the control parameter with the step size, results into an exponential growth curve or higher SNR/PSNR values depending on the degree of derivative.
- It is evident from the performance of the PSNR and SNR plots that denoising technique using the **"least square weighted regularization"** reduces the noise from the orignal Noisy Signal.

In [25]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from scipy.sparse import diags as spdiags
from numpy.linalg import matrix_rank as rank
from numpy.random import randn as randn
from pandas import DataFrame
```

In [26]:
```
read_file=pd.read_csv(r'data.txt',header=None)
read_file.to_csv(r'data.csv',index=None)
Orignal_Signal=pd.read_csv('data.csv')
N=len(Orignal_Signal)
Noisy_Signal=Orignal_Signal+(0.1*randn(300,1))
```

```
0
```

In [3]:
```
data2=np.array([1,-2,1])
data1=np.array([1,-1])

diags2=np.array([0,1,2])
diags1=np.array([0,1])

D_2=spdiags(data2,diags2,shape=(N-2,N)).toarray()
D_1=spdiags(data1,diags1,shape=(N-2,N)).toarray()



print("First five corner of Second order derivative matrix is:\n",D_2[:5,:5])
```

```
First five corner of Second order derivative matrix is:
 [[ 1. -2.  1.  0.  0.]
 [ 0.  1. -2.  1.  0.]
 [ 0.  0.  1. -2.  1.]
 [ 0.  0.  0.  1. -2.]
 [ 0.  0.  0.  0.  1.]]
```

In [4]:
```
def Min_x(Noisy_Signal,lamda,deg):
    banded_matrix = (np.identity(N))+(lamda*(deg.T@deg))
    Smooth_Signal=np.linalg.inv(banded_matrix)@Noisy_Signal
    return(Smooth_Signal)
```

In [5]:
```
def plot_Orignal():
    plt.subplot(211)
    plt.title('Orignal Signal')
    plt.plot(Orignal_Signal)
    plt.grid(True)
```

```
    plt.xlabel('Sample_Number')
    plt.ylabel('Amplitude')
    plt.show()
def plot_Noisy():
    plt.subplot(212)
    plt.title('Noisy Signal')
    plt.plot(Noisy_Signal)
    plt.grid(True)
    plt.xlabel('Sample_Number')
    plt.ylabel('Amplitude')
    plt.show()
def plot_Smooth(lamda,deg):
    Smooth_Signal=Min_x(Noisy_Signal,lamda,deg)
    plt.subplot(212)
    plt.title('Denoised Signal'+"(lamda:"+str(lamda)+')')
    plt.plot(Smooth_Signal)
    plt.grid(True)
    plt.xlabel('Sample_Number')
    plt.ylabel('Amplitude')
    plt.show()
    return Smooth_Signal
```

In [6]:
```
from math import log10,sqrt
def PSNR(Orignal,Denoised,Peak_value_signal):
    MSE=np.mean((Orignal - Denoised) ** 2)
    if(MSE==0):
        pass
    else:
        psnr=20*log10(Peak_value_signal/sqrt(MSE))
        return psnr
```

In [7]:
```
def PSNR_Compute(deg,Orignal_Signal,Noisy_Signal):
    plot_Orignal()
    plot_Noisy()

    L=np.arange(0,1.1,0.1)
    List=[]
    for i in L:
        Denoised_Signal=Min_x(Noisy_Signal,i,deg)
        y=np.array(Orignal_Signal)
        z=np.array(Denoised_Signal)
        peak_value_signal=max(y)
        PSNR_RES=PSNR(y,z,peak_value_signal)
        List.append(PSNR_RES)

