

Práctica 2

Almeraya, Kimberly., Tinoco, Sergio., De Los Ríos, Flavio.
Instituto Politécnico Nacional, Escuela Superior de Cómputo.

Procesamiento de Señales

02 de Noviembre de 2022

En esta práctica se realiza una aplicación GUI en App Designer, herramienta de Matlab para poder representar de manera gráfica los conocimientos vistos en clase, tratándose de las operaciones básicas sobre señales como lo son el escalamiento de amplitud (señal en tiempo continuo de amplitud y duración, suma, multiplicación, diferenciación, integración, escalamiento de tiempo, reflexión y corrimiento en tiempo o desplazamiento a través de la variable dependiente de las señales básicas.

Palabras clave: aplicación, variable dependiente, variable independiente, operación.

I. INTRODUCCIÓN

En la práctica se incluyeron las operaciones básicas de procesamiento de señales como lo son el escalamiento de amplitud, la suma, la multiplicación, la diferenciación, la integración, el escalamiento de tiempo, la reflexión, el corrimiento en tiempo o desplazamiento además también se incluyeron la regla de la precedencia para el corrimiento en el tiempo y el escalamiento en el tiempo.

Para ello usamos como base la aplicación previamente generada en el examen, agregando nuevos componentes a la GUI para que el usuario pudiera seleccionar si desea realizar operaciones con respecto a la variable dependiente o independiente sobre las señales elegidas.

App Designer es un entorno de desarrollo interactivo para diseñar una aplicación y programar su comportamiento. Proporciona una versión totalmente integrada del editor de MATLAB y un

gran conjunto de componentes interactivos de la IU. Gracias al software Matlab y sus herramientas para visualizar gráficos, podemos simular las diferentes señales.

El escalonado de la señal continua consiste en ajustar toda la muestra dentro del rango admitido por el procesador para luego ser digitalizada por el dispositivo mediante operadores internos.

Cuando una señal se describe como suma de señales senoidales, se dice que estas señales las componen y se dice que las frecuencias de dichas señales senoidales están presentes en la señal resultante.

OBJETIVO Y PLANTEAMIENTO DEL PROBLEMA

El problema de esta práctica es observar el cambio en las señales en función de sus variables, usando App Designer, herramienta de Matlab a través de las fórmulas vistas en la clase.

El objetivo de esta práctica es crear una app GUI que graficara para poder visualizar cómo cambian los gráficos de las señales resultantes de las operaciones básicas en función de las variables independientes o dependientes.

II. DESARROLLO EXPERIMENTAL

Para el desarrollo de la App GUI, se tuvo presente la “plantilla” de la aplicación en App Designer previamente diseñada con las señales básicas, en la cual se implementaron nuevos componentes, ya que se requiere realizar operaciones con respecto a la variable dependiente o independiente sobre una de las señales seleccionadas.

Y se usaron variables globales para trabajar en todo el entorno.

```
properties (Access = private)
    f      %Parametro para la frecuencia
    a      %Parametro para la amplitud
    f2     %Parametro para la frecuencia
    a2     %Parametro para la amplitud
    c='r'; %Color de la línea en el plot
    r=1;   %Parametro para la reflexion
    d      % Parametro para corrimiento
    d2     % Parametro para corrimiento
    Fs     %Frecuencia muestral
    x      %Variable independiente
    x2     %Variable independiente 2
    xint;  %Parametro para integrar
    num_signal=0; %Numero de señales
    aux_res %Variable auxiliar para la suma/multiplicacion
end
```

Captura de pantalla 1. Variables globales.

