UNIVERSIDAD DEL NORTE
DEPARTAMENTO DE INGENIERÍAS ELÉCTRICA Y ELECTRÓNICA
Asignatura: Señales y Sistemas. Docente: Carlos Cardenas Perez.

Lab 1: Generación y Operación de Señales con PYTHON

Grupo 02: Xelena Maiguel, Gerald Mass, David Vallejo
{xmaiguel, gmass, davallejo} @uninorte.edu.co
15 de septiembre de 2022

Resumen. En este informe se presentará el diseño, generación e implementación de distintas señales para realizar operaciones básicas consigo mismos, como desplazar y escalar. En este, teórico se aplicaran los conceptos que soportan lo que se ejecuta en la interfaz gráfica del usuario usando el lenguaje de programación de alto nivel Python.

Palabras clave: Amplitud, Frecuencia, Señal, desplazamiento, escalamiento, Python.

1. INTRODUCCIÓN

Se puede definir a una señal como una función de una o varias variables que pueden representar una cantidad física; estas señales contienen la información acerca del comportamiento natural de los fenómenos; por ejemplo, las señales eléctricas, acústicas, de video, biológicas, entre otras. Para el caso de una dimensión, la señal se representa mediante la forma x(t), siendo t la variable independiente y x la variable dependiente, estas señales tienen distintas clasificaciones: aleatorias, determinísticas, reales y complejas, continuas, discretas, etc. Estas señales se pueden representar mediante distintas funciones matemáticas como una señal senoidal, exponencial, lineal, cuadrática, etc. Para poder representar estas gráficas se realizan por medio de programas o sistemas que puedan realizarlo los cuales son programados por medio de lenguajes de alto nivel de programación como por ejemplo Python o C + + los cuales por medio de una interfaz gráfica se pueden representar las gráficas de las señales.

2. OBJETIVOS

- Desarrollar e implementar algoritmos para generar, desplazar y escalar señales usando Python.

3. PROCEDIMIENTOS

1. Generación de señales.

En esta primera parte se realizará un programa usando el lenguaje de programación de Python este programa permitirá crear diferentes tipos de señales tal como se describen a continuación:

- Señal senoidal.
- Señal pulso.
- Señal cuadrática.
- Señal exponencial.
- Señal lineal.
- Señal triangular.
- Señal cuadrada.
- Secuencia de impulsos.

Para este programa, el usuario que lo use debe poder seleccionar desde una lista de opciones, la señal a graficar que quiere graficar.

A. Señal Senoidal.

En esta primera señal se hará una señal sinusoidal que se observa a continuación:

$$y(t) = Asin(2\pi ft) \quad (1)$$

Para esta señal el usuario debe ingresar la frecuencia y la amplitud, por último se debe garantizar una buena visualización de la señal. Definiendo el vector tiempo con base a la frecuencia ingresada por el usuario, más específicamente el incremento y el valor final.

Para poder garantizar el una buena visualización de la señal se definirá el periodo de oscilación de la onda y el paso de esta con las siguientes ecuaciones:

$$T = \frac{6}{f} \qquad (2)$$

$$Paso = \frac{1}{(4 \times 2\pi f)} \quad (3)$$

Finalmente se hará el código de Python usando Visual Studio Code como se observa en la Figura 1.

```
if selectedbox == "Señal Senoidal":
    st.title("Señal Senoidal")
    f = st.number_input('Ingrese frecuencia f:')
    amplitude = st.number_input('Ingrese amplitud A:')
    T=6/f
    paso=1/(4*2*np.pi*f)
    t = np.arange(0,T,paso)
    y = amplitude*np.sin(2*np.pi*f*t)
    fig, ax = plt.subplots()
    pl.xlabel('Tiempo [s]')
    pl.ylabel('Amplitud')
    pl.title('Senoidal')
    ax.plot(t,y)
    st.pyplot(fig)
```

Figura 1. Código para graficar la señal senoidal.

