WIGGLES

EXPERIMENT #01

AUGUST 2022

EXPERIMENT NAME

Generation of Basic signals (Continuous and Discrete) in python using a custom made library i.e. Wiggles

1. SIN WAVE

Theory

1. A **sine wave**, **sinusoidal wave**, or just **sinusoid** is a mathematical curve defined in terms of the **sine** trigonometric function, of which it is the graph.
2. It is a type of continuous wave and also a smooth periodic function. It occurs often in mathematics, as well as in physics, engineering, signal processing and many other fields.

Expression

$$y(t) = A\sin(\omega t+\varphi) = A\sin(2\star\pi\star f\star t+\varphi)$$

- *A, **amplitude**, the peak deviation of the function from zero.*
- *f, **ordinary frequency**, the number of oscillations (cycles) that occur each second of time.*
- *$\omega = 2\pi f$, **angular frequency,** the rate of change of the function argument in units of radians per second.*
- *φ , **phase,** specifies (in radians) where in its cycle the oscillation is at **t = 0.***

When φ is non-zero, the entire waveform appears to be shifted in time by the amount φ/ω seconds. A negative value represents a delay, and a positive value represents an advance.

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

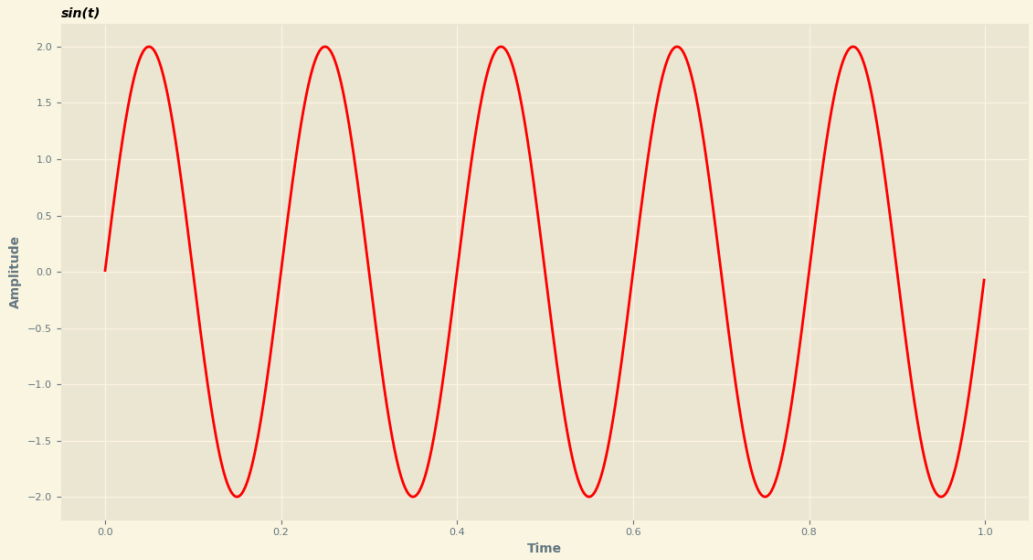
- Generate **sine waves** in python and plot them using the same.

PROGRAM CODE

```
import math
from wiggles import signals as sp

#Adjust using these variables
A = 2
f = 5
w = 2*f*math.pi
Q= 0

#The sin function
def sin(t):
      return A*math.sin((w*t)+Q)

#building the signal
y = sp.continuous(sin)

#Adjusting properties and displaying the signal
y.name="sin(t)"
y.show()

#building the Discrete signal
y2 = sp.continuous(sin,step=0.01)
y2.is_descrete=True

#Adjusting properties and displaying the Discrete signal
y2.name="sin[t]"
y2.show()
```

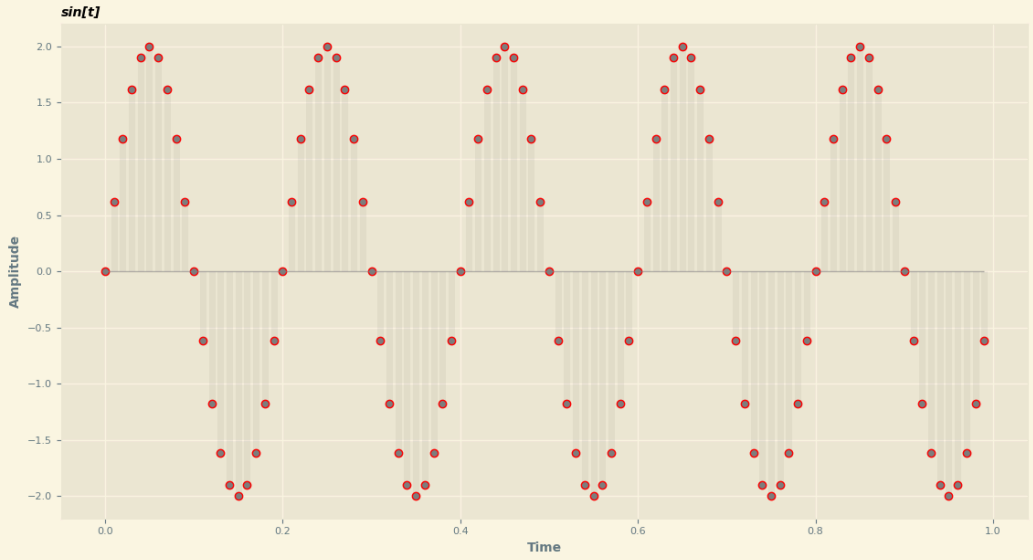
OUTPUT

Plotted graph for continuous signal



Sin Graph plotted in the continuous time domain. Represented through a continuous line graph.

Plotted graph for discrete signal



Sin Graph plotted in the discrete time domain. Represented through a stem graph.

2. COS WAVE

Theory

1. A **cosine wave** is a mathematical curve defined in terms of the **cosine** trigonometric function, of which it is the graph.
2. It is a type of continuous wave and also a smooth periodic function. It occurs often in mathematics, as well as in physics, engineering, signal processing and many other fields.

Expression

$$y(t) = A\cos(\omega t+\varphi) = A\cos(2*\pi*f*t+\varphi)$$

- *A, **amplitude**, the peak deviation of the function from zero.*
- *f, **ordinary frequency**, the number of oscillations (cycles) that occur each second of time.*
- *$\omega = 2\pi f$, **angular frequency,** the rate of change of the function argument in units of radians per second.*
- *φ , **phase,** specifies (in radians) where in its cycle the oscillation is at **t = 0.***

When φ is non-zero, the entire waveform appears to be shifted in time by the amount φ/ω seconds. A negative value represents a delay, and a positive value represents an advance.

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

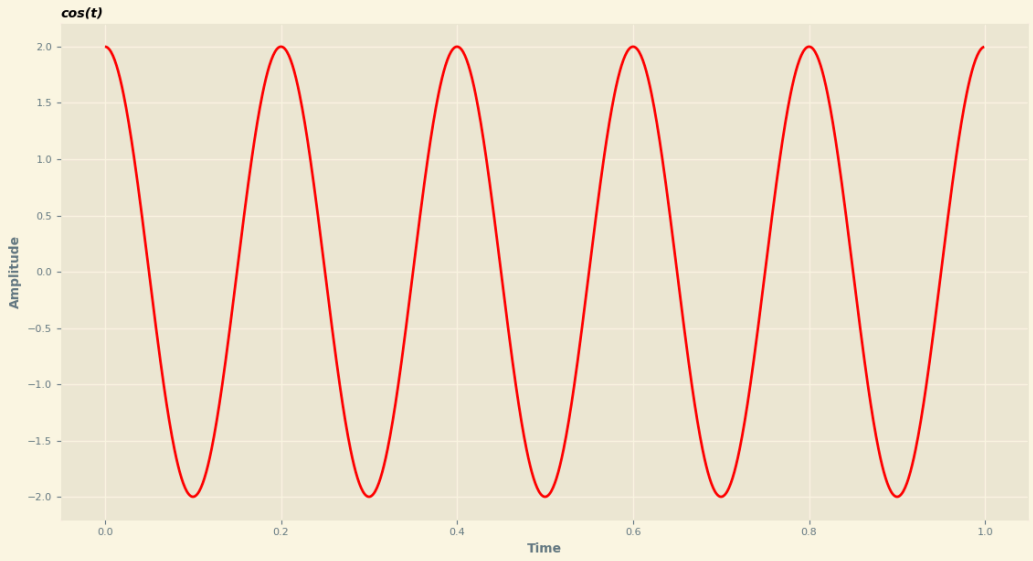
- Generate **cosine waves** in python and plot them using the same.

PROGRAM CODE

```
import math
from wiggles import signals as sp

#Adjust using these variables
A = 2
f = 5
w = 2*f*math.pi
Q= 0

#The cos function
def cos(t):
      return A*math.cos((w*t)+Q)

#building the signal
y = sp.continuous(cos)

#Adjusting properties and displaying the signal
y.name="cos(t)"
y.show()

#building the Discrete signal
y2 = sp.continuous(cos,step=0.01)
y2.is_descrete=True

#Adjusting properties and displaying the Discrete signal
y2.name="cos[t]"
y2.show()
```

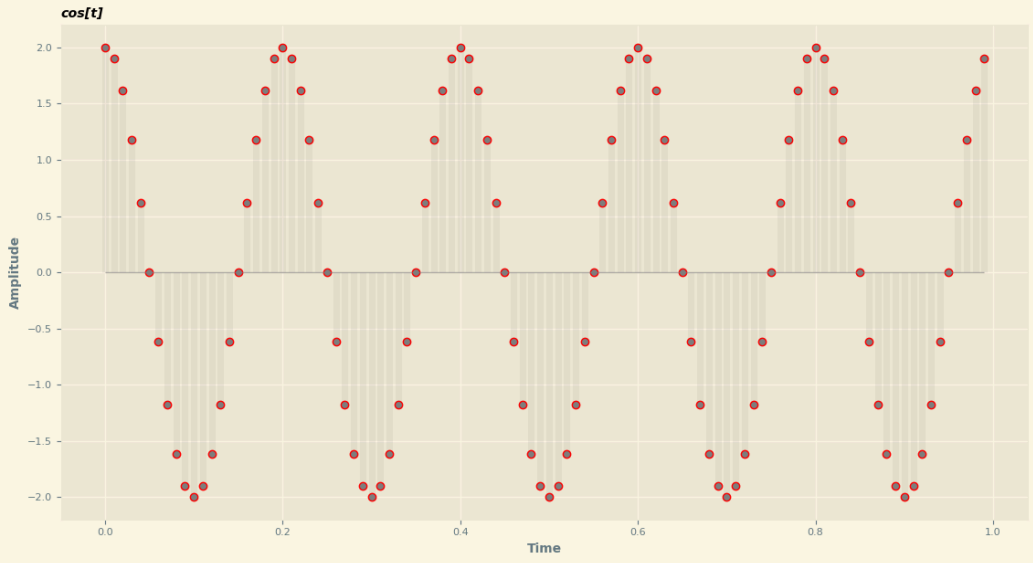
OUTPUT

Plotted graph for continuous signal



Cos Graph plotted in the continuous time domain. Represented through a continuous line graph.

Plotted graph for discrete signal



Sin Graph plotted in the discrete time domain. Represented through a stem graph.

3. EXPONENTIAL CURVE

Theory

4. **The exponential function** is a mathematical function denoted by **f (x) = exp (x) f(x)=\exp(x) or e x e^{x}** (where the argument x is written as an exponent). Unless otherwise specified, the term generally refers to the positive-valued function of a real variable, although it can be extended to the complex numbers or generalized to other mathematical objects like matrices or Lie algebras.

5. The exponential function originated from the notion of exponentiation (repeated multiplication), but modern definitions (there are several equivalent characterizations) allow it to be rigorously extended to all real arguments, including irrational numbers.

Expression

$$y(t) = A\ e^{-\omega t}$$

- *A,* ***amplitude,*** *the peak deviation of the function from zero.*
- ω, ***angular frequency,*** *the rate of change of the function argument in units of radians per second.*

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

- Generate **exponentially growing and exponentially decaying waves** in python and plot them using the same.

PROGRAM CODE

Exponentially Growing

```
import math
from wiggles import signals as sp

#Adjust using these variables
A = 2
a = -4

#The exp function
def exp(t):
      return A*math.exp(-1*a*t)

#building the signal
y = sp.continuous(exp)

#Adjusting properties and displaying the signal
y.name="Exponentially Growing"
y.show()
```

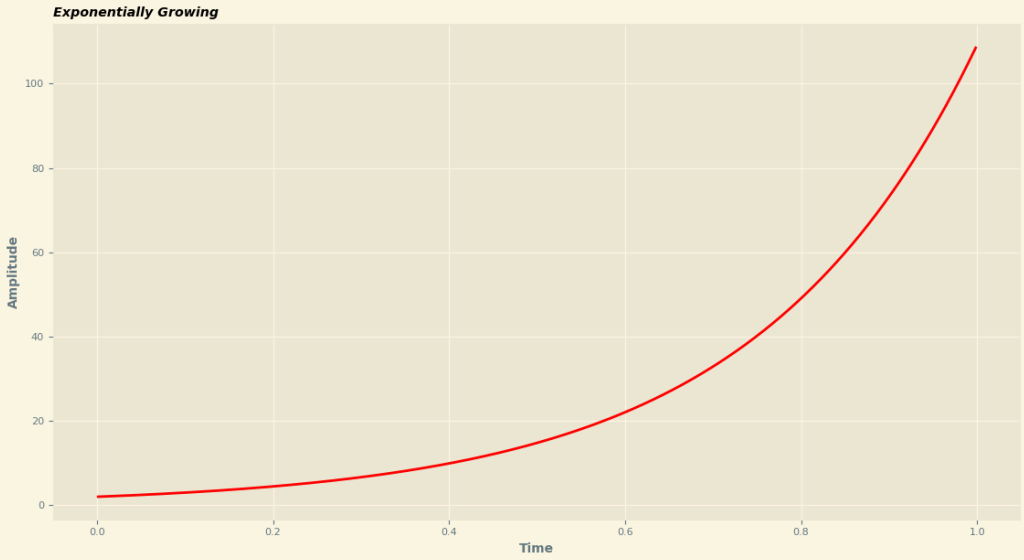
Exponentially Decaying

```
from wiggles import signals as sp

#Adjust using these variables
A = 2
a = 4

#The exp function
def exp(t):
      return A*math.exp(-1*a*t)

#building the signal
y = sp.continuous(exp)

#Adjusting properties and displaying the signal
y.name="Exponentially Decaying"
```

```
y.show()
```

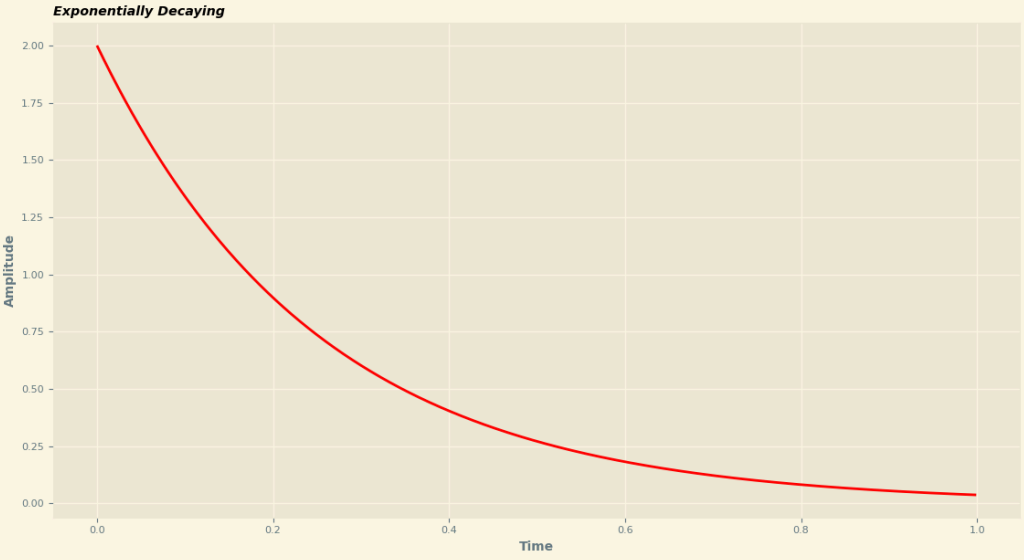
OUTPUT

Plotted graph for exponentially growing signal



Exponentially growing curve plotted in the continuous time domain. Represented through a continuous line graph.

Plotted graph for exponentially decaying signal



Exponentially decaying curve plotted in the continuous time domain.
Represented through a continuous line graph.

4. EXPONENTIALLY GROWING AND DECAYING SINUSOIDAL CURVES

Theory

1. **The exponential function** is a mathematical function denoted by **f (x) = exp (x) f(x)=\exp(x) or e x e^{x}** (where the argument x is written as an exponent).

2. A **sinusoidal wave**, or just **sinusoid,** is a mathematical curve defined in terms of the **sine** trigonometric function, of which it is the graph.

Expression

a. $y(t) = A\ e^{-\omega t}$ [Exponential]

b. $y(t) = Asin(\omega t+\varphi) = Asin(2**f*t+\varphi)$ [Sin]

c. $y(t) = Acos(\omega t+\varphi) = Acos(2*\pi*f*t+\varphi)$ [Cos]

- *A,* ***amplitude****, the peak deviation of the function from zero.*
- *f,* ***ordinary frequency****, the number of oscillations (cycles) that occur each second of time.*
- *$\omega = 2\pi f$,* ***angular frequency,*** *the rate of change of the function argument in units of radians per second.*
- *φ ,* ***phase,*** *specifies (in radians) where in its cycle the oscillation is at* ***t = 0.***

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

- Generate **exponentially growing and exponentially decaying sine and cosine waves** in python and plot them using the same.

PROGRAM CODE

Exponentially Growing sin

```
import math
from wiggles import signals as sp

#Adjust using these variables
A = 1
a = -5
f = 7
w = 2*f*math.pi
Q= 0

#The exp function
def exp(t):
      return A*math.exp(-1*a*t)

#The sin function
def sin(t):
      return A*math.sin((w*t)+Q)

#building and operating on the signal
expwave = sp.continuous(exp)
sinwave = sp.continuous(sin)
expsin = expwave*sinwave

#Adjusting properties and displaying the signal
expwave.name = "Exponentially Growing Wave"
sinwave.name = "Sin Wave"
expsin.name = "Exponentially Growing Sin"

expwave.compare(sinwave,expsin,spacing=0.407)
```

Exponentially Decaying sin

```
import math
from wiggles import signals as sp

#Adjust using these variables
A = 1
a = 5
f = 7
w = 2*f*math.pi
Q= 0

#The exp function
def exp(t):
      return A*math.exp(-1*a*t)

#The sin function
def sin(t):
      return A*math.sin((w*t)+Q)

#building and operating on the signal
expwave = sp.continuous(exp)
sinwave = sp.continuous(sin)
expsin = expwave*sinwave

#Adjusting properties and displaying the signal
expwave.name = "Exponentially Decaying Wave"
sinwave.name = "Sin Wave"
expsin.name = "Exponentially Decaying Sin"

expwave.compare(sinwave,expsin,spacing=0.407)
```