    return L,List
```

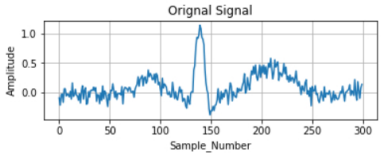
In [8]:
```
def Performance_PSNR(D):
    L,List=PSNR_Compute(D,Orignal_Signal,Noisy_Signal)
    plot_Smooth(1.0,D)
    plt.subplot(211)
    plt.title('Performance')
    plt.xlabel('Lamda')
    plt.ylabel('PSNR')
    plt.plot(L,List)
    plt.grid(True)
    plt.show
    return List
```

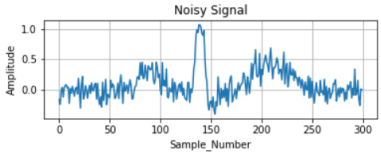
In [9]:
```
def SNR(Orignal,Denoised):
    PSignal=np.sum(np.square(Orignal))/len(Orignal)
    PNoise=np.sum(np.square(Denoised-Orignal))/len(Denoised)
    snr=(10*np.log10(PSignal))-(10*np.log10(PNoise))
    return snr
```

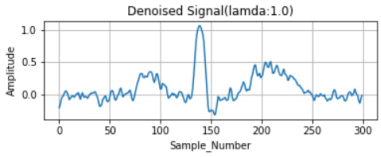
In [10]:
```
def SNR_Compute(deg,Orignal_Signal,Noisy_Signal):
    plot_Orignal()
    plot_Noisy()
    L=np.arange(0,1.1,0.1)
    List=[]
    for i in L:
        Denoised_Signal=Min_x(Noisy_Signal,i,deg)
        y=np.array(Orignal_Signal)
        z=np.array(Denoised_Signal)
        peak_value=max(Denoised_Signal)
        SNR_RES=SNR(y,z)
        List.append(SNR_RES)
    return L,List
```

In [11]:
```
def Performance_SNR(D):
    L,List=SNR_Compute(D,Orignal_Signal,Noisy_Signal)
    plot_Smooth(1.0,D)
    plt.subplot(211)
    plt.title('Performance')
    plt.xlabel('Lamda')
    plt.ylabel('SNR')
    plt.plot(L,List)
    plt.grid(True)
    plt.show
    return List
```

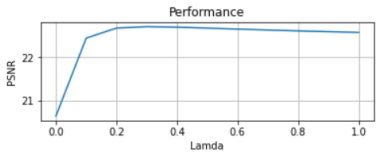
In [12]:
```
List1=Performance_PSNR(D_2)
print("Highest PSNR value acheieved for lamda at 1.0 for 2nd order derivative is:",List1[10])
```



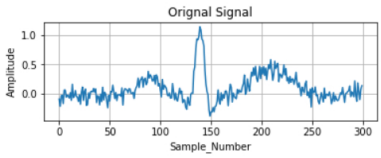


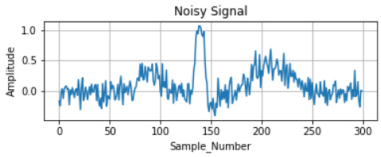


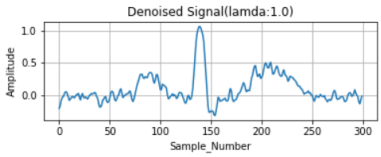

```
Highest PSNR value acheieved for lamda at 1.0 for 2nd order derivative is: 22.583758815437676
```



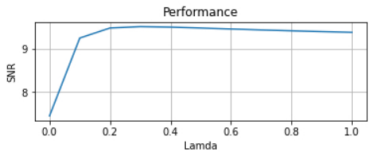
In [13]:
```
List1=Performance_SNR(D_2)
print("Highest SNR value acheieved for lamda at 1.0 for 2nd order derivative is:",List1[10])
```








