Práctica 2. Modelado de la señal de voz como un proceso aleatorio y su aplicación en la transmisión sobre un canal con pérdidas (modelo 2)

Dada una señal de voz digitalizada, con una frecuencia de muestreo de 8kHz representada por una profundidad 8 bits, esta se transmite sobre un canal de transmisión con pérdidas.

El objetivo de la práctica es modelar la señal de forma que podamos estimar el valor de las muestras de voz que no han llegado al receptor, y hacer uso de la estimación para reconstruir una señal.

Nos dan 8 señales sobre las que trabajar.

```
frequency = 8000; % 8kHz
quasicero = 1e-15;
m=[-128:127]
```

```
m = 1×256
  -128  -127  -126  -125  -124  -123  -122  -121  -120  -119  -118  -117  -116 ···
```

Añado el parámetro quasicero, un valor muy pequeño con el que voy a sustuir las probabilidades cero.

MODELO DOS

DESCRIPCIÓN

Este modelo será mejor que el anterior, pues consideramos la señal de voz como un proceso aleatorio de Markov. Como estamos trabajando con datos discretos, este proceso aleatorio de Markov recibe el nombre de ***Cadena de Markov***.

Vamos a recuperar de la anterior parte nuestra base de datos con todas las muestras, y durante el transcurso de la práctica, nos iremos familiarizando con los conceptos.

```
S = [s1;s2;s3;s4;s5]
```

```
S = 85502×1
     0
     0
     0
     0
     0
     0
     0
     0
     0
     0
     0
     0
     0
     0
    -1
     ⋮
```

Los procesos aleatorios de Markov se sustentan en base a la **propiedad de Markov**. Esta propiedad dice, que para predecir qué pasará después, solo importa dónde estamos ahora. No importa cómo llegamos ahí, ni qué pasó antes.

Los procesos aleatorios de Markov están conformados por dos grandes ideas, los **estados**, que son las diferentes "situaciones" o "condiciones" posibles en las que puede estar nuestro sistema, las **transiciones**, los cambios de un estado a otro. Las probabilidades se las asignamos a las transiciones.

Las cadenas de Markov pueden ser **homogéneas**, si la probabilidad de ir de un estado A a un estado B en un paso no depende del tiempo en el que se encuentra la cadena, o **no homogéneas**, en caso contrario.

Vamos a asumir los siguientes aspectos en esta práctica:

- Que nuestra cadena de Markov es homogénea y,
- que es de primer oren, es decir, que una transición depende exclusivamente del estado actual, y no de la historia anterior.

Con esto en mente, decimos que la probabilidad de transición para una cadena homogénea de primer orden viene denotado por la siguiente expresión:

$$p_{\mathrm{ij}} = P[X_{n+1} = j | X_n = i]$$

y con las probablidades de transición en un paso, variando los índices i y j sobre el espacio de estados posibles (los 256 valores que puede tomar cada muestra), obtenemos la matriz de probabilidades de transición:

```
TransitionMatrix = zeros([256,256]);

for i = 1:length(S) - 1
    TransitionMatrix(S(i) + 129, S(i + 1) + 129) = (TransitionMatrix(S(i) +
129, S(i + 1) + 129) + 1);
    % Esta línea cuenta cuántas veces ocurre cada transición entre los
valores de la señal.
    % S(i) me da el valor de la muestra. S(i + 1) me da el valor de la
    % muestra siguiente.
end

TransitionMatrix(TransitionMatrix==0) = quasicero
```

```
TransitionMatrix = 256×256
10^4 ×
    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000 ···
    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000
    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000
    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000
    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000
    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000
    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000
    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000
    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000
    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000
    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000
    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000
    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000
```