Se invocan las funciones en cada operación, sin embargo solo varía el orden de uso. También se tomaron en cuenta condicionamientos de las entradas, por ejemplo, en el caso de las operaciones con variables dependientes, tenemos el escalamiento de amplitud, la suma, la multiplicación, la diferenciación y la integración. Y solo se muestra esto mientras que se haya seleccionado la primera señal básica, entonces se selecciona la operación para tener una entrada, luego un proceso seleccionando la operación. En caso de que no se requiera

una operación de variable dependiente, se selecciona la independiente. Entonces se activan las flags correspondientes que habilitan cada botón, pero no la de la segunda señal. Igual es posible habilitar todas las flags marcando ambas casillas.

```
% Callbacks that handle component events
methods (Access = private)

% Code that executes after component creation
function startupFcn(app)
    app.f=app.EscalamientodetiempoSlider.Value;
    app.a=app.AmplitudSlider_2.Value;
    app.f2=app.Escalamientodetiempo2Slider.Value;
    app.a2=app.Amplitud2Slider.Value;
    app.d=app.ValorSpinner.Value;
    app.d2=app.ValorSpinner_2.Value;
```

Captura de pantalla 2. Valores de eventos.

Después de seleccionar la primera señal, la operación a realizar, dependiendo de cuál se haya seleccionado, podrá o no seleccionar una segunda señal, cual sea su caso, presiona el botón de 'Resultado', que es la flag para que se lleve a cabo la operación y poder visualizarla.

Las validaciones en la parte del código también están presentes en el diseño a través de eventos, ya sea que se quiera ver el resultado en tiempo discreto o continuo y el color. Se está presente que si un botón u otro se selecciona, se cambia al instante cuál sea la señal que se esté visualizando. Al menos en esta parte. Se recomienda dejar esto para el final aunque es posible cambiar desde la selección de la primera señal.

```
% Value changed function: ColordelaGraficaListBox
function ColordelaGraficaListBoxValueChanged(app, event)
    value = app.ColordelaGraficaListBox.Value;
    switch value
        case "Azul"
            app.c='b';
        case "Rojo"
            app.c='r';
        case "Verde"
            app.c='g';
        case "Amarillo"
            app.c='y';
        case "Cian"
            app.c='c';
    end
    ListBoxValueChanged(app,event);
```

Captura de pantalla 3. Evento cambio de color.

Se implementaron los procedimientos necesarios para las 5 operaciones posibles a realizar según el usuario, entre ellas tenemos el escalamiento de amplitud, suma, multiplicación, diferenciación e integración, se pueden visualizar las funciones en las siguientes capturas de pantalla:

```
% Value changed function: VDependienteCheckBox
function VDependienteCheckBoxValueChanged(app, event)
    value = app.VDependienteCheckBox.Value;
    if value==0
        app.SumaButton.Enable="off";
        app.MultiplicacinButton.Enable="off";
        app.DiferenciacinButton.Enable="off";
        app.IntegracionButton.Enable="off";
        app.EscalamientodeamplitudButton.Enable="off";
    else
        app.SumaButton.Enable="on";
        app.MultiplicacinButton.Enable="on";
        app.DiferenciacinButton.Enable="on";
        app.IntegracionButton.Enable="on";
        app.Escalamientoeatiempobutton.Enable="on";
    end
```

Captura de pantalla 4. Función variable dependiente.

```
% Value changed function: VIndependienteCheckBox
function VIndependienteCheckBoxValueChanged(app, event)
    value = app.VIndependienteCheckBox.Value;
    if value==0
        app.ReflexinButton.Enable="off";
        app.CorrimientoeneltiempoButton.Enable="off";
        app.Escalamientoeatiempobutton.Enable="off";
    else
        app.ReflexinButton.Enable="on";
        app.CorrimientoeneltiempoButton.Enable="on";
        app.Escalamientoeatiempobutton.Enable="on";
    end
```

Captura de pantalla 5. Función variable independiente.