```
Highest SNR value acheieved for lamda at 1.0 for 2nd order derivative is: 9.39162817626688
```



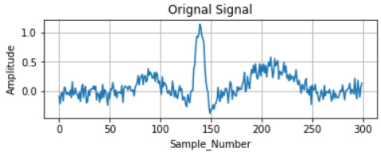
Q2. In the least square based signal denoising, change the D matrix from second-order to first-order and repeat the experiment mentioned in **Q.No.1.** Report your observations.

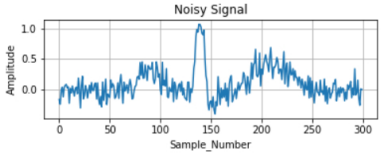
Justification:

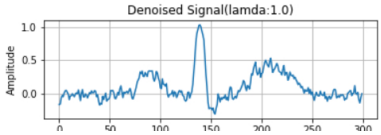
- The smoothness of the signal can be measured by the energy of its **derivative**.
- With the increase in smoothness,energy decreases and vice-versa. This can be proved with the performance of PSNR/SNR plots.
- Changing the D matrix from second-order to first-order,**SNR value has increased** from **9.648dB to 10.325dB**,which indicates that removal of noise from the orignal noisy signal gradually dropped in turn affecting the smoothness of the signal.

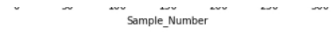
In [14]:
```
print("First five corner of First-order derivative matrix is:\n",D_1[:5,:5])
List1=Performance_SNR(D_1)
print("Highest SNR value acheieved for lamda at 1.0 for 1st order derivative is:",List1[10])
```

```
First five corner of First-order derivative matrix is:
 [[ 1. -1.  0.  0.  0.]
 [ 0.  1. -1.  0.  0.]
 [ 0.  0.  1. -1.  0.]
 [ 0.  0.  0.  1. -1.]
 [ 0.  0.  0.  0.  1.]]
```

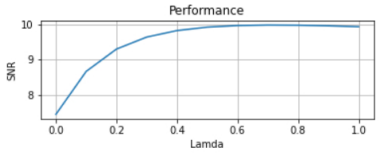









```
Highest SNR value acheieved for lamda at 1.0 for 1st order derivative is: 9.931380713911357
```



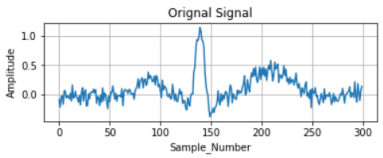
Q3. In the least square based signal densoing, change the D matrix from second order to third order and repeat the experiment mentioned in **Q.No.1.** Report your observations.

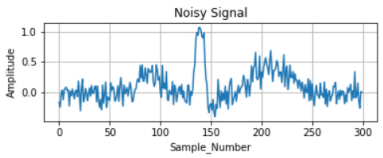
Justification:

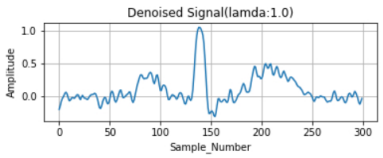
- Changing the matrix to higher order derivative from lower order derivative increases the smoothness in turn reducing the energy values.This also indicates **information loss**.
- As we see from the plots that SNR value has decreased from **9.648dB to 9.51dB** indicating the amount of noise removed from the signal and the peak value drop compared to Denoised signal of second order derivative
- It is widely appreciated that differentiation degrades the signal-to-noise ratio (SNR)

In [15]:
```
data3=np.array([1,-3,3,-1])
diags3=np.array([0,1,2,3])
D_3=spdiags(data3,diags3,shape=(N-2,N)).toarray()
print("First five corner of third-order derivative matrix is:\n",D_3[:5,:5])
List1=Performance_SNR(D_3)
print("Highest SNR value acheieved for lamda at 1.0 for 3rd order derivative is:",List1[10])
```

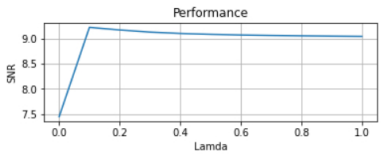
```
First five corner of third-order derivative matrix is:
 [[ 1. -3.  3. -1.  0.]
 [ 0.  1. -3.  3. -1.]
 [ 0.  0.  1. -3.  3.]
 [ 0.  0.  0.  1. -3.]
 [ 0.  0.  0.  0.  1.]]
```








