International Journal of Mathematical Archive-5(8), 2014, 30-35
Available online through www.ijma.info ISSN 2229 – 5046

SIMULATION OF LAPLACE TRANSFORMS WITH PYTHON

Gajanan Dhanorkar* and Deepak Sonawane

VPCOE, Vidyanagari MIDC, Baramati, Pune, (M.S.)-413133, India.

(Received On: 21-07-14; Revised & Accepted On: 14-08-14)

ABSTRACT

In this paper we have given applications of Laplace Transform to analyses signals in time domain to frequency domain using python, solving differential equations with initial conditions and computing the results in graphical format. This technique found useful and create the interest among the students at large.

1. INTRODUCTION

Python is very flexible and open source nature, it is widely accepted language in scientific computing community from several years. Different software tools are available for computing mathematical equations such as MATLAB [1], Mathematica, Mapal, C & C++, etc. The major drawback of MATLAB is that it is not free and not open source which make difficult for researcher to share the coding. In the teaching learning paradigm of mathematics it becomes absolutely essential to display the graphical format of the equations and to understand its physical interpretation. In this context we have attempted to use the power and flexibility of open source, general purpose programming language, Python[2]. Python is used with its extension modules NumPy [3], SciPy [4] and Matplotlib [5], and now Sage [6], which adds more power to this scientific computing language to display the graphical results.

2. LAPLACE TRANSFORM

Laplace transform is an integral transform and is a powerful technique to solve differential equations. Particularly useful in solving linear ordinary differential equations (ODE). It finds very wide applications in various areas of physics, electrical engineering, control engineering, optics, mathematics and signal processing. The Laplace transform can be interpreted as a transformation from the time domain where inputs and outputs are functions of time to the frequency domain where inputs and outputs are functions of complex angular frequency.
The sufficient condition for any function of time f(t) to be Laplace transformable, it
must satisfy the following Dirichlet conditions [7]:

1. f(t) must be piecewise continuous which means that it must be single valued but can have a finite number of finite isolated discontinuities for t > 0.
2. f(t) must be exponential order which means that f(t) must remain less than Me^{ct} as t approaches to ∞, where S is a positive constant and c is a real positive number.

Definition 2.1: A Laplace Transform of f(t) is F(s) defined by

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

and its inverse is

$$L^{-1}\{F(s)\} = f(t)$$

We are using python for finding Lapalace Transform.

3. PRELIMINARIES

3.1. Why Python?

i) Very rich scientific computing libraries (a bit less than Matlab, though).
ii) Well thought out language, allowing to write very readable and well structured code: we code what we think.
iii) Many libraries for other tasks than scientific computing (web server management, serial port access, etc.)
iv) Free and open-source software, widely spread, with a vibrant community.

*Corresponding Author: Gajanan Dhanorkar**

3.2. Algebra: using SymPy

First we have to import Sympy as "from sympy import *" and define symbol x as "x = Symbol ('x')".

3.2.1. To find partial fraction decomposition of $\frac{1}{(x^2+2x+1)(x^2-1)}$ **is**

```
>>>from sympy import *
>>> x = Symbol('x')
>>> apart(1/( (x**2+2*x+1)*(x**2-1) ), x) (perform partial fraction decomposition.)
```

Output: -1/(8*(x + 1)) - 1/(4*(x + 1)**2) - 1/(2*(x + 1)**3) + 1/(8*(x - 1))

3. 2.2. To combine fraction of $-\frac{1}{8(x+1)}-\frac{1}{4(x+1)^2}-\frac{1}{2(x+1)^3}+\frac{1}{8(x-1)}$ **is**

```
>>> together(-1/(8*(x + 1)) - 1/(4*(x + 1)**2) - 1/(2*(x + 1)**3) + 1/(8*(x - 1))) (To combine expressions)
```

Output: (-4*x - (x - 1)*(x + 1)**2 - 2*(x - 1)*(x + 1) + (x + 1)**3 + 4)/(8*(x - 1)*(x + 1)**3)

Expand and separate

3. 2.3. Find expansion of $(a-b)^3$

```
>>> a, b = symbols('a b')
>>> ((a-b)**3).expand()
```

Output: a**3 - 3*a**2*b + 3*a*b**2 - b**3

3. 2.4. Find expansion of $(a+b)(c+d)$

```
>>> separate((a+b)*(c+d)).expand()
```

Output: ac + ad + bc + bd

3.3. Calculus: using SymPy

3. 3.1. To find limit of the function:

$\lim_{x\to 0}\frac{\sin x}{x}$

```
>>> limit(sin(x)/x, x, 0)
```

Output: 1

3. 3.2. To find limit of $\lim_{x\to\infty}\left(\frac{1}{x}\right)^{\frac{1}{x}}$

```
>>> limit((1/x)**(1/x), x, oo)
```

Output: 1

3.4 Derivative of Functions:

3. 4.1. To find derivative of $\cos^3 x$

```
>>> diff(cos(x**3), x)
```

Output: -3*x**2*sin(x**3)

3. 4.2. To find derivative of $\sin^3 x$

```
>>> diff(sin(x**3), x)
```

Output: 3*x**2*cos(x**3)

3.5. Series representation of the function:

3. 5.1. To find series expansion of $\cos x$ upto x^{14}
>>> cos(x).series(x, 0, 14) (series representation up to x 14.)

Output: 1 - x**2/2 + x**4/24 - x**6/720 + x**8/40320 - x**10/3628800 + x**12/479001600+ O(x**14)

3. 5.2. To find series expansion of $\sin x$ upto x^{14}
>>> sin(x).series(x, 0, 14) (series representation up to x 14.)

Output: 1 - x**2/2 + x**4/24 - x**6/720 + x**8/40320 - x**10/3628800 + x**12/479001600+ O(x**14)

3.6. Integration:

3. 6.1. To find $\int (x^2 + 2x + 4)dx$, we can write in python as
>>> integrate(x**2 + 2*x + 4, x)

Output: x**3/3 + x**2 + 4*x

3. 6.2. To find $\int 5x^2 e^x \sin x dx$, we can write in python as
>>> integrate(5*x**2 * exp(x) * sin(x), x)

Output: 5*x**2*exp(x)*sin(x)/2 - 5*x**2*exp(x)*cos(x)/2 + 5*x*exp(x)*cos(x) - 5*exp(x)*sin(x)/2
-5*exp(x)*cos(x)/2

3. 6.3. To find $\int_0^{\pi} \cos^2 x e^x dx$, we can write in python as
>>> integrate (cos(x)**2*exp(x), (x, 0, pi))

Output: -3/5 + 3*exp (pi)/5

3. 6.4. To find $\int_0^{\infty} \cos^2 x e^{-x} dx$, we can write in python as
>>> integrate(cos(x)**2*exp(-x), (x, 0, oo))

Output: 3/5

3.7. Differential Equation:

3.7.1. To solve Differential equation $\frac{d^2 f(x)}{dx^2} + 9 f(x) = 0$
First we import,
>>> from sympy import Function, dsolve, Eq, Derivative, sin, cos
>>> from sympy.abc import x
>>> f = Function(' f ')
>>> dsolve(Derivative(f(x),x,x)+9*f(x), f(x))

Output: f(x) == C1*cos(3*x) + C2*sin(3*x)

3.7.2. Solve $y''(t) - 2y'(t) + y(t) = e^{2t}$
>>> dsolve (Derivative(y(t),t,t)-2*Derivative(y(t),t)+ y(t)-exp(2*t), y(t))

Output: y(t) = (c1 + c2t)e**t + e**2t

Note: Also we can write above equation in following way
diffeq = Eq(y(t).diff(t,t)-2*y(t).diff(t)+ y(t), exp(2*t))

4. LAPLACE TRANSFORM OF STANDARD FUNCTIONS

4.1. To find L.T. of f(t) = 1

```
>>> from sympy.integrals import laplace_ transform
>>> from sympy.abc import t, s, a
>>> laplace_ transform(1, t, s)
```

Outpu: $\left(\frac{1}{s}, 0, True\right)$

4.2. To find L.T. of f(t) = t

```
>>> laplace_ transform(t, t, s)
```

Output: $\left(\frac{1}{s^2}, 0, True\right)$

4.3. To find L.T. of c

```
>>> laplace _transform(t**a, t, s)
```

Output: (s**(-a)*gamma(a+1) / s, 0, -re (a) <1)

4.4. To find L.T. of sint

```
>>> laplace _transform(sin(t), t, s)
```

Output: (1/s2+1; 0; True)

