**DAILY ASSESSMENT FORMAT**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date:** | **26/05/2020** | **Name:** | **Mamatha.m** |
| **Course:** | **Digital signal processing** | **USN:** | **4AL16EC035** |
| **Topic:** | **Find the Z-Transform of sequence using Matlab**  **Transform and Laplace Transform** | **Semester & Section:** | **6 B** |
| **Github Repository:** | **Mamatha\_m** |  |  |

|  |
| --- |
| **FORENOON SESSION DETAILS** |
| **Image of session** |
| **Report**  **Convolution.** The convolution of two functions is particularly well-behaved  in the Fourier domain, being the product of the two Fourier transformed functions.  Define the convolution of two functions f(x) and g(x) as f \_ g:    Thus, multiplying functions in the frequency domain is the same as convolving  functions in the spatial domain. This will be particularly useful for control  systems and transfer functions with the related Laplace transform.  Calculating the Laplace F(s) transform of a function f(t) is quite simple in Matlab. First you need to specify that the variable t and s are symbolic ones. This is done with the command  >>syms t s  Next you define the function f(t). The actual command to calculate the transform is  >> F=laplace(f,t,s)  To make the expression more readable one can use the commands, simplify and pretty.  >>syms t s  >> f=-1.25+3.5\*t\*exp(-2\*t)+1.25\*exp(-2\*t);  >> F=laplace(f,t,s) F = -5/4/s+7/2/(s+2)^2+5/4/(s+2)  >> simplify(F) ans = (s-5)/s/(s+2)^2  >> pretty(ans)  The **Z Transform** has a strong relationship to the DTFT, and is incredibly useful in transforming, analyzing, and manipulating discrete calculus equations. The Z transform is named such because the letter 'z' (a lower-case Z) is used as the transformation variable.  clc;  close all;  clear all;  syms 'z';  disp('If you input a finite duration sequence x(n), we will give you its z-transform');  nf=input('Please input the initial value of n = ');  nl=input('Please input the final value of n = ');  x= input('Please input the sequence x(n)= ');  syms 'm';  syms 'y';  f(y,m)=(y\*(z^(-m)));  disp('Z-transform of the input sequence is displayed below');  k=1;  for n=nf:1:nl      answer(k)=(f((x(k)),n));     k=k+1;  end  disp(sum(answer)); |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Date:** | **26/5/2020** | **Name:** | **Mamatha.m** | |
| **Course:** | **Python** | **USN:** | **4al16ec035** | |
| **Topic:** | **Application 4: Build a**  **Personal Website with**  **Python and Flask** | **Semester & Section:** | **6 B** | |
| **AFTERNOON SESSION DETAILS** | | | |
| CSS separates the content contained in HTML files from how the content should be displayed. It is important to separate the content from the rules for how it should be rendered primarily because it is easier to reuse those rules across many pages. CSS files are also much easier to maintain on large projects than styles embedded within the HTML files. CSS preprocessors A CSS preprocessor compiles a processed language into plain CSS code. CSS preprocessing languages add syntax such as variables, mixins and functions to reduce code duplication. The additional syntax also makes it possible for designers to use these basic programming constructs to write maintainable front end code.  View source screenshot for the fsp.css file in index.html.  **Navigation menu:**  /\* Add a black background color to the top navigation \*/ .topnav {   background-color: #333;   overflow: hidden; }  /\* Style the links inside the navigation bar \*/ .topnav a {   float: left;   color: #f2f2f2;   text-align: center;   padding: 14px 16px;   text-decoration: none;   font-size: 17px; }  /\* Change the color of links on hover \*/ .topnav a:hover {   background-color: #ddd;   color: black; }  /\* Add a color to the active/current link \*/ .topnava.active {   background-color: #4CAF50;   color: white; } | | | |
|  | | | |