

DAILY ASSESSMENT REPORT

Date:	26/05/2020	Name:	Abhishek M Shastry K
Subject:	Digital Signal Processing	USN:	4AL17EC002
Topic:	1] Fourier Series & Gibbs Phenomena using Python 2] Fourier Transform 3] Fourier Transform Derivatives 4] Fourier Transform and Convolution 5] Intuition of Fourier Transform and Laplace Transform 6] Laplace Transform of First order 7] Implementation of Laplace Transform using Matlab 8] Applications of Z-Transform 9] Find the Z-Transform of sequence using Matlab	Semester & Section:	6 th 'A'
Github Repository:	AbhishekShastry-Courses		

FORENOON SESSION DETAILS

Image of session

The screenshot shows a YouTube video player with the title "Fourier Series and Gibbs Phenomena [Python]". The video content features a presenter on the left and a Jupyter Notebook on the right. The Jupyter Notebook displays a plot of a square wave and its Fourier series approximation, illustrating the Gibbs phenomenon. The video has 2,555 views and is dated Mar 15, 2020. The presenter is Steve Brunton.

Fourier Series and Gibbs Phenomena [Python]

2,555 views · Mar 15, 2020

Up next: Denoising Data with FFT [Python] by Steve Brunton

Laplace Transform and Inverse Laplace Transform Using MATLAB | Mad Over Matlab Tutorials

23,463 views • Feb 22, 2017

86 13 SHARE SAVE

Up next

AUTOPLAY

How to find laplace & inverse laplace by matlab

Hakar Muhammad

The screenshot shows a MATLAB R2016a environment. The Editor window displays a script named 'laplace.m' with the following code:

```
1 = clear
2 = clear all;
3
4 = syms t f t;
5 = (exp(-3*t)*sin(2*t))/t;
6 = laplace(f);
```

The Command Window shows the execution of the script, resulting in the Laplace transform:

```
f =
(sin(2*t)*exp(-3*t))/t
L =
atan(2/(s + 3))
f >>
```

How to calculate Z-Transform in Matlab ??

19,673 views • Jul 29, 2015

69 10 SHARE SAVE

Helping you connect

The screenshot shows a MATLAB R2016a environment. The Editor window displays a script named 'z_transform.m' with the following code:

```
8 = syms n N0?
9 = % signal
10 = a = n + 1;
11 = disp('The input equation is');
12 = disp(a);
13 = % taking z-Transform
14 = b = ztrans(a);
```

The Command Window shows the execution of the script, resulting in the Z-Transform:

```
>> syms n;
>> a = n+1;
>> b = ztrans(a);
>> disp(b)
z/(z - 1) + z/(z - 1)^2
>> pretty(b)
z      z
----- + -----
z - 1  (z - 1)^2
```

Subscribe for more Videos

Report

Derivatives of functions

The Fourier transform of the derivative of a function is given by:

$$\begin{aligned}\mathcal{F}\left(\frac{d}{dx}f(x)\right) &= \int_{-\infty}^{\infty} \overbrace{f'(x)}^{dv} \overbrace{e^{-i\omega x}}^u dx \\ &= \left[\underbrace{f(x)e^{-i\omega x}}_{uv} \right]_{-\infty}^{\infty} - \int_{-\infty}^{\infty} \underbrace{f(x)}_v \left[\underbrace{-i\omega e^{-i\omega x}}_{du} \right] dx \\ &= i\omega \int_{-\infty}^{\infty} f(x)e^{-i\omega x} dx \\ &= i\omega \mathcal{F}(f(x)).\end{aligned}$$

- This is an extremely important property of the Fourier transform, as it will allow us to turn PDEs into ODEs, closely related to the separation of variables:

$$\begin{array}{ccc} u_{tt} = cu_{xx} & \xrightarrow{\mathcal{F}} & \hat{u}_{tt} = -c\omega^2 \hat{u}. \\ \text{(PDE)} & & \text{(ODE)} \end{array}$$

Linearity of Fourier transforms

The Fourier transform is a linear operator, so that:

$$\mathcal{F}(\alpha f(x) + \beta g(x)) = \alpha \mathcal{F}(f) + \beta \mathcal{F}(g).$$

$$\mathcal{F}^{-1}(\alpha \hat{f}(\omega) + \beta \hat{g}(\omega)) = \alpha \mathcal{F}^{-1}(\hat{f}) + \beta \mathcal{F}^{-1}(\hat{g}).$$

Parseval's theorem

$$\int_{-\infty}^{\infty} |\hat{f}(\omega)|^2 d\omega = 2\pi \int_{-\infty}^{\infty} |f(x)|^2 dx.$$

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 $f * g$:

$$(f * g)(x) = \int_{-\infty}^{\infty} f(x - \xi)g(\xi) d\xi.$$

If we let $\hat{f} = \mathcal{F}(f)$ and $\hat{g} = \mathcal{F}(g)$, then:

$$\begin{aligned}\mathcal{F}^{-1}(\hat{f}\hat{g})(x) &= \frac{1}{2\pi} \int_{-\infty}^{\infty} \hat{f}(\omega)\hat{g}(\omega)e^{i\omega x} d\omega \\ &= \int_{-\infty}^{\infty} \hat{f}(\omega)e^{i\omega x} \left(\frac{1}{2\pi} \int_{-\infty}^{\infty} g(y)e^{-i\omega y} dy \right) d\omega \\ &= \frac{1}{2\pi} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} g(y)\hat{f}(\omega)e^{i\omega(x-y)} d\omega dy \\ &= \int_{-\infty}^{\infty} g(y) \underbrace{\left(\frac{1}{2\pi} \int_{-\infty}^{\infty} \hat{f}(\omega)e^{i\omega(x-y)} d\omega \right)}_{f(x-y)} dy \\ &= \int_{-\infty}^{\infty} g(y)f(x-y) dy = g * f = f * g.\end{aligned}$$

Laplace Transform

Laplace transform of a signal (function) f is the function $F = \mathcal{L}(f)$ defined by:

$$F(s) = \int_0^{\infty} f(t)e^{-st} dt$$

For those $s \in \mathbb{C}$ for which the integral makes sense.

- F is a complex-valued function of complex numbers.
- s is called the (complex) frequency variable, with units sec^{-1} .
- t is called the time variable (in sec).
- Assume f contains no impulses at $t = 0$.

Date:	26/05/2020	Name:	Abhishek M Shastry K
Course:	The Python Mega Course: Build 10 Real World Applications	USN:	4AL17EC002
Topic:	1] Graphical User Interfaces with Tkinter	Semester & Section:	6 th 'A'
Github Repository:	AbhishekShastry-Courses		

AFTERNOON SESSION DETAILS

Image of session

The screenshot shows a Udemy course page for 'The Python Mega Course: Build 10 Real World Applications'. The specific lesson is 'Create a Multi-widget GUI (Practice)'. The instructions state: 'Create a Python program that expects a kilogram input value and converts that value to grams, pounds, and ounces when the user pushes the *Convert* button. The program will look similar to the one in the following picture:'. A small image of the GUI is shown, featuring a text input field with 'kg', a 'Convert' button, and three output fields showing '25000.0', '55.1155', and '881.85'. The right sidebar shows the course content, with Section 21: Graphical User Interfaces with Tkinter (5/5 | 22min) being the current section. The bottom of the image shows a Windows taskbar with various application icons.

The screenshot shows a Visual Studio Code editor with the Python code for the GUI application. The code is as follows:

```

1 from tkinter import *
2
3 # Create an empty Tkinter window
4 window=Tk()
5
6 def from_kg():
7     # Get user value from input box and multiply by 1000 to get kilograms
8     gram=float(e2_value.get())*1000
9
10    # Get user value from input box and multiply by 2.20462 to
11    pound=float(e2_value.get())*2.20462
12
13    # Get user value from input box and multiply by 35.274 to
14    ounce=float(e2_value.get())*35.274
15
16    # Empty the Text boxes if they had text from the previous
17    t1.delete("1.0", END) # Deletes the content of the Text box from start to END
18    t1.insert(END,gram) # Fill in the text box with the value of gram variable
19    t2.delete("1.0", END)
20    t2.insert(END,pound)
21    t3.delete("1.0", END)
22    t3.insert(END,ounce)
23
24 # Create a Label widget with "Kg" as label
25 e1=Label(window,text="Kg")

```

The output of the program is visible in the terminal window, showing the conversion results for 25 kg: 25000.0 grams, 55.1155 pounds, and 881.85 ounces.

Report

Graphical User Interfaces with Tkinter

- The graphical user interface is a form of user interface that allows users to interact with electronic devices through graphical icons and audio indicator such as primary notation, instead of text-based user interfaces, typed command labels or text navigation.
- **Tkinter** is the standard GUI library for Python. Python when combined with Tkinter provides a fast and easy way to create GUI applications. Tkinter provides a powerful object-oriented interface to the Tk GUI toolkit.
- The two main elements of GUI are **Window** and **Widgets**.
- A window is an area on the screen that displays information, with its contents being displayed independently from the rest of the screen.
- Interface elements known as **graphical control elements, controls** or **widgets** are software components that a computer user interacts with through direct manipulation to read or edit information about an application. Each widget facilitates a specific user-computer interaction.
- The **Tk ()** function is used to create a GUI window.
- The **mainloop ()** function is used to keep GUI window open until user closes window mandatorily.
- To create widgets:
 - ✓ The **Button ()** function is used to implement various kinds of buttons. Buttons can contain text or images, and you can associate a Python function or method with each button. When the button is pressed, Tkinter automatically calls that function or method.
 - ✓ The **Entry ()** function is used to accept single-line text strings from a user.
 - ✓ The **text ()** function is used to display text documents, containing either plain text or formatted text (using different fonts, embedded images, and other embellishments). The text widget can also be used as a text editor.