Exponentially Growing cos

```
import math
from wiggles import signals as sp

#Adjust using these variables
A = 1
a = -5
f = 7
w = 2*f*math.pi
Q= 0

#The exp function
def exp(t):
      return A*math.exp(-1*a*t)
```

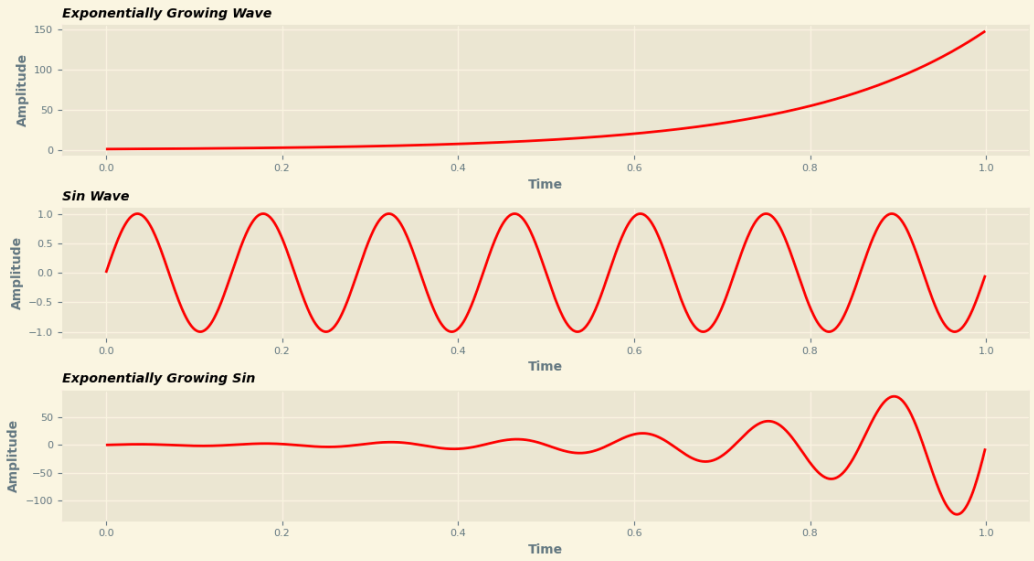
```
#The cos function
def cos(t):
      return A*math.cos((w*t)+Q)

#building and operating on the signal
expwave = sp.continuous(exp)
sinwave = sp.continuous(cos)
expsin = expwave*sinwave

#Adjusting properties and displaying the signal
expwave.name = "Exponentially Growing Wave"
sinwave.name = "Cos Wave"
expsin.name = "Exponentially Growing Cos"

expwave.compare(sinwave,expsin,spacing=0.407)
```

Exponentially Decaying cos

```
import math
from wiggles import signals as sp

#Adjust using these variables
A = 1
a = 5
f = 7
w = 2*f*math.pi
Q= 0

#The exp function
def exp(t):
      return A*math.exp(-1*a*t)

#The cos function
def cos(t):
      return A*math.cos((w*t)+Q)

#building and operating on the signal
expwave = sp.continuous(exp)
sinwave = sp.continuous(cos)
expsin = expwave*sinwave

#Adjusting properties and displaying the signal
expwave.name = "Decaying Growing Wave"
sinwave.name = "Cos Wave"
expsin.name = "Decaying Growing Cos"

expwave.compare(sinwave,expsin,spacing=0.407)
```