Enseguida, se observan las señales que se usaron como base y parte primordial de esta GUI de operaciones:

```
case 'Escalon Unitario'
    app.x = (n*app.r)+app.d>=0;
    titulo="Escalon Unitario";

case 'Rampa Unitaria'
    escalon = (n*app.r)+app.d>=0;
    app.x = ((n*app.r)+app.d)."escalon";
    titulo="Rampa Unitaria";

case 'Exponencial'
    app.x=(app.a).^((n*app.d)*app.r);
    titulo="Exponencial";

    app.xint=int((app.r)*(app.a).^nt,nt);%Calcula la integral con la variable s
    app.xint= double(subs(app.xint,nt,n)); %Sustituye la variable simbolica con

case 'Exponencial Compleja'
    e=(cos(app.f*pi*(app.r)*(n*app.d)))+(1i*sin((app.f)*pi*(app.r)*(n*app.d)));
    app.x=(e.^n);

    app.xint=int(cos(app.f*pi*(app.r)*nt)+(1i*sin((app.f)*pi*(app.r)*nt),nt);%
    app.xint= double(subs(app.xint,nt,n)); %Sustituye la variable simbolica con
```

Captura de pantalla 6. Casos de las operaciones.

```
case 'Senoidal'
    app.x=(app.a)*sin(app.f*2*pi*n+app.d*app.r);
    titulo="Senoidal";

    app.xint=int((app.a)*sin(app.f*2*pi*(app.r)*(nt+app.d)),nt); %Calcula la
    app.xint= double(subs(app.xint,nt,n)); %Sustituye la variable simbolica con

case 'Onda Cuadrada'
    app.x=(app.a)*square(app.f*2*pi*(app.r*n)+app.d);
    titulo="Onda Cuadrada";

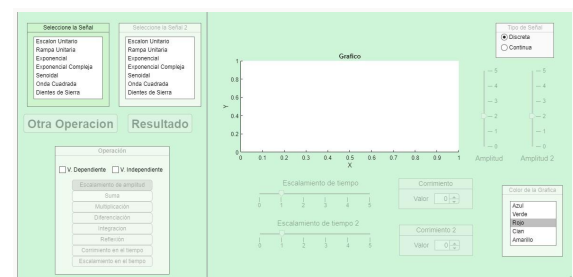
case 'Dientes de Sierra'
    app.x = (app.a)*sawtooth(app.f*2*pi*(app.r*n)+app.d,0.5); %0.5 para gen
    titulo="Dientes de Sierra";

otherwise
```

Captura de pantalla 7. Casos de las operaciones.

III. DISCUSIÓN Y RESULTADOS

La App GUI se ve así inicialmente.



Captura de pantalla 8. App GUI inicializada.

Tabla 1. Suma de señales.

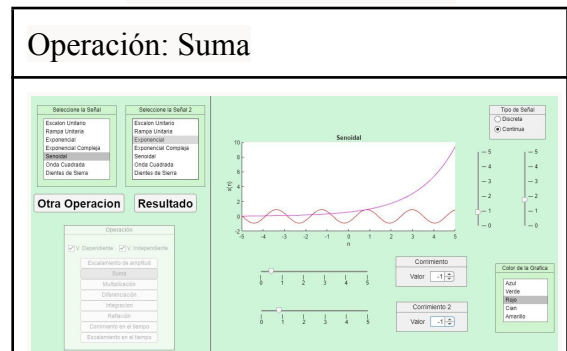


Figura 1. Captura de pantalla. Suma de señal senoidal y exponencial.

Descripción: El resultado que visualizamos fue obtenido a partir de seleccionar primero una señal la operación suma, la segunda señal y un corrimiento en cada una.

Tabla 2. Suma de señales.

Operación: Suma

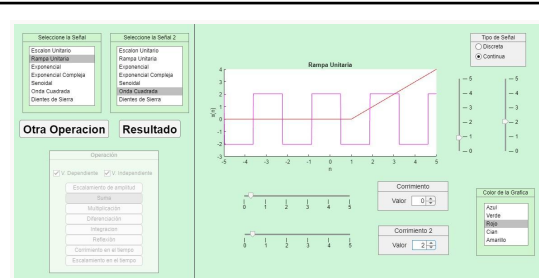


Figura 2. Captura de pantalla. Suma de señal de rampa unitaria y de onda cuadrada.

Descripción: El resultado que visualizamos fue obtenido a partir de seleccionar primero una señal la operación suma, la segunda señal y un corrimiento en la segunda señal.