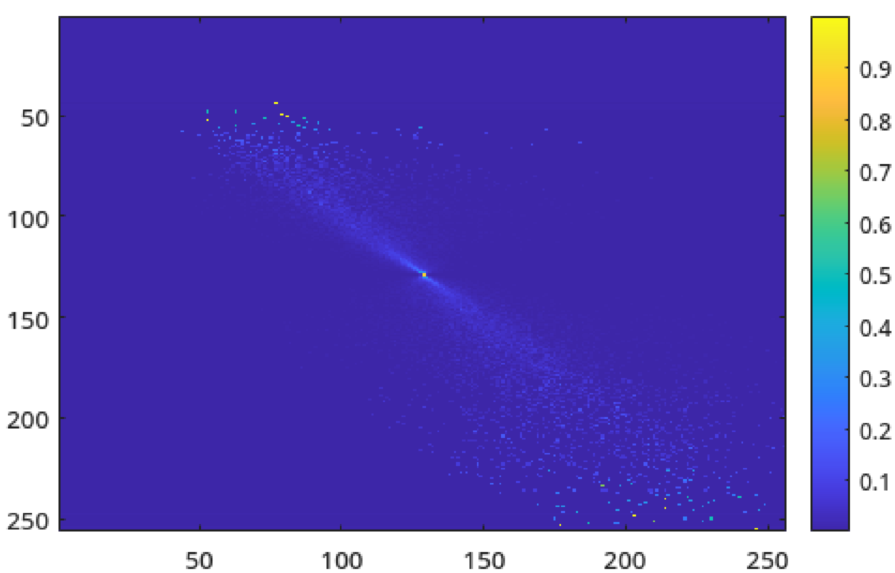
```
    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000
    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000
      ⋮
```

```
TransitionProbabilityMatrix = TransitionMatrix./sum(TransitionMatrix, 2) % Pongo 2 para que la suma sea sobre cada fila y no columna.
```

```
TransitionProbabilityMatrix = 256×256
    0.0039    0.0039    0.0039    0.0039    0.0039    0.0039    0.0039    0.0039 ⋯
    0.0039    0.0039    0.0039    0.0039    0.0039    0.0039    0.0039    0.0039
    0.0039    0.0039    0.0039    0.0039    0.0039    0.0039    0.0039    0.0039
    0.0039    0.0039    0.0039    0.0039    0.0039    0.0039    0.0039    0.0039
    0.0039    0.0039    0.0039    0.0039    0.0039    0.0039    0.0039    0.0039
    0.0039    0.0039    0.0039    0.0039    0.0039    0.0039    0.0039    0.0039
    0.0039    0.0039    0.0039    0.0039    0.0039    0.0039    0.0039    0.0039
    0.0039    0.0039    0.0039    0.0039    0.0039    0.0039    0.0039    0.0039
    0.0039    0.0039    0.0039    0.0039    0.0039    0.0039    0.0039    0.0039
    0.0039    0.0039    0.0039    0.0039    0.0039    0.0039    0.0039    0.0039
    0.0039    0.0039    0.0039    0.0039    0.0039    0.0039    0.0039    0.0039
    0.0039    0.0039    0.0039    0.0039    0.0039    0.0039    0.0039    0.0039
    0.0039    0.0039    0.0039    0.0039    0.0039    0.0039    0.0039    0.0039
    0.0039    0.0039    0.0039    0.0039    0.0039    0.0039    0.0039    0.0039
    0.0039    0.0039    0.0039    0.0039    0.0039    0.0039    0.0039    0.0039
      ⋮
```

Visualmente, nuestra matriz de probabilidad de transición queda como:

```
figure;
imagesc(TransitionProbabilityMatrix)
colorbar
```



RECONSTRUCCIÓN

Ya hemos descrito nuestra muestra. Sabemos que nuestra señales afrontarán medios sin pérdidas. Ahora tenemos que abordar cómo actuaremos en los momentos en los que perdemos una muestra.