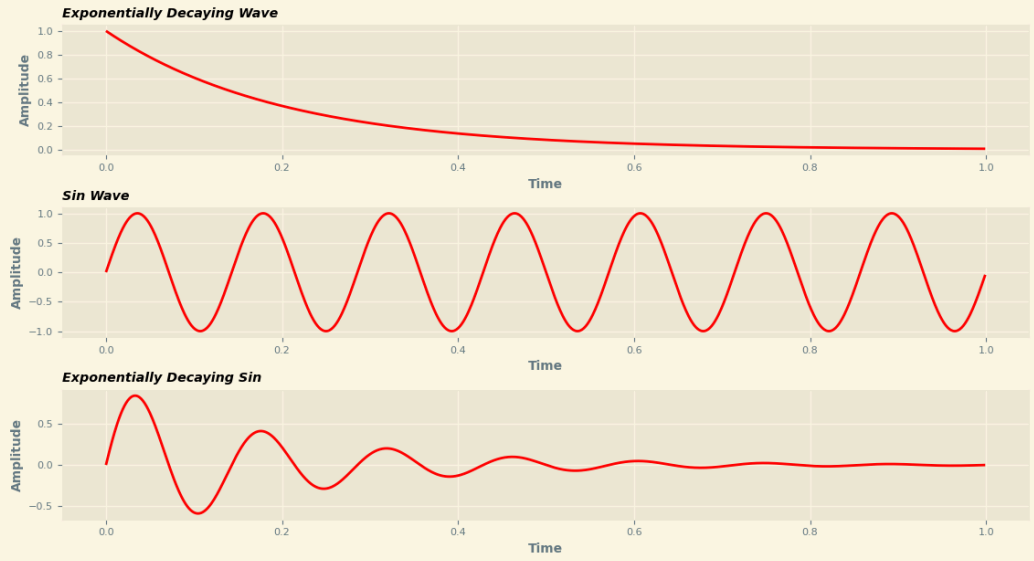
OUTPUT

Plotted graph for exponentially growing sin signal



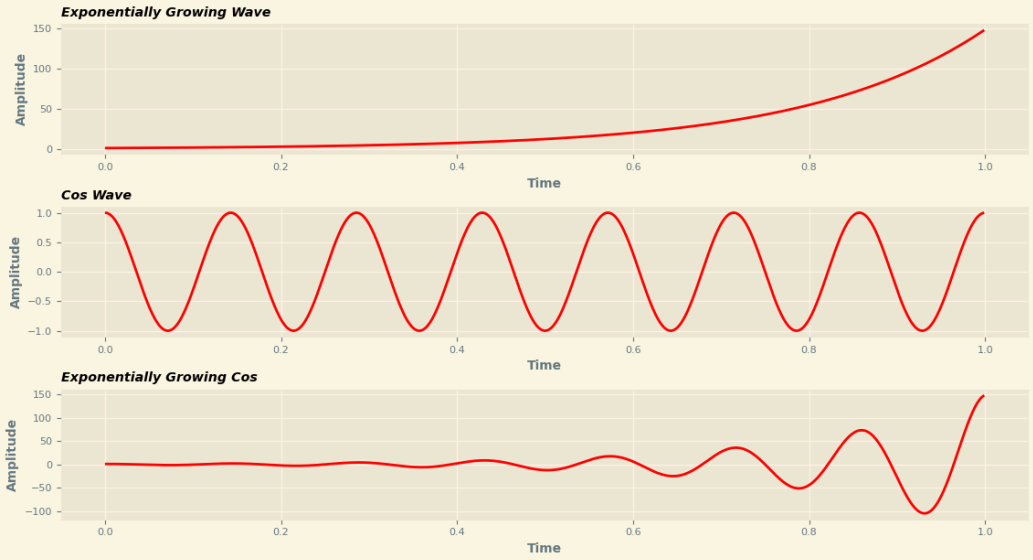
Exponentially growing sin curve plotted in the continuous time domain.
Represented through a continuous line graph.

Plotted graph for exponentially decaying sin signal



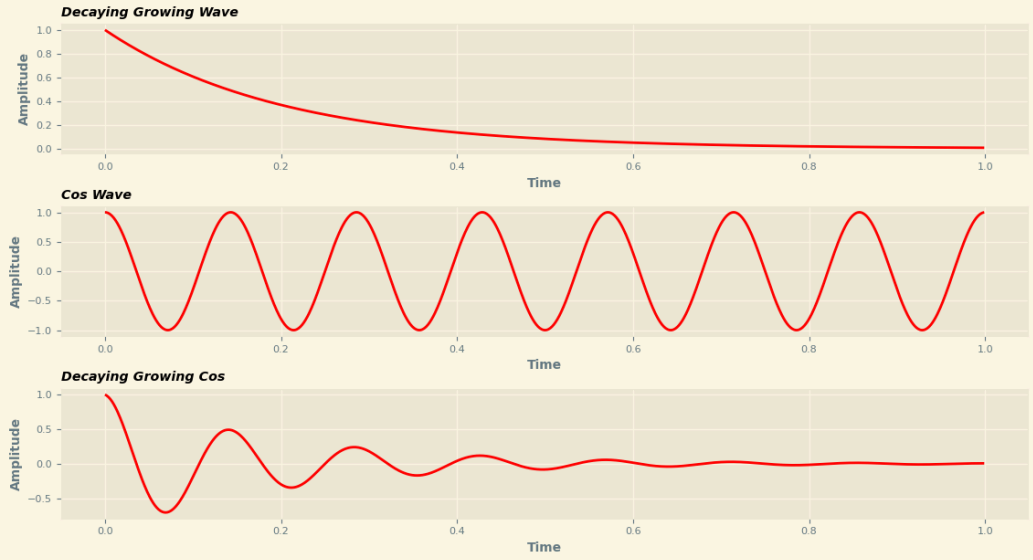
Exponentially decaying sin curve plotted in the continuous time domain.
Represented through a continuous line graph.

Plotted graph for exponentially growing cos signal



Exponentially growing cos curve plotted in the continuous time domain. Represented through a continuous line graph.

Plotted graph for exponentially decaying cos signal



Exponentially decaying cos curve plotted in the continuous time domain. Represented through a continuous line graph.