Tabla 3. Multiplicación de señales.

Operación: Multiplicación

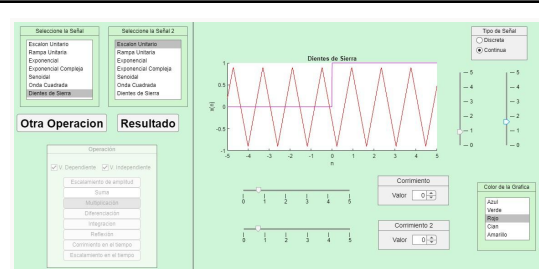


Figura 3. Captura de pantalla. Multiplicación entre señal de dientes de sierra y de escalón unitario.

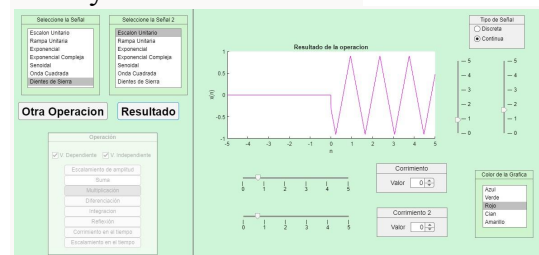


Figura 4. Captura de pantalla. Multiplicación entre señal de dientes de sierra y de escalón unitario.

Descripción: El resultado que visualizamos fue obtenido a partir de seleccionar primero una señal la operación multiplicación, la segunda señal y un corrimiento en la primera señal.

Tabla 4. Multiplicación de señales.

Operación: Multiplicación

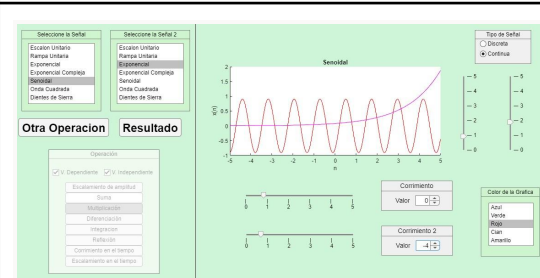


Figura 5. Captura de pantalla. Captura de pantalla. Multiplicación entre señal senoidal y exponencial.

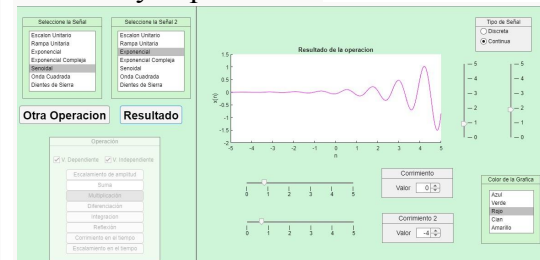


Figura 6. Captura de pantalla. Captura de pantalla. Multiplicación entre señal senoidal y exponencial.

Descripción: El resultado que visualizamos fue obtenido a partir de seleccionar primero una señal la operación multiplicación, la segunda señal y un corrimiento en ambas.

Tabla 5. Diferenciación de señales.

Operación: Diferenciación

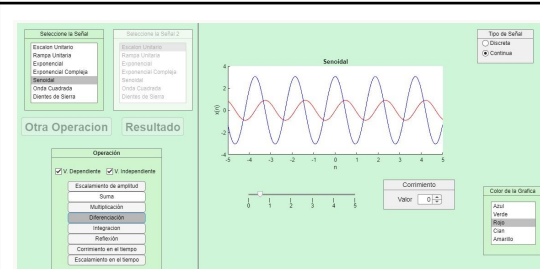


Figura 7. Captura de pantalla. Diferenciación de una señal senoidal.

Descripción: El resultado que visualizamos fue obtenido a partir de elegir una señal y aplicar la operación de diferenciación.

Tabla 6. Diferenciación de señales.