Para determinar el valor de las muestras perdidas, recurriremos a dos estimaciones:

LA MODA

En el modelo anterior, la moda es solo un número (el valor más repetido globalmente). Pero, en este nuevo modelo, al trabajar con cadenas de Markov, la moda es un vector que relaciona el valor de una muestra anterior ($X[i-1]=m_k$) con el valor ($X[i]=m_j$) que maximiza la probabilidad condicional $P[X[i]=m_j|X[i-1]=m_k]$.

```
trendProbabilityValues = zeros(length(m),1)
```

```
trendProbabilityValues = 256×1
     0
     0
     0
     0
     0
     0
     0
     0
     0
     0
     0
     0
     0
     0
     0
     ⋮
```

```
for k = 1:length(m)
    [~, trendProbabilityPosition] = max(TransitionProbabilityMatrix(k, :));
    trendProbabilityValues(k) = m(trendProbabilityPosition);
end
```

PROBABILIDAD PONDERADA

Aquí sucede lo mismo, necesitamos un vector que haga la misma relación que en la estimación anterior, pero la muestra que estimo viene dada por una probabilidad ponderada:

$$\widehat{x_2}=\sum_{j=1}^{256} m_j P[X[i]=m_j|X[i-1]=m_k]$$

Implemento la fórmula de la sumatoria a través de un bucle **for**:

```
weightProbabilityValues = zeros(length(m),1)
```

```
weightProbabilityValues = 256×1
     0
     0
     0
     0
     0
     0
```

```
     0
     0
     0
     0
     0
     0
     0
     0
     0
     ⋮
```

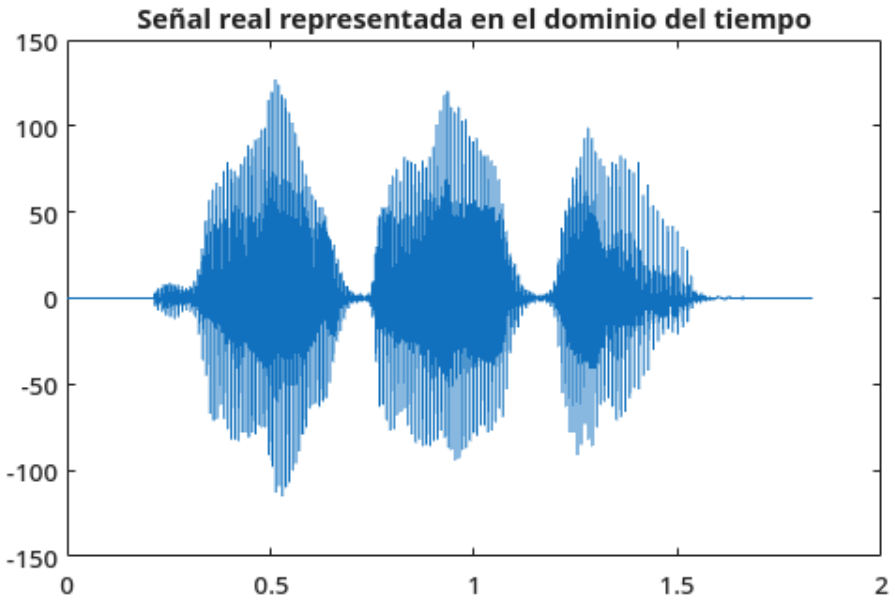
```
for k = 1:length(m)
    weightProbabilityValues(k) = sum(m.* TransitionProbabilityMatrix(k, :));
end
```

Nota: *El operador* ***.**** *le dice a MATLAB que multiplique cada posible valor que puede tomar cada muestre por su respectiva probabilidad.*