B. Señal Pulso.

Para esta segunda señal se generará una señal pulso o una señal cajón, para esta señal la amplitud y el ancho de la señal debe ser ingresado por el usuario, para esto se utilizara como límite a el ancho de la señal, se establecerá a el punto del eje en 0 y se hará un escalón unitario en -ancho/2 y ancho/2 como se observa en la siguiente ecuación:

$$y(t) = [U(t + \frac{Ancho}{2}) - U(t - \frac{Ancho}{2})] \, (4)$$

Finalmente se hará el código de Python usando Visual Studio Code como se observa en la Figura 2.

```
st.title("Señal Pulso")
amplitud=st.number_input("introduce el valor de amplitud:")
ancho=st.number_input("introduce el valor de ancho:")
t= np.arange(-ancho,ancho,0.01)
y=np.zeros(len(t))
Y= t*0
y[(t>-ancho/2) & (t< ancho/2)]=amplitud
fig,ax=plt.subplots()
ax.plot(t,y)
ax.set_title("Función Pulso")
ax.set_xlabel("t[s]")
ax.set_ylabel("Amplitud")
ax.grid(True)
st.pyplot(fig)
```

Figura 2. Código para graficar la señal pulso.

C. Señal Cuadrática.

Para esta tercera señal será una señal cuadrática como se observa a continuación:

$$y(t) = at^2 + bt + c \, (5)$$

Para esta tercera señal el usuario debe ingresar el valor de las constantes así como la frecuencia y la amplitud para poder graficar la señal como se muestra en la siguiente ecuación:

$$y(t) = (at^2 + bt + c) \times amplitud \, (6)$$

Finalmente se hará el código de Python usando Visual Studio Code como se observa en la Figura 3.

```
st.write("Se presenta la ecuacion y(t)= at^2+bt+c")
a=st.number_input("introduce el valor de a:")
b=st.number_input("introduce el valor de b:")
c=st.number_input("introduce el valor de c:")
amplitud=st.number_input("introduce el valor de amplitud: ")
F=st.number_input("introduce el valor de frecuencia:")
T=1/F
t=np.arange(0,F,0.01)
y=(a*t**2+b*t+c)*amplitud

fig,ax=plt.subplots()
ax.plot(t,y)
ax.set_title("Función cuadratica")
ax.set_xlabel("Eje x")
ax.set_ylabel("Eje y")
ax.grid(True)
st.pyplot(fig)
```

Figura 3. Código para graficar la señal cuadrática.

D. Señal Exponencial.

Para esta cuarta señal se hará una señal exponencial como se observa en la siguiente ecuación:

$$y(t) = Ae^{-bt} \, (7)$$

cabe destacar que b es negativo por lo que la gráfica de la señal exponencial estará invertida, para esta señal el usuario tendrá que ingresar el valor de A y el valor de b para poder graficar la señal. Finalmente se realizó el código de Python usando Visual Studio Code como se observa en la Figura 4.

```
st.title('Señal Exponencial')
A = st.number_input('Ingrese el valor de A:')
b= st.number_input('Ingrese el valor de b:')
t = np.arange(-0.25,0.25,0.01)
y = A*np.exp(-b*t)
fig, ax = plt.subplots()
pl.xlabel('t [s]')
pl.ylabel('Amplitud')
pl.title('Señal Exponencial')
ax.plot(t,y)
ax.grid(True)
st.pyplot(fig)
```

Figura 4. Código para graficar la señal exponencial.

E. Señal Lineal.

Para esta quinta señal se hará una señal lineal como se observa en la siguiente ecuación:

$$y(t) = mt + b \quad (8)$$

Para esta señal el usuario tendrá que ingresar el valor de la pendiente m y el valor de b para poder graficar la señal. Finalmente se realizó el código de Python usando Visual Studio Code como se observa en la Figura 5.

```
st.title('Prueba de señal Lineal')
m = st.number_input('ingrese el valor de la pendiente m:')
b= st.number_input('ingrese el valor de b:')
t = np.arange(-5, 5, 0.01)
y = m*t + b
fig, ax = plt.subplots()
ax.plot(t,y)
ax.grid(True)
st.pyplot(fig)
```

Figura 5. Código para graficar la señal lineal.