Operación: Diferenciación

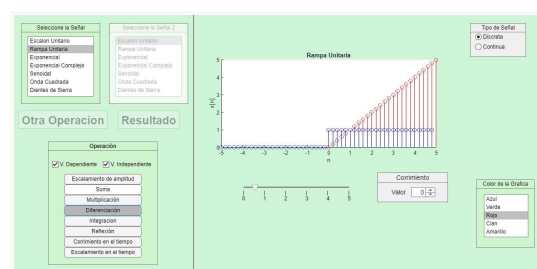


Figura 8. Captura de pantalla. Diferenciación de una señal de rampa unitaria.

Descripción: El resultado que visualizamos fue obtenido a partir de elegir una señal y aplicar la operación de diferenciación.

compleja.

Descripción: El resultado que visualizamos fue obtenido a partir de elegir una señal y aplicar la operación de integración.

Tabla 9. Reflexión de una señal.

Operación: Reflexión

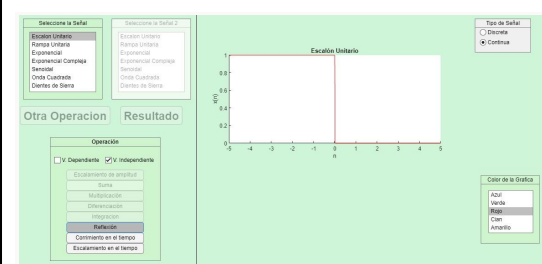


Figura 11. Captura de pantalla. Reflexión de una señal de escalón unitario.

Descripción: El resultado que visualizamos fue obtenido a partir de elegir una señal y aplicar la operación de reflexión.

Tabla 7. Integración de señales.

Operación: Integración

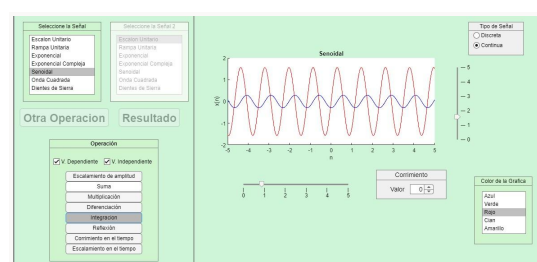


Figura 9. Captura de pantalla. Integración de una señal senoidal.

Descripción: El resultado que visualizamos fue obtenido a partir de elegir una señal y aplicar la operación de integración.

Tabla 10. Escalamiento en el tiempo de una señal.

Operación: Escalamiento en el tiempo

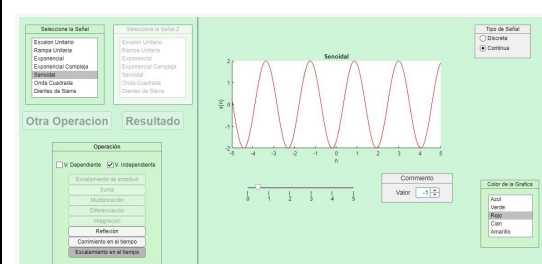


Figura 12. Captura de pantalla. Escalamiento en el tiempo de una señal senoidal.

Descripción: El resultado que visualizamos fue obtenido a partir de elegir una señal y aplicar la operación de escalamiento en el tiempo.

Tabla 8. Integración de señales.

Operación: Integración

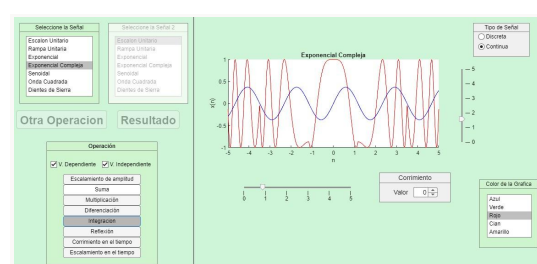
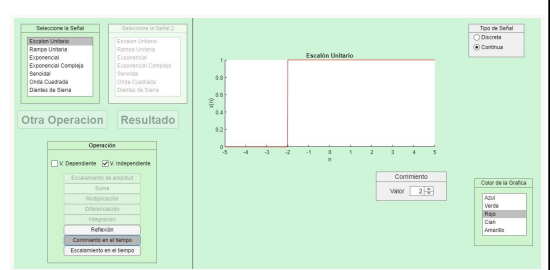


Figura 10. Captura de pantalla. Integración de una señal exponencial

Tabla 11. Corrimiento en el tiempo de una señal.