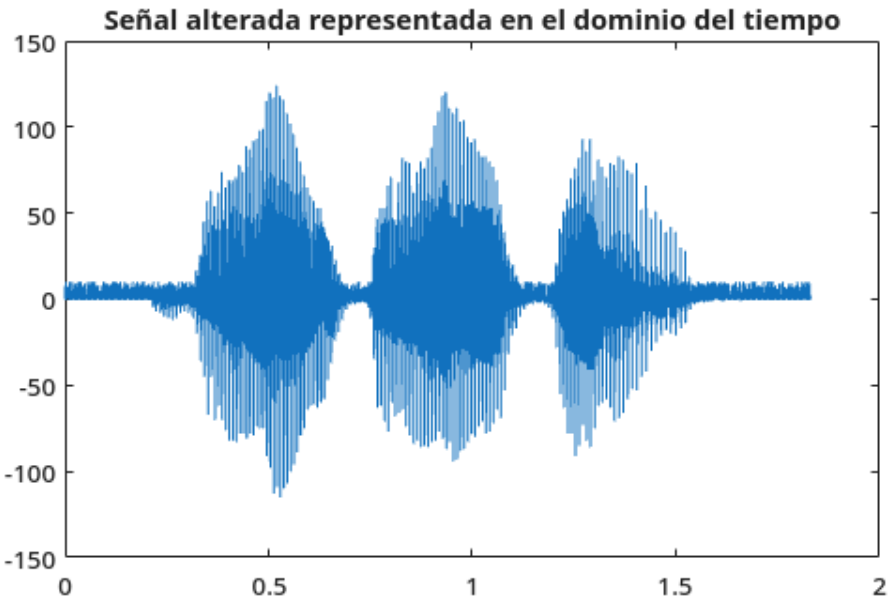
EVALUACIÓN

Recuperamos de la anterior parte nuestras variables *realSignal* y *lossySignal* (ahorramos así recursos de cómputo):

```
figure;
plot([1:length(realSignal2)]/frequency, realSignal2)
title("Señal real representada en el dominio del tiempo")
```




```
figure;
plot([1:length(lossySignal2)]/frequency, lossySignal2)
title("Señal alterada representada en el dominio del tiempo")
```



Como sabemos que perdemos información cada 10 muestras, significa que cada 10 muestras tendremos que aplicar nuestros métodos de estimación. Al tener dos métodos, haremos dos reconstrucciones y las compararemos usando la SNR:

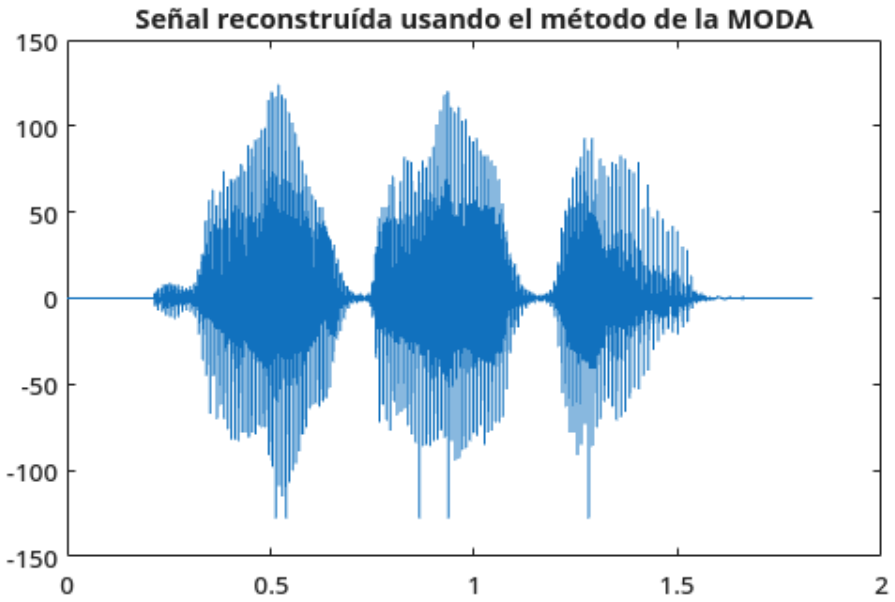
```
trend_reconstructedSignal_Markov = lossySignal;
weightProbability_reconstructedSignal_Markov = lossySignal;

for row_index = 1:10:length(lossySignal2)
    if (row_index ~= 1)
        trend_reconstructedSignal_Markov(row_index, 1) =
trendProbabilityValues(trend_reconstructedSignal_Markov(row_index - 1) +
129);
        weightProbability_reconstructedSignal_Markov(row_index, 1) =
weightProbabilityValues(weightProbability_reconstructedSignal_Markov(row_inde
x - 1) + 129);
        % Hacemos "row_index - 1" para recuperar el valor anterior de la
        % muestra.
    else
        % Hago este else para asginar un valor a la muestra número 1. Cómo
        % antes de este no hay ninguna muestra, asumo que es 0.
        trend_reconstructedSignal_Markov(row_index, 1) =
trendProbabilityValues(0 + 129);
        weightProbability_reconstructedSignal_Markov(row_index, 1) =
weightProbabilityValues(0 + 129);
    end
end
```