F. Señal Triangular.

Para esta sexta señal se hará una señal triangular, para esta señal el usuario deberá ingresar la amplitud y la frecuencia para que pueda graficar la señal. En este caso para que la señal pueda realizar la repetición de la señal triangular se usará el comando *"sawtooth"* como se puede observar en la Figura 6.

```
y = sg.sawtooth(2*np.pi*F*t)*amplitud
```

Figura 6. Código de ecuación de la señal usando el comando sawtooth.

Finalmente se realizó el código de Python usando Visual Studio Code como se observa en la Figura 7.

```
st.write("Se presenta la funcion triangular")
amplitud=st.number_input("introduce el valor de amplitud:")
F=st.number_input("introduce el valor de frecuencia: ")
t=np.arange(0,F,0.01)
y = sg.sawtooth(2*np.pi*F*t)*amplitud
fig,ax=plt.subplots()
ax.plot(t,y)
ax.set_title("Función triangular")
ax.set_xlabel("Eje x")
ax.set_ylabel("Eje y")
ax.grid(True)
st.pyplot(fig)
```

Figura 7. Código para graficar la señal triangular.

G. Señal Cuadrada.

Para esta séptima señal se hará una señal cuadrada, al igual que la señal triangular el usuario deberá ingresar la amplitud y la frecuencia para que pueda graficar la señal. En este caso para que la señal pueda realizar la repetición de la señal cuadrada se usará el comando *"square"* como se puede observar en la Figura 8.

```
y=sg.square(2*np.pi*F*t)*amplitud
```

Figura 8. Código de ecuación de la señal usando el comando square.

Finalmente se realizó el código de Python usando Visual Studio Code como se observa en la Figura 9.

```
st.write("Se presenta la funcion cuadrada")
amplitud=st.number_input("introduce el valor de amplitud:  ")
F=st.number_input("introduce el valor de frecuencia:  ")
t=np.arange(0,F,0.01)
y=sg.square(2*np.pi*F*t)*amplitud
fig,ax=plt.subplots()
ax.plot(t,y)
ax.set_title("Función Cuadrada")
ax.set_xlabel("Eje x")
ax.set_ylabel("Eje y")
ax.grid(True)
st.pyplot(fig)
```

Figura 9. Código para graficar la señal cuadrada.

H. Secuencia de Pulsos.

Para esta octava y última señal se hará una señal secuencia de pulsos en donde el usuario introducirá los valores de los pulsos, también deberá introducir la posición en la que estarán los valores de los pulsos introducidos anteriormente, estos valores introducidos deben estar separados con una "," . Finalmente se realizó el código de Python usando Visual Studio Code como se observa en la Figura 10.

```
else:
    if selectedbox == "Secuencia de Pulsos":
        st.sidebar.write("se presenta la secuencia de pulsos")
        y = st.text_input("introduzca los valores de los pulsos y separelos con coma
        y = y.split(',')
        y =[int(i) for i in y]
        t = st.text_input("introduzca la posicion de cada pulso y separelo por coma
        t = t.split(',')
        t =[int(i) for i in t]
        fig,ax=plt.subplots()
        ax.stem(t,y)
        ax.set_title("secuencia de pulsos")
        ax.set_xlabel("Eje t")
        ax.set_ylabel("Eje y")
        ax.grid(True)
        st.pyplot(fig)
    end_fill
```

Figura 10. Código para graficar la secuencia de impulso.

2. Operación con señales.

En esta segunda parte de la práctica se realizará desplazamiento en el tiempo y escalamiento en amplitud y en el tiempo para cada una de las señales que se asignaron para graficar, así mismo se debe mostrar el proceso de desplazamiento y escalamiento en tiempo real con una animación que muestre el proceso.

A. Escalamiento en Amplitud.

Para esta parte se realizará un escalamiento en amplitud, este tipo de escalamiento la señal sólo será afectada la amplitud pero no en su desplazamiento en el tiempo por lo que la posición de la señal será igual. El usuario deberá introducir un valor "k" en el que si el valor es positivo hará un ensanchamiento de la amplitud mientras que si el valor es negativo comprimirá la señal en amplitud.