4.5. To find L.T. of $f(t) = t^2 + t\sin t$

```
>>> laplace _transform(t**2 + t*sin(t), t, s)
```

Output: $\left(\frac{2s}{\left(s^2+1\right)^2} + \frac{2s}{s^3}, 0, True\right)$

5. INVERSE LAPLACE TRANSFORM

5.1. Inverse L.T. of $F(s) = e^{-as}$

```
>>> from sympy.integrals.transforms import inverse_laplace_transform
>>> from sympy import exp, Symbol
>>> from sympy.abc import s, t
>>> a = Symbol('a', positive=True)
>>> inverse_laplace_transform(exp(-a*s)/s, s, t)
```

Output: Heaviside(-a + t)

5.2. Inverse L.T. of $F(s) = \frac{1}{s-1}$

```
>>> inverse_laplace_transform(1/(s-1) , s, t)
```

Output: e^t

5.3. **Inverse L.T. of** $F(s) = \frac{s}{s^2+1}$

```
>>>inverse_laplace_transform(s/(s**2+1) , s, t)
```

Output: cos(t)

5.4. Inverse L.T. of $F(s)=\frac{1}{s(s^2+1)}$

```
>>> inverse_laplace_transform(1/(s*(s**2+1)) , s, t)
```

Output: $-(\cos(t)-1)$

6. APPLICATIONS TO DIFFERENTIAL EQUATIONS

Slove Differential equation $y''(t)-2y'(t)+y(t)=e^{2t},\ y(t)=y'(t)=0$ (1)

```
>>> y(s)=laplace_transform(Derivative(y(t),t,t)-2*Derivative(y(t),t)+y(t)-exp(2*t), t, s)
```

Output: $\left(L_t[y(t)](s)-2L_t\left[\frac{d}{dt}y(t)\right](s)+L_t\left[\frac{d^2}{dt^2}y(t)\right](s)-\frac{1}{s-2},2,\frac{s}{2}\neq 1\right)$

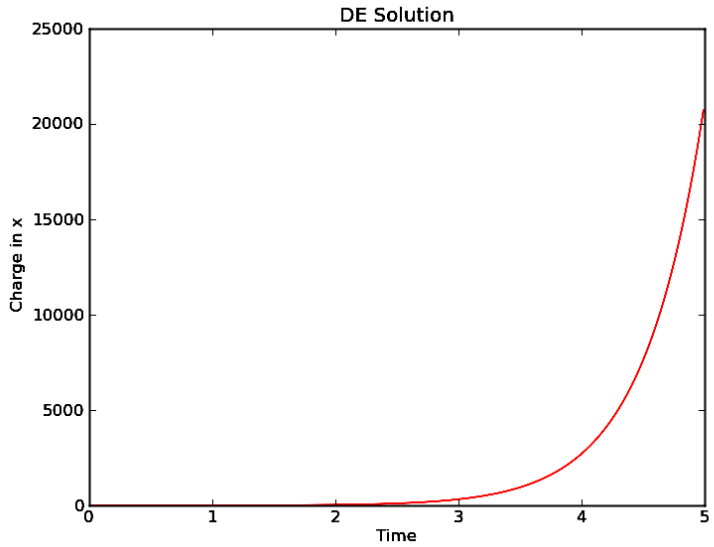
```
>>> f = 1/((s-2)*(s-1)**2)
>>> inverse_laplace_transform(f, s, t)
```

Output: $(e^{2t}-e^{t}-te^{t})$

7. GRAPH

Graph of solution of above differential equation (1) using python coding is

```
from pylab import *
t=arange(0,5,0.01)
q=exp(2*t)-exp(t)-t*exp(t) plot(t,q,'r')
xlabel('Time')
ylabel(' Charge in x')
title ('DE Solution')
show ()
```



REFERENCES

[1] The Math Works. 2010. MATLAB Release R2010b. The Math-Works, Natick, Massachusetts.

[2] Guido van Rossum and Fred L. Drake. 2006. Python Language Reference Manual. Network Theory Limited, Bristol, UK.

[3] Travis Oliphant. 2006. Guide To NumPy.Trelgol Publishing, USA.

[4] Eric Jones, Travis Oliphant, Pearu Peterson, et al. 2001. SciPy: Open source scientific tools for Python. http://www.scipy.org.

[5] John D. Hunter. 2007. Matplotlib: A 2D graphics environment. Computing In Science and Engineering, 9(3):90-95.

[6] Sage for Power Users by William Stein.

[7] A. D. Poularikas, The Transforms and Applications Hand- book (McGraw Hill, 2000), 2nd ed.

[8] Higher Engineering Mathematics, B.V.Ramana.

Source of support: Nil, Conflict of interest: None Declared