WIGGLES

EXPERIMENT #09

NOVEMBER 2022

EXPERIMENT NAME

Finding the laplace transform and inverse laplace transform of the given signals.

1. LAPLACE TRANSFORM

Theory

1. A function is said to be a **piecewise continuous** function if it has a finite number of breaks and it does not blow up to infinity anywhere. Let us assume that the **function f(t)** is a piecewise continuous function, **then f(t)** is defined using the Laplace transform. The Laplace transform of a function is represented by **L{f(t)} or F(s).**
2. **Laplace transform** is the **integral transform** of the given derivative function with real variable **t** to convert into a **complex function** with variable **s**. For **t ≥ 0**, let **f(t)** be given and assume the function satisfies certain conditions to be stated later on.
3. The Laplace transform **of f(t)**, that is denoted by **L{f(t)} or F(s)** is defined by the Laplace transform formula:

Expression

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

- Provided that the integral exists. Where the Laplace Operator, s = σ + jω; will be real or complex j = √(-1)

Getting the environment ready

- **Python 3.10** is installed in the system and added to the system variables.
- The library is installed through pip i.e. through the command **“pip install wiggles.”**
- Here, **vs code** is used to code and test out the results.
- The code is written to best find the solution of the given problem and then is evaluated and displayed using the inbuilt **‘show()’** or the **‘compare()’** function in wiggles.

PROBLEM

- Find the laplace transform of the following signals :

- Given,

1. $x(t) \; = \; 2 \times \delta(t) + e^{-3t}$

2. $x(t) \; = \; u(t-1) - 2e^{-1t}$

PROGRAM CODE 1

```
from wiggles import symbols as sy

#Given Expression
def x(t):
      return sy.unit_impulse(t)+sy.exp((-3)*t)

#Making time domain object
expression = sy.time_domain(x)
print("The Expression in time Domain: ",expression)

#Laplace transformation
y = expression.laplace_transform()
print("The Expression in frequency Domain is: ",y)
```

OUTPUT 1

```
The Expression in time Domain:  DiracDelta(t) + exp(-3*t)
The Expression in frequency Domain is:  (s + 4)/(s + 3)
```

PROGRAM CODE 2

```
from wiggles import symbols as sy

#Given Expression
def x(t):
      return sy.unit_step(t-1)-sy.exp((-2)*(-t))

#Making time domain object
expression = sy.time_domain(x)
print("The Expression in time Domain: ",expression)

#Laplace transformation
y = expression.laplace_transform()
print("The Expression in frequency Domain is: ",y)
```

OUTPUT 2

```
The Expression in time Domain:  -exp(2*t) + Heaviside(t - 1)
The Expression in frequency Domain is:  (-s*exp(s) + s - 2)*exp(-s)/(s*(s - 2))
```

2. INVERSE LAPLACE TRANSFORM

Theory

1. The **inverse Laplace transform** of a function **F(s)** is the piecewise-continuous and exponentially-restricted real function **f(t)** which has the property:

$$\mathcal{L}\{f\}(s) = \mathcal{L}\{f(t)\}(s) = F(s),$$

2. An **integral formula** for the **inverse Laplace transform,** called the **Mellin's inverse formula**, **the Bromwich integral**, or the **Fourier-Mellin integral**, is given by the line integral:.

Expression

$$f(t) = \frac{1}{2\pi i} \lim_{T \to \infty} \int_{\gamma - iT}^{\gamma + iT} F(s).e^{st}\, ds$$

- Provided that the integral exists. Where the Laplace Operator, $s = \sigma + j\omega$; will be real or complex $j = \sqrt{(-1)}$

- where the integration is done along the **vertical line Re(s) = γ** in the complex plane such that **γ** is greater than the real part of all singularities of F(s) and F(s) is bounded on the line, for example if the contour path is in the region of convergence.

Getting the environment ready

- **Python 3.10** is installed in the system and added to the system variables.
- The library is installed through pip i.e. through the command **"pip install wiggles."**
- Here, **vs code** is used to code and test out the results.
- The code is written to best find the solution of the given problem and then is evaluated and displayed using the inbuilt **'show()'** or the **'compare()'** function in wiggles.

