Decifrando Código Enigma

Camilo Acosta Acosta
Instagram/GitHub: @acosmilo
Esteba Albuja

Julio 2019

1. Objetivo

Crear un programa que decifre los mensajes del enemigo.

2. Solución

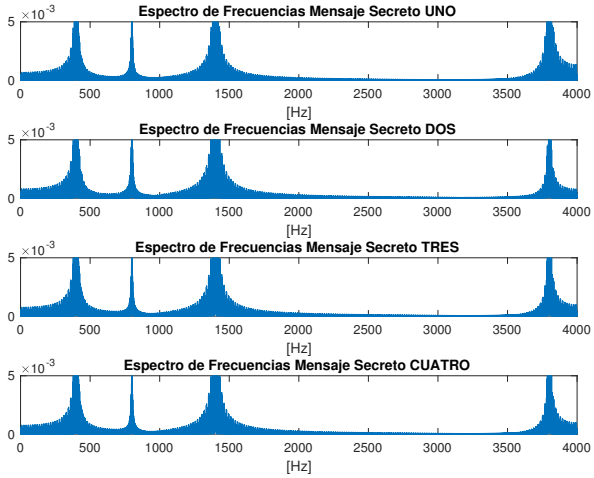
Primero se identificó en cada mensaje las frecuencias que comprenden cada uno de ellos, sabiendo que deben ser 4, una para punto: raya, espacio entre letras y espacio entre palabras (la codificación usada en ENIGMA es similar al código morse). Para esto se usó la función *fft()* de Matlab para encontrar la transformada de Fourier de cada audio, se obtuvo que los 4 mensajes estaban compuestos por las 4 frecuencias mostradas a continuación: 400Hz, 800Hz, 1400Hz, 3800Hz.

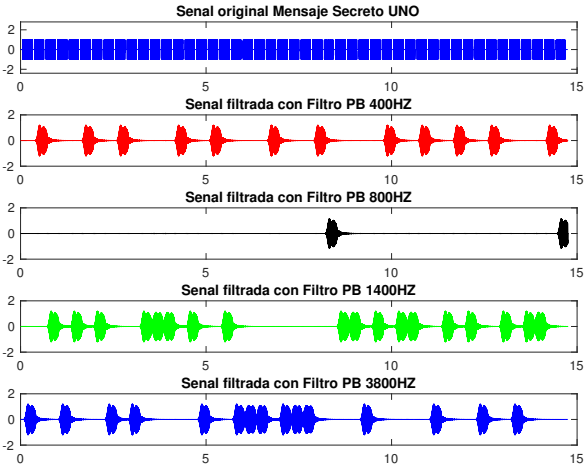
A continuación, conociendo las frecuencias de cada mensaje se crearon 4 filtros pasa banda de cada uno de los 4 tipos (Butterworth, Chebyshev I, ChebyshevII y Elíptico), así se determinó que se usaría la aproximación Butterworth, se escogió Butterworth porque como se verá más adelante se suman los valores de las señales filtradas y al filtrar se cola ruido por delante del pulso de la señal, y si sumamos los valores correspondientes al pulso con ruido puede arrojar un falso positivo en nuestra sentencia comparadora si da como resultado un valor mayor al indicado, en otras palabras el filtro Butterworth hace que el pulso de adelante tenga menos ruido y menos valores por lo que al sumarse el resultado será muy bajo y pasará por la sentencia comparadora con menos inconvenientes. Así se pudo filtrar cada mensaje obteniendo 4 gráficas que nos mostraban las frecuencias correspondientes, para así visualizar que partes del mensaje comprenden los puntos, las rayas y los espacios entre letras y palabras, comparándolas a su vez con los mensajes que obtuvo el espía caído.

- La gráfica del audio que deja pasar las frecuencias de 800Hz(negra) al ser la que más información filtró del mensaje original se puede deducir que es la que representa los espacios entre palabras.
- La gráfica del audio que deja pasar las frecuencias de 400Hz(roja) al ser la segunda que más información filtró del mensaje original se puede deducir que es la que representa los espacios entre letras.

- Dejando así las frecuencias de 1400Hz(verde) y 3800Hz(azul) como las posibles frecuencias que representen puntos o rayas. Para saber cuál frecuencia correspondiente a punto y cual a raya se usó el mensaje uno (SM1.wav) el cual dice: "TRABAJO FINAL", entonces el primer carácter (en este caso T) que en morse es representado por una raya, sabiendo así que la frecuencia de rayas es la de 3800Hz y la de puntos es la de 1400 Hz, según la gráfica.