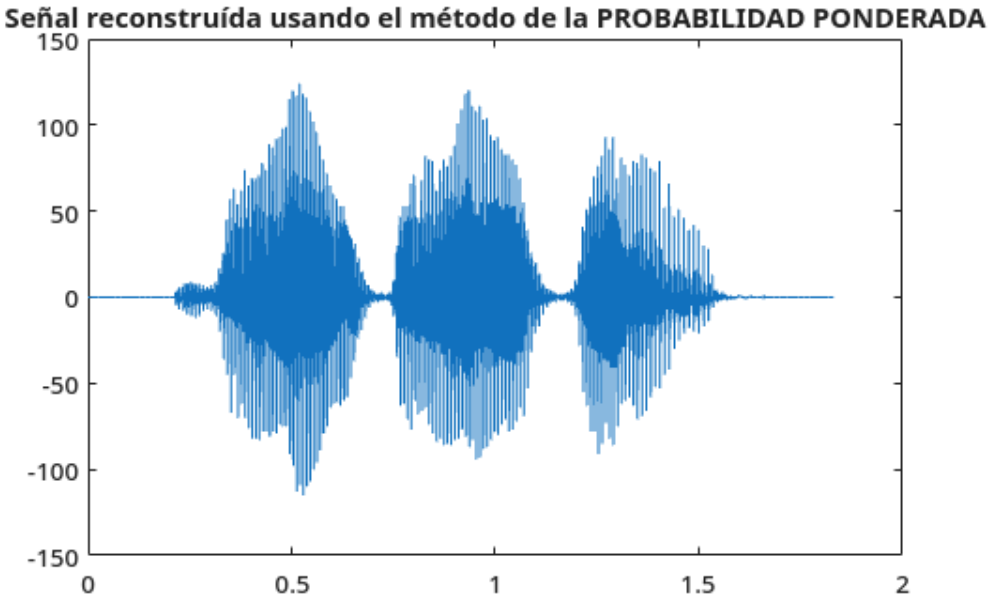
Representando gráficamente, veamos como quedan nuestras señales reconstruídas:

```
figure;
```

```
plot([1:length(trend_reconstructedSignal_Markov)]/frequency,
trend_reconstructedSignal_Markov)
title("Señal reconstruída usando el método de la MODA")
```




```
figure;
plot([1:length(weightProbability_reconstructedSignal_Markov)]/frequency,
weightProbability_reconstructedSignal_Markov)
title("Señal reconstruída usando el método de la PROBABILIDAD PONDERADA")
```



Para escucharlo, usaríamos la función **sound()**.

Ahora, vamos a calcular la SNR de ambas señales. La SNR viene dada por la siguiente expresión:

$$\mathrm{SNR}_{k(}(\mathrm{dB}) = 10\log_{10}\frac{\sum_{n=1}^{N} x[n]^{2}}{\sum_{n=1}^{N}(x[n]-r_k[n])^2}$$

donde $x[n]$ es la señal enviada y $r_k[n]$correspondería a las reconstrucciones, $k=1$ siendo la reconstruída con el método de la moda y $k=2$, con la probabilidad ponderada. Ncorresponde a la suma total total de muestras.

```
SNR1 = 10*log10(((realSignal'*realSignal))/
((realSignal-trend_reconstructedSignal_Markov)'*(realSignal-
trend_reconstructedSignal_Markov)))
```

```
SNR1 =
15.4107
```

```
SNR2 = 10*log10(((realSignal'*realSignal))/
((realSignal-weightProbability_reconstructedSignal_Markov)'*(realSignal-
weightProbability_reconstructedSignal_Markov)))
```

```
SNR2 =
15.8525
```

Obtenemos dos SNR positivas. ¿Qué implica? Pues que la señal es aproximadamente 30 veces más fuerte que el ruido. Además observamos que, la estimación usando el método de la probabilidad ponderada es mejor.

```
% sound(trend_reconstructedSignal_Markov)
% sound(weightProbability_reconstructedSignal_Markov)
% Redactado por Alexandru Theodor Muntenaş.
```