Finalmente se procedió a realizar el proceso de escalamiento de la señal de las distintas señales como se observan en las siguientes Figuras:

```
if k>0:
    y = (k*amplitude)*np.sin(2*np.pi*f*t)
    fig, ax = plt.subplots()
    pl.xlabel('Tiempo [s]')
    pl.ylabel('Amplitud')
    pl.title('Senoidal')
    ax.plot(t,y)
    st.pyplot(fig)
else:
    y = (amplitude/(-k))*np.sin(2*np.pi*f*t)
    fig, ax = plt.subplots()
    pl.xlabel('Tiempo [s]')
    pl.ylabel('Amplitud')
    pl.title('Senoidal')
    ax.plot(t,y)
    st.pyplot(fig)
end_fill
```

Figura 11. Código para escalar en amplitud la señal senoidal.

```
if k>0:
    y = (k*A)*np.exp(-b*t)
    fig, ax = plt.subplots()
    pl.xlabel('t [s]')
    pl.ylabel('Amplitud')
    pl.title('Señal Exponencial')
    ax.plot(t,y)
    ax.grid(True)
    st.pyplot(fig)
else:
    y = (A/(-k))*np.exp(-b*t)
    fig, ax = plt.subplots()
    pl.xlabel('t [s]')
    pl.ylabel('Amplitud')
    pl.title('Señal Exponencial')
    ax.plot(t,y)
    ax.grid(True)
    st.pyplot(fig)
```

Figura 12. Código para escalar en amplitud la señal exponencial.

```
if k>0:
    y = (m*k)*t + b
    fig, ax = plt.subplots()
    ax.plot(t,y)
    ax.grid(True)
    st.pyplot(fig)
else:
    y = (m/(-k))*t + b
    fig, ax = plt.subplots()
    ax.plot(t,y)
    ax.grid(True)
    st.pyplot(fig)
end_fill
```

Figura 13. Código para escalar en amplitud la señal lineal.

```
if k>0:
    y=(a*t**2+b*t+c)*(amplitud*k)
    fig,ax=plt.subplots()
    ax.plot(t,y)
    ax.set_title("Función cuadratica")
    ax.set_xlabel("Eje x")
    ax.set_ylabel("Eje y")
    ax.grid(True)
    st.pyplot(fig)
else:
    y=(a*t**2+b*t+c)*(amplitud/(-k))
    fig,ax=plt.subplots()
    ax.plot(t,y)
    ax.set_title("Función cuadratica")
    ax.set_xlabel("Eje x")
    ax.set_ylabel("Eje y")
    ax.grid(True)
    st.pyplot(fig)
end_fill
```

Figura 14. Código para escalar en amplitud la señal cuadrática.

```
if k>0:
    y[(t>-ancho/2) & (t< ancho/2)]=amplitud
    fig,ax=plt.subplots()
    ax.plot(t,(y*k))
    ax.set_title("Función Pulso")
    ax.set_xlabel("t[s]")
    ax.set_ylabel("Amplitud")
    ax.grid(True)
    st.pyplot(fig)
else:
    y[(t>-ancho/2) & (t< ancho/2)]=amplitud
    fig,ax=plt.subplots()
    ax.plot(t,(y/(-k)))
    ax.set_title("Función Pulso")
    ax.set_xlabel("t[s]")
    ax.set_ylabel("Amplitud")
    ax.grid(True)
    st.pyplot(fig)
end_fill
```

Figura 15. Código para escalar en amplitud la señal de pulso.

```
if k>0:
    y = sg.sawtooth(2*np.pi*F*t)*(amplitud*k)
    fig,ax=plt.subplots()
    ax.plot(t,y)
    ax.set_title("Función triangular")
    ax.set_xlabel("Eje x")
    ax.set_ylabel("Eje y")
    ax.grid(True)
    st.pyplot(fig)
else:
    y = sg.sawtooth(2*np.pi*F*t)*(amplitud/(-k))
    fig,ax=plt.subplots()
    ax.plot(t,y)
    ax.set_title("Función triangular")
    ax.set_xlabel("Eje x")
    ax.set_ylabel("Eje y")
    ax.grid(True)
    st.pyplot(fig)
end_fill
```