```
Highest SNR value acheieved for lamda at 1.0 for 3rd order derivative is: 9.041257668814685
```



Q4. Generate the signal with the noise uisng the following lines of matlab code.

t=linspace(0,100,1000); x=sin(2*pi*50*t); plot(t,x); grid('on'); xn=x+0.05*randn(1,1000);

Identify the parameter to change the noise level of the signal. Increase the noise level of the signal. Repeat the experiment mentioned in Q.No.1 for the current noise level and the higher noise level you opted. Report your observations..

Justification:

- By increasing the noise level of the signal, PSNR and SNR shoots up.Hence, PSNR and SNR values are relatively high compared to before.
- PSNR value increased from 0.0086 to 25DB.

In [31]:
```
t=np.linspace(0,100,1000)
x=np.sin(2*(np.pi)*50*t)
alpha=0.05
xn=x+alpha*randn(1000)
Orignal_Signal=x
Noisy_Signal=xn
N=len(Orignal_Signal)
data=np.array([1,-2,1])
diags=np.array([0,1,2])
D_2=spdiags(data,diags,shape=(N-2,N)).toarray()


data1=np.array([1,-1])
diags1=np.array([0,1])

D_1=spdiags(data1,diags1,shape=(N-2,N)).toarray()

data3=np.array([1,-3,3,-1])
diags3=np.array([0,1,2,3])

D_3=spdiags(data1,diags1,shape=(N-2,N)).toarray()
```

In [ ]:
```
List1=Performance_SNR(D_2)
print("Highest SNR value acheieved for lamda at 1.0 for 2nd order derivative is:",List1[10])
```

Increasing the noise level of the signal by changing the value of control parameter **alpha** from 0.05 to 0.09

In [29]:
```
alpha=0.09
xn=x+alpha*randn(1000)
Orignal_Signal=x
Noisy_Signal=xn
```

In [ ]:
```
List2=Performance_SNR(D_2)
print("Highest SNR value acheieved for lamda at 1.0 for 2nd order derivative is:",List2[10])
```

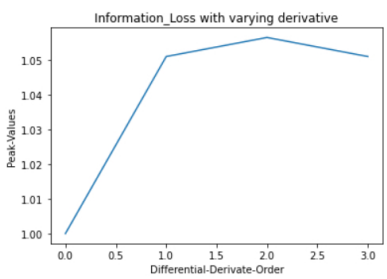
In [ ]:
```
L=np.arange(0.0,1.1,0.1)
plt.figure()
plt.title('SNR Vs noise level')
plt.xlabel('Noise_level')
plt.ylabel('SNR')
plt.plot(L,List1,'b')
plt.plot(L,List2,'r')
plt.legend(['Noise level=0.05','Noise level=0.09'])
```

NOTE: Its visible from the above plots that SNR decreases with increasing noise level.Its dependent on the derivative and the lamda values which are same here
As lamda is responsible for smoothening the signal and derivative is responsible for retaining the energy of the signal,increased amount of noise in the signal results in SNR drop indicating that there is more unwanted noise than the actual information.Hence, SNR value should be high for **information gain**.

In [33]:
```
def Peak_Value(deg,Orignal_Signal,Noisy_Signal):
    L=np.arange(0.0,1.1,0.1)
    Denoised_Signal=Min_x(Noisy_Signal,max(L),deg)
    peak_value=max(Denoised_Signal)
    return peak_value

Peak_Value0=max(Orignal_Signal)
print(Peak_Value0)
Peak_Value1=Peak_Value(D_1,Orignal_Signal,Noisy_Signal)
print(Peak_Value1)
Peak_Value2=Peak_Value(D_2,Orignal_Signal,Noisy_Signal)
print(Peak_Value2)
Peak_Value3=Peak_Value(D_3,Orignal_Signal,Noisy_Signal)
print(Peak_Value3)
D=[0,1,2,3]
Peak_Values=[Peak_Value0,Peak_Value1,Peak_Value2,Peak_Value3]
plt.figure()
plt.title('Information_Loss with varying derivative')
plt.xlabel('Differential-Derivate-Order')
plt.ylabel('Peak-Values')
plt.plot(D,Peak_Values)
plt.show()
```

```
0.9999987638285975
1.0
1.0509314551941031
1.0
1.0564112822819332
1.0
1.0509314551941031
```



In [ ]: