CONTROL DE RUIDO ACUSTICO
FILTRO ADAPTATIVO – LMS

1. Introducción

Dando una explicación personal de lo que es un filtro adaptativo LMS (Least Mean Squares), es una herramienta digital utilizada en procesamiento de señales. Imaginando que tienes una radio que recibe señales, pero estas señales a veces vienen con ruido o interferencias, como un zumbido o estática. Quieres escuchar la música o la voz claramente, sin ese ruido. Aquí es donde entra en juego el filtro LMS.

Este filtro es como un asistente inteligente que escucha la señal y aprende cómo suena el ruido. Luego, de manera automática, ajusta sus parámetros para reducir o eliminar ese ruido. Lo hace comparando constantemente la señal de entrada (con ruido) y la señal deseada (sin ruido). Intenta minimizar la diferencia entre estas dos señales, de ahí el nombre "Least Mean Squares", que significa "mínimos cuadrados medios". Este proceso de ajuste es continuo, por lo que el filtro se adapta y mejora con el tiempo, especialmente útil si el tipo de ruido cambia.

En resumen, es como tener un técnico de sonido muy hábil y automático que está constantemente ajustando los controles para darte la mejor experiencia de escucha posible, eliminando ruidos no deseados.

2. Explicación del problema

En el ámbito del procesamiento de señales, el problema central que aborda el filtro adaptativo LMS es la presencia de ruido y otras interferencias que distorsionan las señales útiles en una variedad de aplicaciones. Este desafío se manifiesta de diferentes maneras, dependiendo del contexto específico:

- **Variabilidad del Ruido:** El ruido no siempre es constante o predecible. Puede cambiar dinámicamente en términos de su intensidad y características. Esto hace que sea difícil usar filtros convencionales, que son estáticos y no se adaptan a cambios en el entorno de la señal.
- **Desconocimiento de Propiedades del Ruido:** En muchas situaciones, las propiedades exactas del ruido no se conocen de antemano. Por lo tanto, se necesita un mecanismo que pueda 'aprender' y 'ajustarse' en tiempo real para combatir eficazmente el ruido.
- **Necesidad de Adaptabilidad:** En entornos donde las señales y el ruido pueden cambiar rápidamente (como en las telecomunicaciones móviles o en el procesamiento de audio en tiempo real), se requiere un sistema que pueda adaptarse con la misma rapidez.

El filtro adaptativo LMS aborda estos problemas mediante su capacidad para autoajustarse basándose en el análisis continuo de la señal de entrada y la referencia. A través de su proceso iterativo, el filtro adapta sus coeficientes para minimizar el error entre la señal de salida del filtro y la señal deseada. Esto lo hace particularmente adecuado para entornos en los que el ruido no se puede caracterizar fácilmente de antemano o donde las condiciones de señal varían con el tiempo.

3. Least Mean Squares (LMS)- Esquema minimización del error cuadrático

El algoritmo Least Mean Squares (LMS) se centra en minimizar el error cuadrático, que es un concepto fundamental en el procesamiento de señales y en muchos otros campos que utilizan técnicas de optimización.

Concepto Básico

El "error cuadrático" se refiere al cuadrado de la diferencia entre dos valores: en el caso del filtro LMS, estos valores son la señal de salida deseada y la señal de salida actual del filtro. Al cuadrar esta diferencia, el LMS se concentra en minimizar no solo el error en sí, sino también el impacto de los errores grandes, ya que se amplifican más que los errores pequeños debido al cuadrado.

Funcionamiento del LMS

Inicialización: El filtro LMS comienza con una serie de coeficientes (o pesos) iniciales, generalmente establecidos en cero o un pequeño valor aleatorio.

Cálculo del Error: Para cada muestra de entrada, el filtro calcula el error, que es la diferencia entre la señal de salida deseada y la salida actual del filtro.

Minimización del Error Cuadrático: El LMS ajusta sus coeficientes para minimizar el promedio del cuadrado de este error a lo largo del tiempo. Esto se hace modificando los coeficientes en la dirección que reduce este error cuadrático medio.

Actualización de Coeficientes: La actualización de los coeficientes se realiza mediante una fórmula que incluye el error calculado y la tasa de aprendizaje del filtro, que determina cuánto se modifican los coeficientes en cada paso. La tasa de aprendizaje es un parámetro crucial que equilibra la velocidad de adaptación con la estabilidad del filtro.

Ventajas de Minimizar el Error Cuadrático

Robustez: Al centrarse en el error cuadrático, el filtro es más robusto a variaciones y ruido, especialmente en situaciones donde el ruido tiene características desconocidas o cambiantes.

Convergencia Estable: El método de minimización del error cuadrático asegura una convergencia estable hacia la solución óptima, siempre que la tasa de aprendizaje esté correctamente configurada.

Aplicaciones

Esta estrategia de minimización del error cuadrático es aplicable en numerosas áreas, como la cancelación de eco en telecomunicaciones, el filtrado de ruido en señales de audio, y en sistemas de control y predicción en ingeniería y finanzas.

4. ANALISIS DEL ALGORITMO LMS- PROBAR CON UN AUDIO PROPIO Y DIFERENTES COEFICIENTES Y CONCLUSIONES.

Se tuvieron algunos inconvenientes por la versión de Matlab, ya que en esta version no existe la función adptfilt.lms, la cual tuvimos que reemplazar como se ve en la línea 76 :
“ha = dsp.LMSFilter('Length', 512, 'StepSize', 0.001);”

A continuación una explicación del funcionamiento del algoritmo.

Carga de Señales de Audio: El script comienza cargando tres señales de audio: la señal de voz sin ruido (voz_sola.wav), la señal de ruido (ruido_solo.wav), y una señal contaminada con ruido (conta.wav).

Definición de Señales y Parámetros del Filtro:

x: Señal de referencia (ruido).

d: Señal deseada (voz con ruido).

Ncoef: Número de coeficientes del filtro adaptativo.

W: Coeficientes del filtro, inicializados a cero.

Mu: Tasa de aprendizaje o paso del algoritmo.

Algoritmo LMS:

El algoritmo procesa las señales muestra por muestra.

En cada iteración, calcula la salida del filtro y como el producto punto entre los coeficientes del filtro W y un vector X que contiene las últimas Ncoef muestras de la señal de referencia.

Calcula el error e como la diferencia entre la señal deseada d y la salida del filtro y.

Actualiza los coeficientes del filtro W basándose en el error y la tasa de aprendizaje Mu.

Calcula el error cuadrático medio (MSE) para cada muestra.

Guardado y Reproducción de Sonidos: El script guarda y reproduce la señal filtrada, que idealmente debería tener menos ruido.

Cálculo de FFT (Fast Fourier Transform): Se calcula la transformada de Fourier para analizar el espectro de frecuencias de las señales.

Comparación con Implementación de MATLAB: Se utiliza una implementación del filtro LMS proporcionada por MATLAB (dsp.LMSFilter) para comparar con la implementación personalizada.

Visualización de Resultados: Se generan gráficos para visualizar el error, la convergencia del MSE, y los espectros de frecuencia de las señales.

En resumen, el algoritmo LMS ajusta los coeficientes de un filtro para minimizar el error entre la salida del filtro y una señal deseada. En tu caso, se utiliza para reducir el ruido de una señal de voz. La clave del LMS está en cómo ajusta los coeficientes del filtro (W) iterativamente para reducir el error cuadrático medio.

Pruebas con audio propio (Audio extraído de un video de youtube)

Se creó un algoritmo que genere un ruido al audio adjuntado, almacene el audio original, el audio con el ruido, y el audio solo del ruido generado para poder hacer las pruebas.

```
% Carga el archivo de audio original
[originalAudio, Fs] = audioread('test.wav');

% Determinar si la señal es mono o estéreo
[numMuestras, numCanales] = size(originalAudio);

% Generar ruido aleatorio
% El ruido se genera con una amplitud que es una fracción del máximo
% valor en la señal original, ajusta este valor según sea necesario
amplitudRuido = 0.1 * max(abs(originalAudio(:)));
ruido = amplitudRuido * randn(numMuestras, numCanales);

% Agrega ruido a la señal original
audioConRuido = originalAudio + ruido;

% Guarda el audio con ruido en un nuevo archivo
audiowrite('C:\Users\RafaG\Desktop\DSP (Yessica)\TP Final\lms\creacion_audio\audioConRuido.wav', audioConRuido, Fs);

% Guarda solo el ruido en otro archivo
audiowrite('C:\Users\RafaG\Desktop\DSP (Yessica)\TP Final\lms\creacion_audio\soloRuido.wav', ruido, Fs);

sound(audioConRuido, Fs);
```

Una vez generados estos audios se procedió a utilizarlos en el algoritmo proveído.

```
%--------------------------------------------------------------------------

[senal,FS]= audioread('test.wav');
[ruido,FS]= audioread('soloRuido.wav');
[senal_ruido,FS]= audioread('audioConRuido.wav');
x = ruido(1:size(ruido),1); % Señal de Referencia
d = senal_ruido; % Señal deseada
```

Al ajustar la tasa de aprendizaje (Mu), asignándole un valor demasiado baja como 0.001 se observa que la convergencia es mas lenta, y también asignándole un valor demasiado alto puede hacer que el audio se vuelva inestable, al final se optó por utilizar 0.003 ya que con este valor la convergencia es lo suficientemente rápida.

Al ajustar el numero de coeficiente del filtro (Ncoef) se observó que aumenta la complejidad computacional del algoritmo, pero al subir el valor no se observa un cambio en la reducción del ruido.

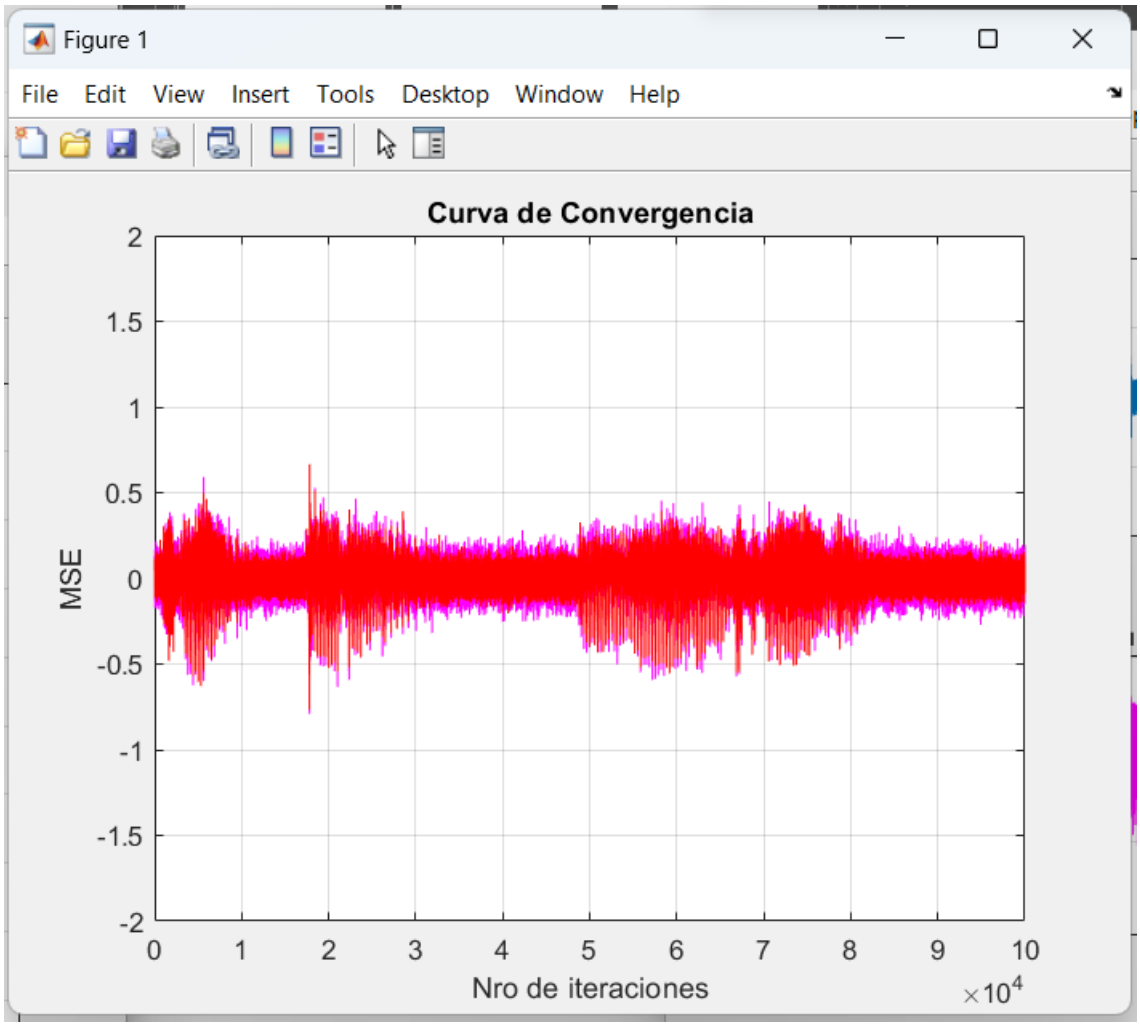
Figure 1
File Edit View Insert Tools Desktop Window Help
Curva de Convergencia
MSE
Nro de iteraciones
×10^4
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
0 1 2 3 4 5 6 7 8 9 10

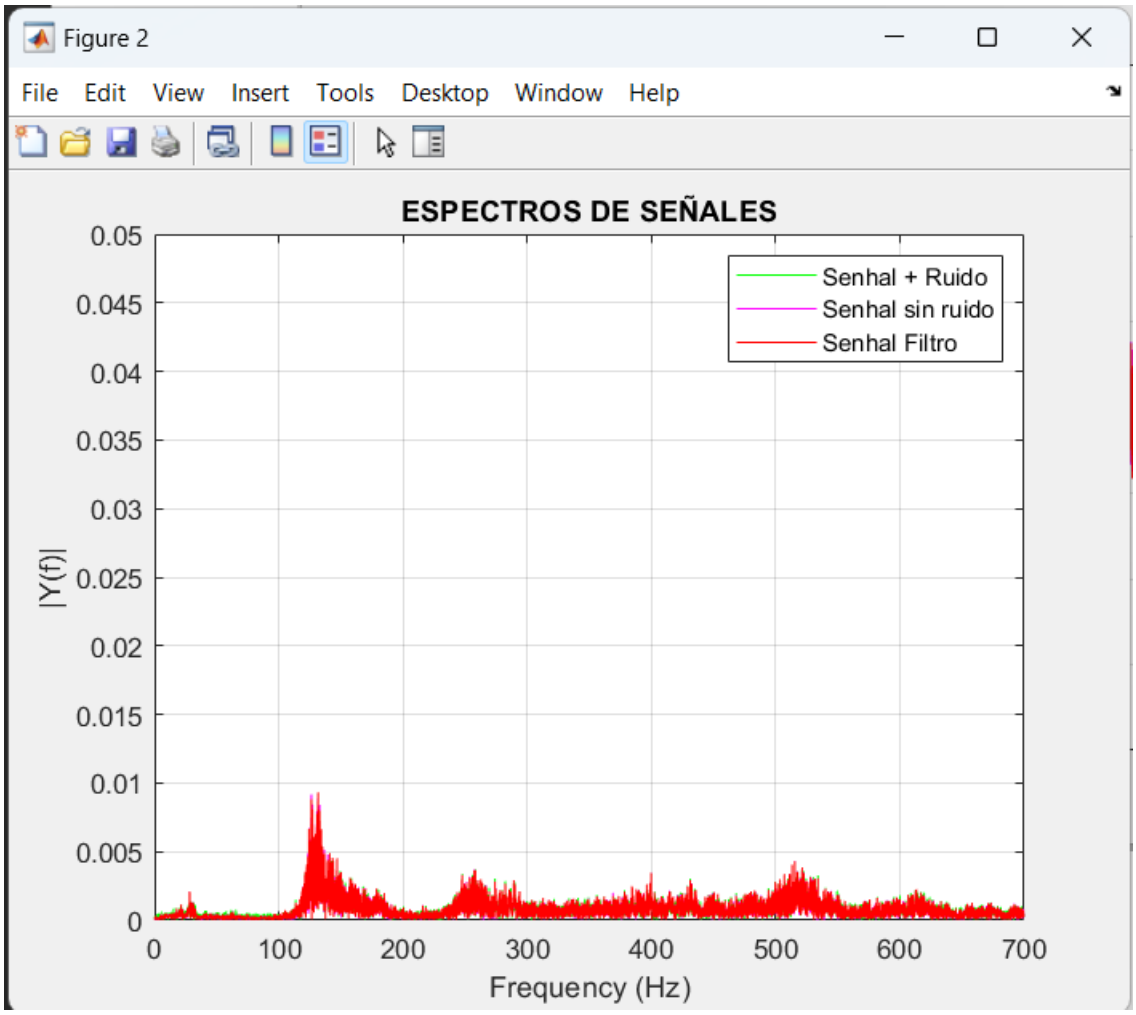
Figure 2
File Edit View Insert Tools Desktop Window Help
ESPECTROS DE SEÑALES
Senhal + Ruido
Senhal sin ruido
Senhal Filtro
|Y(f)|
0.05
0.045
0.04
0.035
0.03
0.025
0.02
0.015
0.01
0.005
0
0 100 200 300 400 500 600 700
Frequency (Hz)

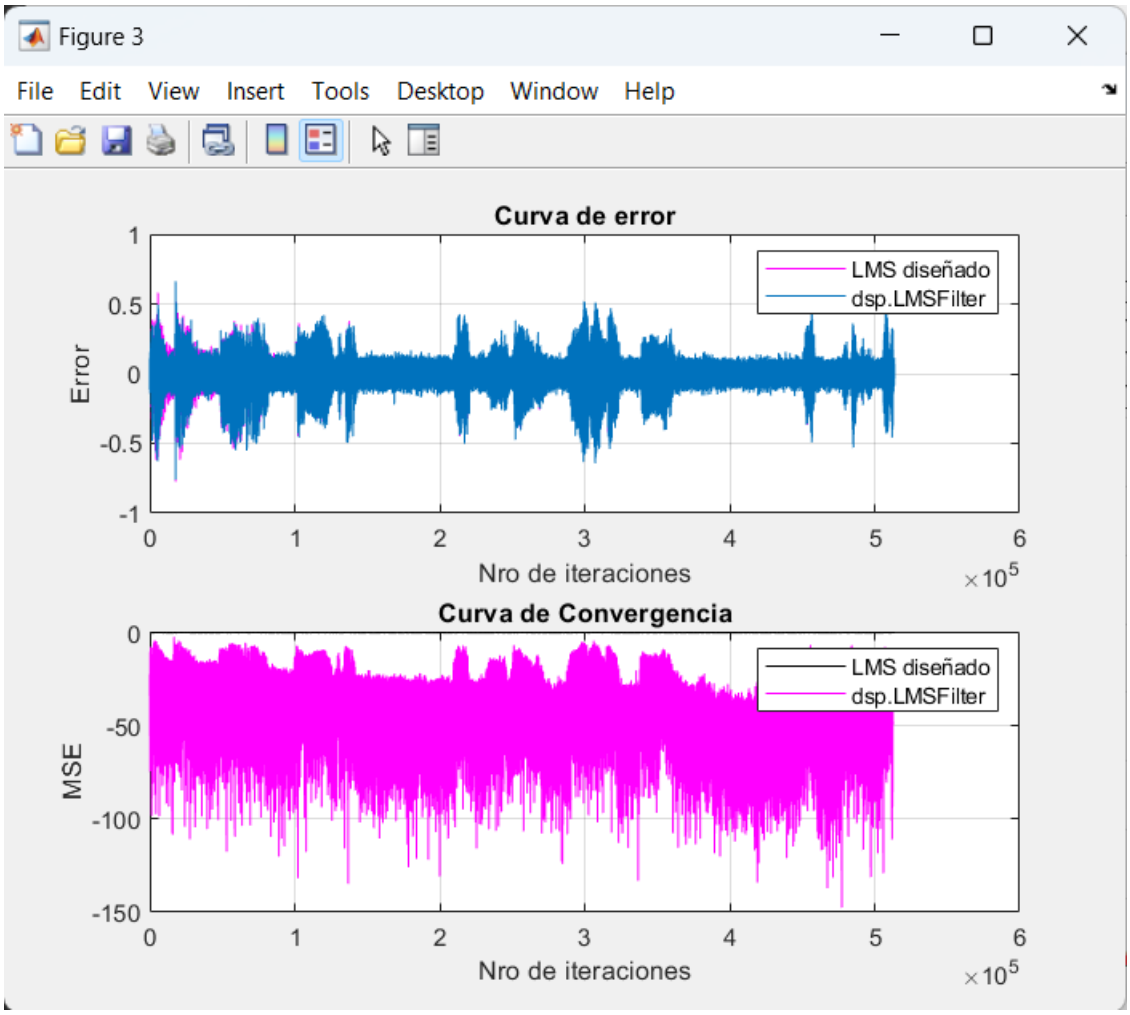
Figure 3
File Edit View Insert Tools Desktop Window Help
Curva de error
LMS diseñado
dsp.LMSFilter
Error
Nro de iteraciones
×10^5
Curva de Convergencia
LMS diseñado
dsp.LMSFilter
MSE
Nro de iteraciones
×10^5