**DAILY ASSESSMENT FORMAT**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date:** | **28-May-2020** | **Name:** | **Raziya Banu** |
| **Course:** | **DSP** | **USN:** | **4AL16EC058** |
| **Topic:** | **Implementation of signal Filtering signal using WT in MatLAb** | **Semester & Section:** | **8th sem & ‘B’ section** |
| **Github Repository:** |  |  |  |

|  |
| --- |
| **FORENOON SESSION DETAILS** |
| **Image of session** |
| **Report –**  In my first session today I have studied about the DSP Implementation of signal Filtering signal using WT in MatLAb Lowpass FIR Filter – Window Method This example shows how to design and implement an FIR filter using two command line functions, fir1 and designfilt, and the interactive **Filter Designer** app.  Create a signal to use in the examples. The signal is a 100 Hz sine wave in additivewhite Gaussian noise. Set the random number generator to the default state for reproducible results.  rng default  Fs = 1000;  t = linspace(0,1,Fs);  x = cos(2\*pi\*100\*t)+0.5\*randn(size(t));  The filter design is an FIR lowpass filter with order equal to 20 and a cutoff frequency of 150 Hz. Use a Kaiser window with length one sample greater than the filter order and. See kaiser for details on the Kaiser window.  Use fir1 to design the filter. fir1 requires normalized frequencies in the interval [0,1], where 1 corresponds torad/sample. To use fir1, you must convert all frequency specifications to normalized frequencies.  Design the filter and view the filter's magnitude response.  fc = 150;  Wn = (2/Fs)\*fc;  b = fir1(20,Wn,'low',kaiser(21,3));  fvtool(b,1,'Fs',Fs)  https://in.mathworks.com/help/examples/signal/win64/FilteringDataWithSignalProcessingToolboxSoftwareExample_01.png  Apply the filter to the signal and plot the result for the first ten periods of the 100 Hz sinusoid.  y = filter(b,1,x);  plot(t,x,t,y)  xlim([0 0.1])  xlabel('Time (s)')  ylabel('Amplitude')  legend('Original Signal','Filtered Data')  https://in.mathworks.com/help/examples/signal/win64/FilteringDataWithSignalProcessingToolboxSoftwareExample_02.png  Design the same filter using designfilt. Set the filter response to 'lowpassfir' and input the specifications as Name,Value pairs. With designfilt, you can specify your filter design in Hz.  Fs = 1000;  Hd = designfilt('lowpassfir','FilterOrder',20,'CutoffFrequency',150, ...  'DesignMethod','window','Window',{@kaiser,3},'SampleRate',Fs);  Filter the data and plot the result.  y1 = filter(Hd,x);  plot(t,x,t,y1)  xlim([0 0.1])  xlabel('Time (s)')  ylabel('Amplitude')  legend('Original Signal','Filtered Data')  https://in.mathworks.com/help/examples/signal/win64/FilteringDataWithSignalProcessingToolboxSoftwareExample_03.png Lowpass FIR Filter with Filter Designer This example shows how to design and implement a lowpass FIR filter using the window method with the interactive **Filter Designer** app.   * Start the app by entering filterDesigner at the command line. * Set the **Response Type** to **Lowpass**. * Set the **Design Method** to **FIR** and select the **Window** method. * Under **Filter Order**, select **Specify order**. Set the order to 20. * Under **Frequency Specifications**, set **Units** to **Hz**, **Fs** to 1000, and **Fc** to 150. * Click **Design Filter**. * Select **File** > **Export...** to export your FIR filter to the MATLAB® workspace as coefficients or a filter object. In this example, export the filter as an object. Specify the variable name as Hd. * Click **Export**. * Filter the input signal in the command window with the exported filter object. Plot the result for the first ten periods of the 100 Hz sinusoid.   y2 = filter(Hd,x);  plot(t,x,t,y2)  xlim([0 0.1])  xlabel('Time (s)')  ylabel('Amplitude')  legend('Original Signal','Filtered Data')   * Select **File** > **Generate MATLAB Code** > **Filter Design Function** to generate a MATLAB function to create a filter object using your specifications.   You can also use the interactive tool filterBuilder to design your filter. Bandpass Filters – Minimum-Order FIR and IIR Systems This example shows how to design a bandpass filter and filter data with minimum-order FIR equiripple and IIR Butterworth filters. You can model many real-world signals as a superposition of oscillating components, a low-frequency trend, and additive noise. For example, economic data often contain oscillations, which represent cycles superimposed on a slowly varying upward or downward trend. In addition, there is an additive noise component, which is a combination of measurement error and the inherent random fluctuations in the process.  