Figura 16. Código para escalar en amplitud la señal triangular.

```
if k>0:
    y=sg.square(2*np.pi*F*t)*(amplitud*k)
    fig,ax=plt.subplots()
    ax.plot(t,y)
    ax.set_title("Función Cuadrada")
    ax.set_xlabel("Eje x")
    ax.set_ylabel("Eje y")
    ax.grid(True)
    st.pyplot(fig)
else:
    y=sg.square(2*np.pi*F*t)*(amplitud/(-k))
    fig,ax=plt.subplots()
    ax.plot(t,y)
    ax.set_title("Función Cuadrada")
    ax.set_xlabel("Eje x")
    ax.set_ylabel("Eje y")
    ax.grid(True)
    st.pyplot(fig)
end_fill
```

Figura 17. Código para escalar en amplitud la señal cuadrada.

B. Desplazamiento.

En esta segunda parte se realizará un desplazamiento a través del tiempo en este desplazamiento sólo será afectado el periodo en el que transcurre o se ejecuta la señal por lo que la forma de la señal como su amplitud no será afectada . En este caso el usuario deberá introducir un valor "m" en el que si el valor es positivo hará un desplazamiento hacia la izquierda mientras que si el valor es negativo hará un desplazamiento hacia la derecha.

Finalmente se procedió a realizar el proceso de escalamiento de la señal de las distintas señales como se observan en las siguientes Figuras:

```
if selectedbox == "Señal Cuadrada":
    st.write("Señal cuadrada")
    amplitud=st.number_input("introduce el valor de amplitud:  ")
    F=st.number_input("introduce el valor de frecuencia:  ")
    t=np.arange(0,F,0.01)
    y=sg.square(2*np.pi*F*t)*amplitud
    fig,ax=plt.subplots()
    t2=t-m
    ax.plot(t2,y)
    ax.plot(t,y)
    ax.set_title("Función Cuadrada")
    ax.set_xlabel("Eje x")
    ax.set_ylabel("Eje y")
    ax.grid(True)
    st.pyplot(fig)
```

Figura 18. Código para desplazar en el tiempo la señal cuadrada.

```
if selectedbox == "Señal Triangular":
    st.write("Señal triangular")
    amplitud=st.number_input("introduce el valor de amplitud:")
    F=st.number_input("introduce el valor de frecuencia: ")
    t=np.arange(0,F,0.01)
    y = sg.sawtooth(2*np.pi*F*t)*amplitud
    fig,ax=plt.subplots()
    t2=t-m
    ax.plot(t2,y)
    ax.plot(t,y)
    ax.set_title("Función triangular")
    ax.set_xlabel("Eje x")
    ax.set_ylabel("Eje y")
    ax.grid(True)
    st.pyplot(fig)
```

Figura 19. Código para desplazar en el tiempo la señal triangular.

```
if selectedbox == "Señal Pulso":
    st.write("Señal pulso")
    amplitud=st.number_input("introduce el valor de amplitud:   ")
    ancho=st.number_input("introduce el valor de ancho:")
    t= np.arange(-ancho,ancho,0.01)
    y=np.zeros(len(t))
    Y= t*0
    y[(t>-ancho/2) & (t< ancho/2)]=amplitud
    fig,ax=plt.subplots()
    t2=t-m
    ax.plot(t2,y)
    ax.plot(t,y)
    ax.set_title("Función Pulso")
    ax.set_xlabel("t[s]")
    ax.set_ylabel("Amplitud")
    ax.grid(True)
    st.pyplot(fig)
```

Figura 20. Código para desplazar en el tiempo la señal pulso.

```
if selectedbox == "Señal Cuadrática":
    st.write("Señal cuadratica y(t)= at^2+bt+c")
    a=st.number_input("introduce el valor de a:")
    b=st.number_input("introduce el valor de b:")
    c=st.number_input("introduce el valor de c:")
    amplitud=st.number_input("introduce el valor de amplitud: ")
    F=st.number_input("introduce el valor de frecuencia:")
    T=1/F
    t=np.arange(0,F,0.01)
    y=(a*t**2+b*t+c)*amplitud
    fig,ax=plt.subplots()
    ax.plot(t,y)
    t2=t-m
    ax.plot(t2,y)
    ax.set_title("Función cuadratica")
    ax.set_xlabel("Eje x")
    ax.set_ylabel("Eje y")
    ax.grid(True)
    st.pyplot(fig)
```