Operación: Corrimiento en el tiempo

<p>Figura 13. Captura de pantalla. Corrimiento en el tiempo de una señal de escalón unitario.</p>
<p>Descripción: El resultado que visualizamos fue obtenido a partir de elegir una señal y aplicar la operación de corrimiento en el tiempo.</p>

Manual de uso de la GUI:

1. Elegir una señal dando clic sobre ella.
2. Elegir sus parámetros de amplitud, desplazamiento (en caso de ser necesario) y color.
3. Seleccionar una operación de v. dependiente y/o independiente.
4. En caso de ser de variable dependiente, seleccionar la segunda señal dando clic en la misma.
5. (Opcional) Elegir sus parámetros de amplitud, desplazamiento (en caso de ser necesario) y color.
6. Dar clic sobre el botón de resultado para ver la solución.
7. Elegir si será una señal continua o discreta.
8. Para volver a comenzar el procedimiento, luego de haber realizado uno, presionar el botón ‘Otra operación’.

IV. CONCLUSIÓN

Se implementó el funcionamiento de operaciones básicas sobre las señales, por ejemplo, el escalamiento de amplitud, corrimiento en el tiempo, suma, multiplicación, diferenciación e integración. Cada una se puede modificar, en la app GUI, creada en App Designer by Matlab. Es posible modificar la variable independiente, la dependiente, su amplitud, desplazamiento en el tiempo, así como otros elementos como el color y si es continua o discontinua.

Para concluir, esta práctica fue un reto para los integrantes del equipo, puesto que no es común encontrar aplicaciones que funcionen de tal manera con señales, sobre todo en la codificación de la derivación e integración, pues fue necesario instalar herramientas extras en el entorno de App Designer, específicamente Math Toolbox.

Sin embargo, al formar parte de una comunidad que está descubriendo las bondades de la Inteligencia Artificial, hemos podido indagar en App Designer para graficar señales y operarlas, resultó ser una muy buena opción para visualizar una operación entre funciones y transformar las variables independientes. Para comprender las operaciones entre funciones es fundamental poder ver los resultados de una forma tangible. Con ayuda de la App GUI, será mucho más fácil y rápido.

Podría funcionar para generar resultados de señales tomadas de una BD, y transformarlas, es decir, aplicarle alguna de estas operaciones y generar nuevos resultados. Y trabajar con estas nuevas señales.

V. BIBLIOGRAFÍA

Desarrollar apps mediante App Designer - MATLAB & Simulink - MathWorks América Latina. (s. f.). Recuperado 25 de octubre de 2022, de <https://la.mathworks.com/help/matlab/app-designer.html>

GeeksforGeeks. (2022, 15 marzo). Creating Apps Using App Designer in MATLAB. Recuperado 25 de octubre de 2022, de <https://www.geeksforgeeks.org/creating-apps-using-app-designer-in-matlab/>

Object-Oriented Programming. (s. f.). Recuperado 25 de octubre de 2022, de <https://la.mathworks.com/products/matlab/object-oriented-programming.html>

Dr. Vijay Dudhal (2022). EEG ANALYSIS AND CLASSIFICATION (<https://www.mathworks.com/matlabcentral/fileexchange/55112-eeg-analysis-and-classification>), MATLAB Central File Exchange. Recuperado September 7, 2022.

MathWorks, M. (2020). *Funciones de impulso, escalón y rampa*. MathWorks. <https://la.mathworks.com/help/signal/gs/impulse-step-and-ramp-functions.html>

Integración numérica - MATLAB integral - MathWorks América Latina. (s. f.). Recuperado 6 de noviembre de 2022, de <https://la.mathworks.com/help/matlab/ref/integral.html>