**DAILY ASSESSMENT FORMAT**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date:** | 26-5-2020 | **Name:** | Rasika Patil |
| **Course:** | DSP | **USN:** | 4AL16EC057 |
| **Topic:** | 1. Inner Product in Hilbert Transform 2. Complex Fourier Series 3. Fourier Series using Matlab (Use Octave to execute the code) 4. Fourier Series using Python (Experience implementation using Python) | **Semester & Section:** | 8th B |
| **Github Repository:** | Rasika B Patil |  |  |

|  |
| --- |
| **FORENOON SESSION DETAILS** |
| **Image of session**    **C:\Users\Admin\Desktop\26may.jfif**    **C:\Users\User\Pictures\Screenshots\Screenshot (216).png** |
| **Report – Report can be typed or hand written for up to two pages.**  **GENERAL INNER PRODUCT**  The inner product is an algebraic operation that takes two vectors of equal length and computes a single number, a scalar. It introduces a geometric intuition for length and angles of vectors. The inner product is a generalization of the dot product which is the more familiar operation that’s specific to the field of real numbers only. Euclidean space which is limited to 2 and 3 dimensions uses the dot product. The inner product is a structure that generalizes to vector spaces of any dimension. The utility of the inner product is very significant when proving theorems and runing computations. An important aspect that can be derived is the notion of convergence. Building on convergence we can move to represenations of functions, specifally periodic functions which show up frequently. The Fourier series is an expansion of periodic functions specific to sines and cosines. By the end of this paper we will be able to use a Fourier series to represent a wave function. To build toward this result many theorems are stated and only a few have proofs while some proofs are trivial and left for the reader to save time and space.  In mathematics and in signal processing, the Hilbert transform is a specific linear operator that takes a function, u(t) of a real variable and produces another function of a real variable H(u)(t). This linear operator is given by convolution with the function 1/t.  H(u)(t)=1/  the improper integral being understood in the principal value sense. The Hilbert transform has a particularly simple representation in the frequency domain: it imparts a phase shift of -90° to every Fourier component of a function. For example, the Hilbert transform of cos (ωt), where ω > 0, is cos(ωt-)  The Hilbert transform is important in signal processing, where it derives the analytic representation of a real-valued signal u(t). Specifically, the Hilbert transform of u is its harmonic conjugate v, a function of the real variable t such that the complex-valued function u + iv admits an extension to the complex upper half-plane satisfying the Cauchy–Riemann equations. The Hilbert transform was first introduced by David Hilbert in this setting, to solve a special case of the Riemann–Hilbert problem for analytic functions.  **COMPLEX FOURIER SERIES**  The complex Fourier series is presented first with period 2π, then with general period. The connection with the real-valued Fourier series is explained and formulae are given for converting between the two types of representation. Examples are given of computing the complex Fourier series and converting between complex and real serieses.  A representation based on this family of functions is called the “complex Fourier series”. f(t) = X∞ n =−∞ c n eint The coefficients, c n, are normally complex numbers. It is often easier to calculate than the sin/cos Fourier series because integrals with exponentials in are usually easy to evaluate. We will now derive the complex Fourier series equations, as shown above, from the sin/cos Fourier series using the expressions for sin() and cos() in terms of complex exponentials.  Let the function f(x) be defined on the interval [−π,π]. Using the well-known Euler’s formulas  cosφ=eiφ+e−iφ2,sinφ=eiφ−e−iφ2i,  we can write the Fourier series of the function in complex form:  f(x)=a02+∞∑n  =1(ancosnx+bnsinnx)  =a02+∞∑n  =1(aneinx+e−inx2+bneinx−e−inx2i)  =a02+∞∑n=1an−ibn2einx+∞∑n=1an+ibn2e−inx=∞∑n=−∞cneinx.  Here we have used the following notations:  c0=a02,cn=an−ibn2,c−n=an+ibn2.  **FOURIER SERIES USING MATLAB**  Fourier series provides an alternate way of representing data: instead of representing the signal amplitude as a function of time, we represent the signal by how much information is contained at different frequencies. If you ever watched the blinking lights on a stereo equalizer then you have seen Fourier analysis at work. The lights represent whether the music contains lots of bass or treble. Jean Baptiste Joseph Fourier, a French Mathematician who once served as a scientific adviser to Napoleon, is credited with the discovery of the results that now bear his name. Fourier analysis is important in data acquisition just as it is in stereos. Just as you might want to boost the power of the bass on your stereo you might want to filter out high frequency noise from the nearby radio towers in Needham when you are conducting a lab experiment. Fourier analysis allows you to isolate certain frequency ranges. This document will describe some of the basics of Fourier series and will show you how you can easily perform this analysis using MATLAB. While MATLAB makes it easy to translate a signal from the time domain to the frequency domain, one must understand how to interpret the data in the frequency domain. About Fourier Series Models The Fourier series is a sum of sine and cosine functions that describes a periodic signal. It is represented in either the trigonometric form or the exponential form. The toolbox provides this trigonometric Fourier series form  *y*=*a*0+*n*Ξ*i*=1*ai*cos(*iwx*)+*bi*sin(*iwx*)  where a0 models a constant (intercept) term in the data and is associated with the i = 0 cosine term, w is the fundamental frequency of the signal, n is the number of terms (harmonics) in the series, and 1 ≤ n ≤ 8.  For more information about the Fourier series, refer to [Fourier Analysis and Filtering](https://in.mathworks.com/help/matlab/fourier-analysis-and-filtering.html) (MATLAB). Fit Fourier Models Interactively  1. Open the Curve Fitting app by entering cftool. Alternatively, click Curve Fitting on the Apps tab. 2. In the Curve Fitting app, select curve data (**X data** and **Y data**, or just **Y data** against index).   Curve Fitting app creates the default curve fit, Polynomial.   1. Change the model type from Polynomial to Fourier.   C:\Users\User\Pictures\Screenshots\Screenshot (218).png  **FOURIER SERIES USING PYTHON**  Fourier Transform is used to analyze the frequency characteristics of various filters. For images, **2D Discrete Fourier Transform (DFT)** is used to find the frequency domain. A fast algorithm called **Fast Fourier Transform (FFT)** is used for calculation of DFT. Details about these can be found in any image processing or signal processing textbooks. Please see [Additional Resources](https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_transforms/py_fourier_transform/py_fourier_transform.html#additional-resources) section.  For a sinusoidal signal, x(t) = A \sin(2 \pi ft), we can say f is the frequency of signal, and if its frequency domain is taken, we can see a spike at f. If signal is sampled to form a discrete signal, we get the same frequency domain, but is periodic in the range [- \pi, \pi] or [0,2\pi] (or [0,N] for N-point DFT). You can consider an image as a signal which is sampled in two directions. So taking fourier transform in both X and Y directions gives you the frequency representation of image.  More intuitively, for the sinusoidal signal, if the amplitude varies so fast in short time, you can say it is a high frequency signal. If it varies slowly, it is a low frequency signal. You can extend the same idea to images. Where does the amplitude varies drastically in images ? At the edge points, or noises. So we can say, edges and noises are high frequency contents in an image. If there is no much changes in amplitude, it is a low frequency component. |

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