In these examples, assume you sample some process every day for one year. Assume the process has oscillations on approximately one-week and one-month scales. In addition, there is a low-frequency upward trend in the data and additivewhite Gaussian noise.  Create the signal as a superposition of two sine waves with frequencies of 1/7 and 1/30 cycles/day. Add a low-frequency increasing trend term andwhite Gaussian noise. Reset the random number generator for reproducible results. The data is sampled at 1 sample/day. Plot the resulting signal and the power spectral density (PSD) estimate.  rng default  Fs = 1;  n = 1:365;  x = cos(2\*pi\*(1/7)\*n)+cos(2\*pi\*(1/30)\*n-pi/4);  trend = 3\*sin(2\*pi\*(1/1480)\*n);  y = x+trend+0.5\*randn(size(n));  [pxx,f] = periodogram(y,[],[],Fs);  subplot(2,1,1)  plot(n,y)  xlim([1 365])  xlabel('Days')  grid  subplot(2,1,2)  plot(f,10\*log10(pxx))  xlabel('Cycles/day')  ylabel('dB')  grid  https://in.mathworks.com/help/examples/signal/win64/FilteringDataWithSignalProcessingToolboxSoftwareExample_07.png |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Date:** | **28-May-2020** | **Name:** | **Raziya Banu** | |
| **Course:** | **Udemy** | **USN:** | **4AL16EC058** | |
| **Topic:** | **Dictionary in Python** | **Semester & Section:** | **8th sem & ‘B’ section** | |
| **AFTERNOON SESSION DETAILS** | | | |
| **Image of session** | | | |
| **Python Dictionaries:** A dictionary is a collection which is unordered, changeable and indexed. In Python dictionaries are written with curly brackets, and they have keys and values.  **Example**  Create and print a dictionary:  thisdict = {   "brand": "Ford",   "model": "Mustang",   "year": 1964 } print(thisdict) Accessing Items You can access the items of a dictionary by referring to its key name, inside square brackets:  **Example**  Get the value of the "model" key:  x = thisdict["model"]  There is also a method called get() that will give you the same result:  **Example**  Get the value of the "model" key:  x = thisdict.get("model Change Values You can change the value of a specific item by referring to its key name:  **Example**  Change the "year" to 2018:  thisdict = {   "brand": "Ford",   "model": "Mustang",   "year": 1964 } thisdict["year"] = 2018 Loop Through a Dictionary You can loop through a dictionary by using a for loop.  When looping through a dictionary, the return value are the keys of the dictionary, but there are methods to return the values as well.  **Example**  Print all key names in the dictionary, one by one:  for x in thisdict:   print(x)  Example  Print all values in the dictionary, one by one:  for x in thisdict:   print(thisdict[x])  Example  You can also use the values() method to return values of a dictionary:  for x in thisdict.values():   print(x)  Example  Loop through both keys and values, by using the items() method:  for x, y in thisdict.items():   print(x, y) Check if Key Exists To determine if a specified key is present in a dictionary use the in keyword:  **Example**  Check if "model" is present in the dictionary:  thisdict = {   "brand": "Ford",   "model": "Mustang",   "year": 1964 } if "model" in thisdict:   print("Yes, 'model' is one of the keys in the thisdict dictionary") Dictionary Length To determine how many items (key-value pairs) a dictionary has, use the len() function.  **Example**  Print the number of items in the dictionary:  print(len(thisdict)) Adding Items Adding an item to the dictionary is done by using a new index key and assigning a value to it:  **Example**  thisdict = {   "brand": "Ford",   "model": "Mustang",   "year": 1964 } thisdict["color"] = "red" print(thisdict) Removing Items There are several methods to remove items from a dictionary:  **Example**  The pop() method removes the item with the specified key name:  thisdict = {   "brand": "Ford",   "model": "Mustang",   "year": 1964 } thisdict.pop("model") print(thisdict)  **Example**  The popitem() method removes the last inserted item (in versions before 3.7, a random item is removed instead):  thisdict = {   "brand": "Ford",   "model": "Mustang",   "year": 1964 } thisdict.popitem() print(thisdict)  **Example**  The del keyword removes the item with the specified key name:  thisdict = {   "brand": "Ford",   "model": "Mustang",   "year": 1964 } del thisdict["model"] print(thisdict)  **Example**  The del keyword can also delete the dictionary completely:  thisdict = {   "brand": "Ford",   "model": "Mustang",   "year": 1964 } del thisdict print(thisdict) #this will cause an error because "thisdict" no longer exists. | | | |