Hasta esta parte desciframos cuales son las 4 frecuencias utilizadas y que símbolos representan en el código de ENIGMA cada una de ellas.





Lo que se busca es que entre un mensaje de audio y el programa devuelva el mensaje secreto en letras y números despues de ser decodificado. Para esto es necesario que las 4 señales ya filtradas de alguna manera sean interpretadas por Matlab . Entonces aplicamos la siguiente idea:

- Primero se obtiene un valor que corresponde al dividir el tamaño del vector para el

número de pulsaciones que contiene esta señal , asi se obtendra el tamaño de cada pulsacion

- Con el número obtenido anteriormente y con ayuda de un *for()* se puede recorrer la longitud de cualquiera de las señales filtradas de pulso en pulso.

- En cada repetión del ciclo *for()* se hace la sumatoria del valor absoluto de los valores correspondientes al vector de la señal en ese pulso. Y el resultado de la sumatoria se lo asigna a la posición de un nuevo vector correspondiente al ciclo en el que se encuentre el *for()*.

- Debido a que la señal ya esta filtrada habrá pulsos en los que el resultado de la sumatoria serán muy bajos (casi 0) ya que esta parte de la señal de audio no contiene valores y en el caso de contenerlos vendrían a ser lo que conocemos como ruido. Entonces el vector que se formó con cada ciclo *for()* contendrá valores muy pequeños (cercanos al cero) o muy altos (mayores a mil), por lo que se puede crear un vector del mismo tamaño que el anterior donde tenga el valor de 1 en la misma posición en donde el vector anterior tenga un valor mayor a mil, caso contrario se le asignará el valor de 0 .

- El tamaño del vector conformado por 1 y 0 será igual al número total de pulsaciones de la señal original. El proceso anterior se repite para las 4 señales anteriormente filtradas obteniendo 4 vectores conformados por ceros y unos, para poder diferenciarlos a cada vector se lo multiplica por un escalar diferente (para la señal de 400Hz se lo multiplicó por 1, a la señal de 800Hz se la multiplicó por 2, a la señal de 1400Hz se la multiplicó por 3 y a la señal de 3800Hz se la multiplicó por 4). Despues al sumar los 4 vectores se genera un vector con valores de 1,2,3 y 4 en cada posición del vector donde 1 representa al espacio entre letras, 2 representa el espacio entre palabras, 3 reprententa a un punto y 4 representa una raya tal como la codificación de ENIGMA.

- A continuación el vector anterior entra a una función llamada *interP()* donde esta crea un nuevo vector de tamaño 5 (5 porque en código Morse máximo se tiene una combinación de 5 símbolos ya sean rayas o puntos) que inicialmente estará lleno de ceros pero pasara a contener los valores de 3 o 4 que se encuentren antes de un 1. El vector que entra a la función pasa por el ciclo *for()* verifica las posiciones en las que se encuentra en cada instancia, de encontrarse con un 1 o un 2 terminará de escribir el vector que contiene 3 y 4; de ser un 2 escribirá un espacio en una nueva variable llamada **palabra** (la cual mostrará el mensaje final).

- Posteriormente se usa la función *interL()* donde vector conformado por 3 y 4 por medio de un *swich()* selecciona el caracter (letra o número) según la codificación de ENIGMA (Morse). Como la variable seleccionadora de *swich()* debe ser un valor escalar y no un vector como es el caso se crea otra función *baseT()* donde se le dara un peso al vector es decir se lo convierte en un escalar de valor único y que asi se pueda usar en *swich()*.

- Lo que hace *baseT()* es convertir el vector en un valor único asignandole pesos. Para que el resultado de la operación de pesos no se repita entre vectores diferentes (y nos de una letra equivocada) se optó por la siguiente idea, a todos los 3 y 4 del vector

se los convierte en 1 y 2 respectivamente por medio de una resta, entonces el vector lo conformarían valores de 0, 1 y 2 que en su conjunto es un número de base 3, y si se lo convierte a decimal dará un escalar único para cada posible vector (Se aplicó la fórmula general de cambio de base para sistemas de numeración para pasar a base 10 $\sum_{i=1}^{5} a_i \cdot b^{i-1}$).

- A la letra o número seleccionada por *interL()* irá concatenandose a la variable **palabra** con la función *strcat()* y asi despues de recorrer todos los posibles vectores de 3 y 4 la información dentro de **palabra** sera el mensaje ya decodificado.