5. UNIT IMPULSE WITHOUT FUNCTION

Theory

1. An ideal **impulse signal** is a signal that is **zero everywhere** *but* at the **origin (t = 0)**, it is infinitely high.
2. Although, the area of the impulse is finite. The unit impulse signal is the most widely used standard signal used in the analysis of signals and systems.

Expression

$$\delta(t) = \{1 \, for \, t = 0 \; else \; 0\}$$

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

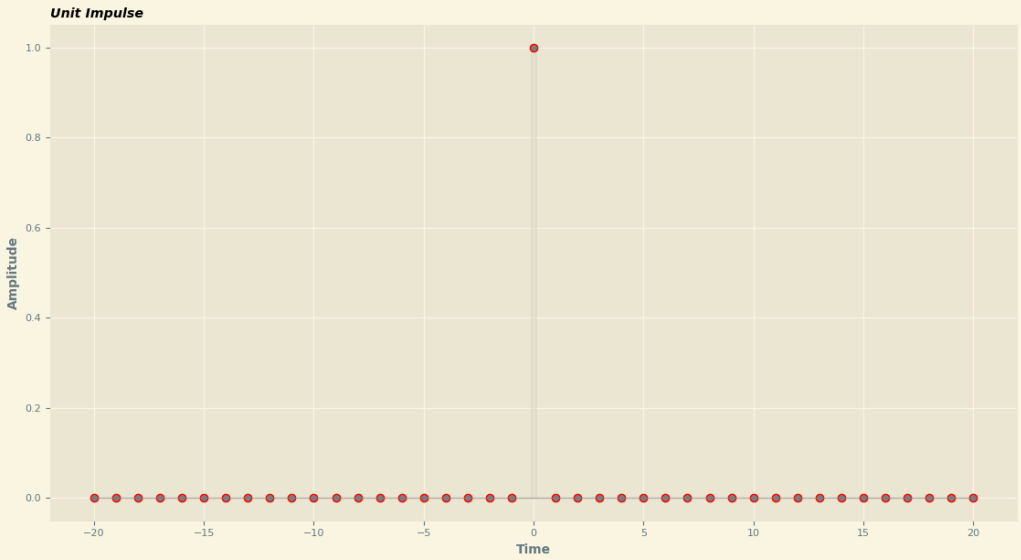
- Generate **unit impulse signal** in python and plot them using the same.

PROGRAM CODE

```
from wiggles import signals as sp

#building amplitude data for unit impulse
length = 20
y=([0]*length)+[1]+([0]*length)

#making signal using the amplitude data 'y' using wiggles and displaying
unitimpulse=sp.discrete(y,-length)
unitimpulse.name="Unit Impulse"
unitimpulse.show()
```

OUTPUT

Plotted graph for unit impulse signal



Unit impulse plotted in the discrete time domain. Represented through a stem graph.

6. UNIT STEP WITHOUT FUNCTION

Theory

1. The **step signal** or **step function** is that type of standard signal which exists only for positive time and it is zero for negative time. In other words, a signal **x(t)** is said to be a step signal if and only if it exists **for t > 0** and **zero for t < 0.** The step signal is an important signal used for analysis of many systems.
2. The **step signal** is equivalent to applying a signal to a system whose magnitude suddenly changes and remains constant forever after application. If we want to obtain a signal which starts at **t = 0**, so that it may have a value of **zero for t < 0**, then we only need to multiply the given signal with the unit step signal **u(t).**