PROBLEM A

- Find the laplace transform of the following signals :

- Given,

1. **X(s)** = $\frac{10s^2+4}{s(s+1)(s+2)^2}$
2. **X(s)** = $\frac{s^3+2s+6}{s(s+3)(s+1)^2}$

PROGRAM CODE 1

```
from wiggles import symbols as sy

#Given Expression
def x(s):
      return (10*(s**2)+4)/(s*(s+1)*(s+2)**2)

#Making frequency domain object
expression = sy.frequency_domain(x)
expression.name="X(s)"
print("The Expression in frequency Domain: ",expression)

#Inverse Laplace transformation
y = expression.inverse_laplace_transform()
y.name="x(t)"
print("The Expression in Time Domain is: ",y)
```

OUTPUT 1

```
The Expression in frequency Domain:  (10*s**2 + 4)/(s*(s + 1)*(s + 2)**2)
The Expression in Time Domain is:  (22*t + exp(2*t) - 14*exp(t) + 13)*exp(-2*t)*Heaviside(t)
```

PROGRAM CODE 2

```
from wiggles import symbols as sy

#Given Expression
def x(s):
      return ((s**3)+(2*s)+6)/(s*(s+3)*(s+1)**2)

#Making frequency domain object
expression = sy.frequency_domain(x)
expression.name="X(s)"
print("The Expression in frequency Domain: ",expression)

#Inverse Laplace transformation
y = expression.inverse_laplace_transform()
y.name="x(t)"
print("The Expression in Time Domain is: ",y)
```

OUTPUT 2

```
The Expression in frequency Domain:  (s**3 + 2*s + 6)/(s*(s + 1)**2*(s + 3))
The Expression in Time Domain is:  (-(6*t + 13)*exp(2*t) + 8*exp(3*t) + 9)*exp(-3*t)*Heaviside(t)/4
```

PROBLEM B

- Use commands to fragment the expression and find inverse laplace of the following :
- Given,

1. **X(s)** = $\frac{4s^{5}+20s^{4}+11s^{3}+10^{2}-12}{s^{4}+5s^{3}+8s^{2}+4s}$

PROGRAM CODE 1

```
from wiggles import symbols as sy

#Given Expression
```

```
def x(s):
      return
(4*(s**5)+20*(s**4)+11*(s**3)+10*(s**2)-12)/((s**4)+5*(s**3)+8*(s**2)+(4*s))

#Making frequency domain object
expression = sy.frequency_domain(x)
expression.name="X(s)"
print("The Expression in frequency Domain: \n",expression)

#Expanding and fragmenting the expression
expression.apart()
print("The Expanded and processed expression: \n",expression)

#Inverse Laplace transformation
y = expression.inverse_laplace_transform()
y.name="x(t)"
print("The Expression in Time Domain is: \n",y)
```

OUTPUT

```
The Expression in frequency Domain:
 (4*s**5 + 20*s**4 + 11*s**3 + 10*s**2 - 12)/(s**4 + 5*s**3 + 8*s**2 + 4*s)
The Expanded and processed expression:
 4*s - 15/(s + 2) + 66/(s + 2)**2 - 3/(s + 1) - 3/s
The Expression in Time Domain is:
 (3*(22*t - 5)*Heaviside(t) + (4*InverseLaplaceTransform(s, s, t, _None) - 3*Heaviside(t))*exp(2*t) - 3*exp(t)*Heaviside(t))*exp(-2*t)
```

3. POLES AND ZEROS

Theory

1. The zeros are the roots of the numerator.
2. The poles are the roots of the denominator.

Getting the environment ready

- **Python 3.10** is installed in the system and added to the system variables.
- The library is installed through pip i.e. through the command **"pip install wiggles."**
- Here, **vs code** is used to code and test out the results.
- The code is written to best find the solution of the given problem and then is evaluated and displayed using the inbuilt **'show()'** or the **'compare()'** function in wiggles.

PROBLEM

- Find the roots of the numerator and the denominator respectively to compute the poles and the zeros of the expression:

- Given,

1. **X(s) =** $\frac{s^2+3s+1}{s^3+4s^2+3s}$

PROGRAM CODE

```
from wiggles import symbols as sy

#given expression
def x(s):
      return ((s**2)+(3*s)+1)/((s**3)+(4*(s**2))+(3*s))

#creating frequency domain object
expression = sy.frequency_domain(x)
expression.name="X(s)"
print("The Expression in frequency Domain: \n",expression)

'''
alternative way:
print("Poles of the expression:",expression.poles())
print("zeros of the expression:",expression.zeros())
'''

#Finding out poles and zeros and displaying
polezero = expression.roots()
print("Poles of the expression:",polezero['poles'])
print("zeros of the expression:",polezero['zeros'])
```

OUTPUT

```
The Expression in frequency Domain:
 (s**2 + 3*s + 1)/(s**3 + 4*s**2 + 3*s)
Poles of the expression: [-3, -1, 0]
zeros of the expression: [-3/2 - sqrt(5)/2, -3/2 + sqrt(5)/2]
```