**DAILY ASSESSMENT FORMAT**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date:** | **27may 2020** | **Name:** | **veronica gudagur** |
| **Course:** | **Digital signal processing** | **USN:** | **4al16ec091** |
| **Topic:** | **Fourier Transforms**  **FFT Fast Fourier Transform Mat lab**  **Implementation of signal Filtering signal using WT in Mat Lab** | **Semester & Section:** | **8-B** |
| **Github Repository:** | **veronica-g** |  |  |

|  |
| --- |
| **FORENOON SESSION DETAILS** |
| **Image of session** |
| **Report**  The DFT is tremendously useful for numerical approximation and computation, but it does not scale well to very large n \_ 1, as the simple formulation involves multiplication by a dense n \_ n matrix, requiring O(n2) operations.    In 1965, James W. Cooley (IBM) and John W. Tukey (Princeton) developed the revolutionary fast Fourier transform (FFT) algorithm [137, 136] that scales as O(n log(n)). As n becomes very large, the log(n) component grows slowly, and the algorithm approaches a linear scaling. Their algorithm was based on a fractal symmetry in the Fourier transform that allows an n dimensional DFT to be solved with a number of smaller dimensional DFT computations.  **Discrete Fourier transform:**  clear all, close all, clc  n = 256;  w = exp(-i\*2\*pi/n);  % Slow  for i=1:n  for j=1:n  DFT(i,j) = wˆ((i-1)\*(j-1));  end  end  % Fast  [I,J] = meshgrid(1:n,1:n);  DFT = w.ˆ((I-1).\*(J-1));  imagesc(real(DFT))  **Fast Fourier transform:**  As mentioned earlier, multiplying by the DFT matrix F involves O(n2) operations.  The fast Fourier transform scales as O(n log(n)), enabling a tremendous range of applications, including audio and image compression in MP3 and JPG formats, streaming video, satellite communications, and the cellular network, to name only a few of the myriad applications.    For example, audio is generallysampled at 44:1 kHz, or 44; 100 samples per second. For 10 seconds of audio,the vector f will have dimension n = 4:41 \_ 105. Computing the DFT usingmatrix multiplication involves approximately 2 \_ 1011, or 200 billion, multiplications.  In contrast, the FFT requires approximately 6 \_ 106, which amounts to a speed-up factor of over 30; 000. Thus, the FFT has become synonymous with the DFT, and FFT libraries are built in to nearly every device and operating system that performs digital signal processing.  >>fhat = **fft**(f); % Fast Fourier transform  >>f = **ifft**(fhat); % Inverse fast Fourier transform  Fast Fourier transform to compute derivatives.  n = 128;  L = 30;  dx = L/(n);  x = -L/2:dx:L/2-dx;  f = **cos**(x).\***exp**(-x.ˆ2/25); % Function  df = -(**sin**(x).\***exp**(-x.ˆ2/25) + (2/25)\*x.\*f); % Derivative  %% Approximate derivative using finite Difference...  **for** kappa=1:**length**(df)-1  dfFD(kappa) = (f(kappa+1)-f(kappa))/dx;  **end**  dfFD(**end**+1) = dfFD(**end**);  %% Derivative using FFT (spectral derivative)  fhat = **fft**(f);  kappa = (2\***pi**/L)\*[-n/2:n/2-1];  kappa = **fftshift**(kappa); % Re-order fft frequencies  dfhat = i\*kappa.\*fhat;  dfFFT = **real**(**ifft**(dfhat));  %% Plotting commands  **plot**(x,df,’k’,’LineWidth’,1.5), **hold** on  **plot**(x,dfFD,’b--’,’LineWidth’,1.2)  **plot**(x,dfFFT,’r--’,’LineWidth’,1.2)  **legend**(’True Derivative’,’FiniteDiff.’,’FFT Derivative’) **Infinite impulse response (IIR) filters:** IIR filters are the most efficient type of filter to implement in DSP (digital signal processing). They are usually provided as "biquad" filters. For example, in the parametric EQ block of a miniDSP plugin, each peak/notch or shelving filter is a single biquad. In the crossover blocks, each crossover uses up to 4 biquads. Each band of a graphic EQ is a single biquad, so a full 31-band graphic EQ uses 31 biquads per channel. **Finite impulse response (FIR) filters** An FIR filter requires more computation time on the DSP and more memory. The DSP chip therefore needs to be more powerful. miniDSP products that support FIR filtering include the [OpenDRC](https://www.minidsp.com/products/opendrc-series) and the [miniSHARC kit](https://www.minidsp.com/products/minidspkits/minisharc-kit).  FIR filters are specified using a large array of numbers. In the case of the OpenDRC, there are 6144 coefficients (or "taps") per channel. In the case of the miniSHARC, there are a total of 10240 taps assignable to all input and output channels. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  | |
|  |  |  |  | |
|  |  |  |  | |
|  | | | |
|  | | | |
|  | | | |