Figura 21. Código para desplazar en el tiempo la señal cuadrática.

```
if selectedbox ==  "Señal Lineal":
    st.title(' Señal Lineal')
    m = st.number_input('ingrese el valor de la pendiente m:')
    b= st.number_input('ingrese el valor de b:')
    t = np.arange(-5, 5, 0.01)
    y = m*t + b
    fig, ax = plt.subplots()
    ax.plot(t,y)
    t2=t-m
    ax.plot(t2,y)
    ax.grid(True)
    st.pyplot(fig)
```

Figura 22. Código para desplazar en el tiempo la señal lineal.

```
if selectedbox == "Señal Exponencial":
    st.title(' Señal Exponencial')
    A = st.number_input('Ingrese el valor de A:')
    b= st.number_input('Ingrese el valor de b:')
    t = np.arange(-0.25,0.25,0.01)
    y = A*np.exp(-b*t)
    fig, ax = plt.subplots()
    pl.xlabel('t [s]')
    pl.ylabel('Amplitud')
    pl.title('Señal Exponencial')
    ax.plot(t,y)
    t2=t-m
    ax.plot(t2,y)
    ax.grid(True)
    st.pyplot(fig)
```

Figura 23. Código para desplazar en el tiempo la señal exponencial.

```
if selectedbox == "Señal Senoidal":
    st.title("Prueba señal senoidal")
    f = st.number_input('Ingrese frecuencia f:')
    amplitude = st.number_input('Ingrese amplitud A:')
    T=6/f
    paso=1/(4*2*np.pi*f)
    t = np.arange(0,T,paso)
    y = amplitude*np.sin(2*np.pi*f*t)
    fig, ax = plt.subplots()
    pl.xlabel('Tiempo [s]')
    pl.ylabel('Amplitud')
    pl.title('Senoidal')
    ax.plot(t,y)
    st.pyplot(fig)
    t2=t-m
    ax.plot(t2,y)
    st.pyplot(fig)
```

Figura 24. Código para desplazar en el tiempo la señal senoidal.

C. Escalamiento en el tiempo.

Para esta tercera parte se realizará un escalamiento en el tiempo, este tipo de escalamiento la señal según se explica en la teoría si k<0 se refleja la señal es decir que se invierte la señal, si k>1 se decrementa k veces la señal y si k esta entre 0 y 1 (0<k<1) la señal se expande. El usuario deberá introducir un valor "k" en el que si el valor es negativo la señal se refleja, si el valor de k es mayor que 1 se comprimira la señal y finalmente si el valor de k está entre 0 y 1 la señal se ensanchara.

Finalmente se procedió a realizar el proceso de escalamiento de las distintas señales como se observan en las siguientes Figuras:

4. CONCLUSIONES

Al finalizar la realización de este laboratorio, se pudo apreciar el comportamiento de las distintas señales básicas asignadas y estudiadas con anterioridad. Además fue posible comprender de manera clara en cómo las distintas señales que se crearon pueden ser generadas y estudiadas con mayor profundidad (exponenciales, sinusoidales, lineales, cuadradas, etc). Del mismo modo, el uso de variedad de librerías y aplicaciones (Visual Studio Code, Anaconda) permitió un mejor manejo en la ejecución con el lenguaje de alto nivel de Python con el fin de aprovechar las opciones que esta nos permite realizar. Por último, este taller permitió llevar a la práctica los conocimientos aprendidos sobre la transformación de señales (escalamiento y desplazamiento), lastimosamente las animaciones que no muestran el proceso de escalamiento y desplazamiento, esto debido a problemas que generaba en el código del programa, aun así el programa puede realizar los distintos procesos de desplazamiento y escalamiento, en donde se puede observar de una forma más precisa que independiente del orden de los procedimientos, el resultado es exactamente el mismo.

REFERENCIAS

[1] J. Tello Portillo, Introducción a las señales y sistemas. Barranquilla (Colombia): Universidad del Norte Editorial, 2017.