Expression

$$u(t) = \{1\, for\, t \geq 0 \quad 0\ for\ t < 0\}$$

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

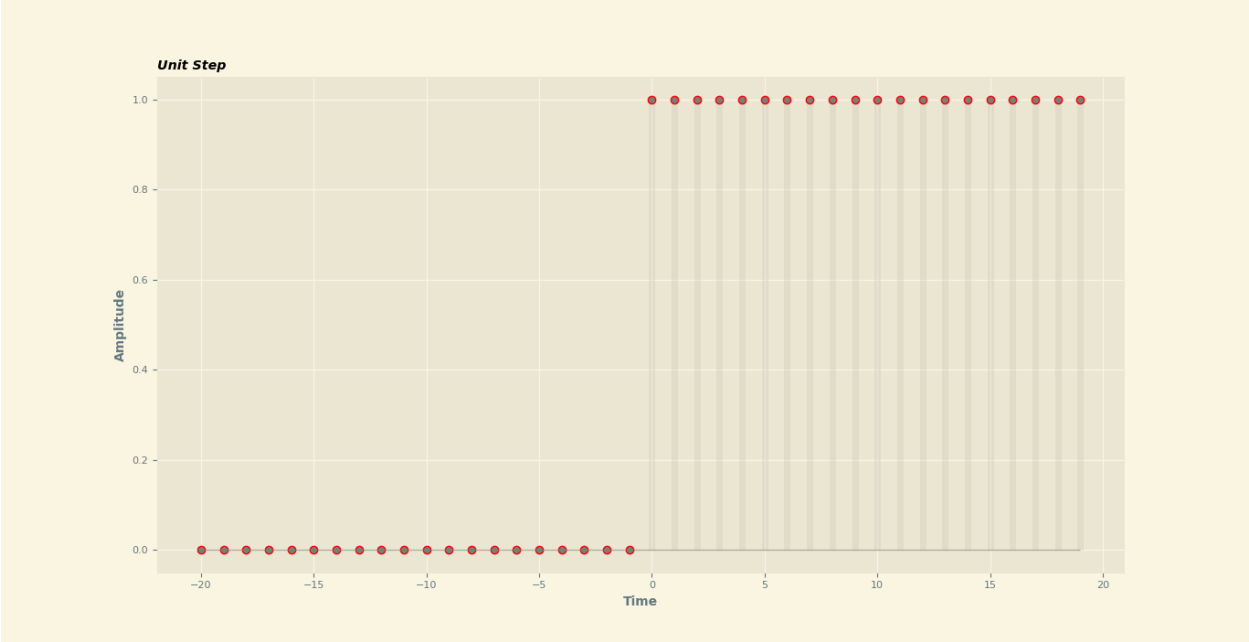
- Generate **unit step signals** in python and plot them using the same.

PROGRAM CODE

```
from wiggles import signals as sp

#building amplitude data for unit Step
length = 20
y=([0]*length)+([1]*length)

#making signal using the amplitude data 'y' using wiggles
unitimpulse=sp.discrete(y,-length)

#Naming and displaying the signal
unitimpulse.name="Unit Step"
unitimpulse.show()
```

OUTPUT

Plotted graph for unit step signal



Unit step plotted in the discrete time domain. Represented through a stem graph.

7. RAMP WITHOUT FUNCTION

Theory

1. A **ramp function** or **ramp signal** is a type of standard signal which starts at t **= 0** and **increases linearly** with time. The unit ramp function has unit slop.
2. The **unit ramp signal** can be obtained by integrating the **unit step signal** with respect to time. In other words, a unit step signal can be obtained by **differentiating the unit ramp signal.**

Expression

$$r(t) = t * \{1 \, for \, t \geq 0 \quad 0 \, for \, t < 0\}$$

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

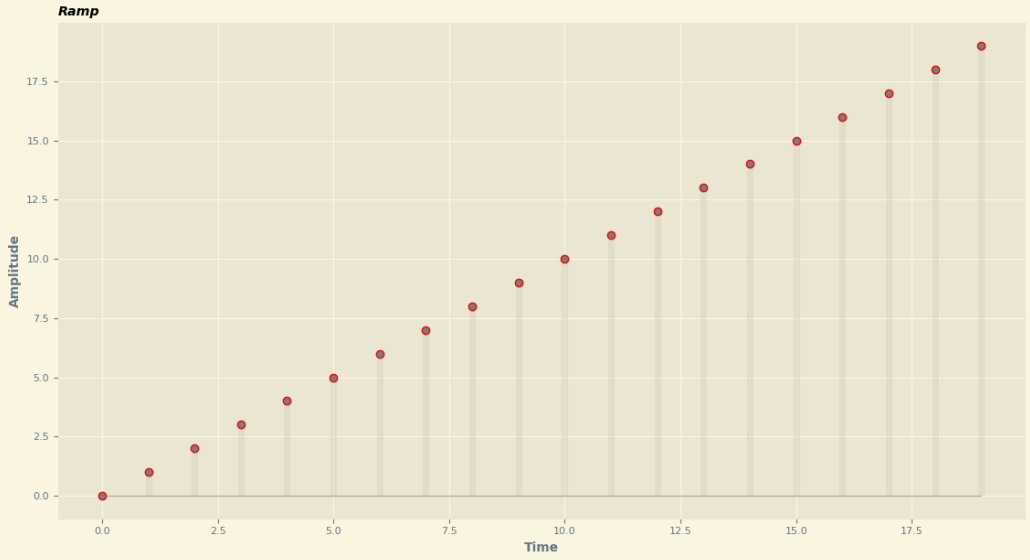
- Generate **ramp signal** in python and plot them using the same.

PROGRAM CODE

```
from wiggles import signals as sp

#building amplitude data for Ramp
length = 20
y=[]
for i in range(length):
      y.append(i)

#making signal using the amplitude data 'y' using wiggles
unitimpulse=sp.discrete(y)

#Naming and displaying the signal
unitimpulse.name="Ramp"
unitimpulse.show()
```

OUTPUT

Plotted graph for ramp signal



Ramp plotted in the discrete time domain. Represented through a stem graph.

CONCLUSION

The above experiments show us how to create continuous and discrete time signals using functions or manually by using arrays (as seen in unit step, unit impulse and ramp programs) and how to display, plot and subplot the signals.