Transformada de Fourier

Diego Branco, Gabriel Colina

Julio 2025

1 Análisis del problema

El sistema Multifrecuencial de Marcación de Tonos (o DTMF por sus siglas en inglés), es un sistema de señal por tonos usando la frecuencia de voz en líneas telefónicas.

La señal DTMF es generada por la suma de dos tonos sinusoidales. Uno de ellos es seleccionado de un grupo de señales de baja frecuencia (que llamaremos f_{lo}), y el otro de señales de alta frecuencia (f_{hi}), la cual podemos generar de la siguiente manera:

$$\frac{(sin(2\pi f_{lo}t) + sin(2\pi f_{hi}t))}{2} \tag{1}$$

donde t representa el tiempo.

En total hay 16 combinaciones, las cuales están descritas en la siguiente tabla: Nuestro proyecto consta de un programa que codifica y decodifica estas

Table 1: Frecuencias usadas al formar señales DTMF

Frequencia	**1209 Hz**	**1336 Hz**	**1477 Hz**	**1633 Hz**
697 Hz	1	2	3	A
770 Hz	4	5	6	B
852 Hz	7	8	9	C
941 Hz	*	0	#	D

señales y las traduce a texto. También permite ver la representación gráfica (continua y discreta) de estas señales.

NOTA: En nuestro proyecto no usaremos las teclas A,B,C,D.

El proceso de decodificación es el siguiente: Primero se toma el archivo WAV y se corta en pequeñas ventanas de 0.25 segundos; estas ventanas son factorizadas usando la implementación de NumPy del algoritmo de Transformación rápida de Fourier (FFT según sus siglas en inglés) en un arreglo sparse de números complejos. Luego se compara este arreglo para identificar los picos correspondientes a las frecuencias DTMF.

Debido a la necesidad de probar el algoritmo de decodificación , también se creó un proceso de codificación para exportar archivos WAV, en el cual se

toman las frecuencias DTMF y se aplica la suma de las dos sinusoidales para generar cada tono, el cual es compilado por el módulo wavfile de scipy.io.

En este proyecto, se usa el algoritmo FFT principalmente para interpretar la señal de sonido, la cual ya sabemos es el resultado de interpolar dos tonos de diferentes frecuencias, tal como vimos en (1), sin embargo, al viajar por líneas telefónicas, la señal se puede distorsionar un poco, y FFT también ayuda a realzar las frecuencias más prominentes, que serían las pertenecientes a los tonos altos y bajos de las señales DTMF.

2 Interpolación trigonométrica

Como estas frecuencias son muy distintas, debería haber algún método para diferenciarlas entre sí, tal que sus diferencias sean amplificadas. Afortunadamente, las ondas sinusoidales forman una base $\mathcal{B}$ ortogonal entre sí, lo cual nos permitirá hacer esto mismo.

Para demostrar que $\mathcal{B} = \{\phi_0, \phi_1\phi_2, ...\phi_n\}$ es un conjunto ortogonal se debe cumplir que:

$$< \phi_j | \phi_k >= 0; \forall j \neq k$$

usando el producto interno definido como

$$< f|g >= \frac{1}{2\pi} \int_{-\pi}^{\pi} f(x)\overline{g(x)}dx \tag{2}$$

(donde $\overline{g(x)}$ es la compuesta conjugada de $g(x)$), obtenemos el siguiente resultado:

$$< \phi_j | \phi_k >= \frac{1}{2\pi} \int_{-\pi}^{\pi} e^{i(j-k)x}dx$$

a simple vista no pareciera que esta integral sea igual a cero, pero luego recordamos la fórmula de Euler:

$$e^{i\theta} = \cos(\theta) + i\sin(\theta)$$

por lo que obtenemos:

$$\frac{1}{2\pi} \int_{-\pi}^{\pi} e^{i(j-k)x}dx = \frac{1}{(j-k)\pi} \sin((j-k)\pi)$$

donde j y k son enteros positivos, por lo que su resta (que llamaremos m) es un número entre 1 y n. y $\sin(m\pi) = 0$ para todo m entero.

Ahora con la anterior demostración parece bastante tentador usar el proceso de Gram-Schmidt para ortogonalizar la señal recibida y así decodificar la señal, ya que tenemos a la base $\mathcal{B}$, y tenemos el producto interno definido en (2), así que podemos hacer la prueba.

$$f =< \phi_1|f > \phi_1 + < \phi_2|f > \phi_2 + \cdots + < \phi_n|f > \phi_n$$

Para usar ese interpolador debemos conocer f en su totalidad, pero si lo conociéramos, ¿para qué interpolar? Es mucho más práctico evaluar f. Desafortunadamente, debemos buscar otro camino.

$$< f|g >_n = \frac{1}{n}\sum_{k=0}^{n-1} f(x_k)\overline{g(x_k)} \tag{3}$$

Esta función es un pseudo-producto interno, ya que no cumple con una de las propiedades necesarias:

$$< x|x >\geq 0 \wedge < x|x >= 0 \Longrightarrow x = 0$$

Como contraejemplo asumamos que $f(x_k) = x - 0.5$ y que $x_0 = 0.5$, adicionalmente, tomamos a $n = 1$

$$< x|x >_1 = |(x_0 - 0.5)|^2 = 0$$

A pesar de que $x_0 \neq 0$ el resultado del producto interno es igual a cero, lo que contradice la propiedad, la cual es necesaria para que sea un producto interno.

Aun así, ella sí cumple con otras propiedades del producto interno:

$$< x|y >= \overline{< y|x >}$$

Lo podemos demostrar de la siguiente forma:

$$\overline{< g|f >} = \frac{1}{n}\sum_{k=0}^{n-1} \overline{g(x_k)\overline{f(x_k)}}$$

$$\frac{1}{n}\sum_{k=0}^{n-1} \overline{\overline{f(x_k)}g(x_k)} = \frac{1}{n}\sum_{k=0}^{n-1} f(x_k)\overline{g(x_k)}$$

Donde esto último no es más que la definición de $< f|g >$

Esta función también cumple la propiedad

$$< x|\alpha y + \beta z >= \alpha < x|y > +\beta < x|z >$$

donde $\alpha, \beta \in \mathbb{R}$

Se demuestra de la siguiente forma:

$$< f|\alpha g + \beta h >= \frac{1}{n}\sum_{k=0}^{n-1} f(x_k)(\overline{\alpha g(x_k) + \beta h(x_k)})$$

Distribuimos $f(x_k)$ y aplicamos propiedades de la sumatoria:

$$\frac{1}{n}\sum_{k=0}^{n-1} \alpha f(x_k)(\overline{g(x_k)}) + \beta f(x_k)\overline{h(x_k)} = \alpha\frac{1}{n}\sum_{k=0}^{n-1} f(x_k)(\overline{g(x_k)}) + \beta\frac{1}{n}\sum_{k=0}^{n-1} f(x_k)\overline{h(x_k)}$$

Pero esto no es más que: $\alpha < f|g > +\beta < f|h >$

La buena noticia es que los ϕ_k mantienen su ortonormalidad en el pseudo-producto interno:

$$< \phi_j | \phi_k >_n = \begin{cases} 1 & \text{si } (j-k) \text{ es divisible por } n \\ 0 & \text{en otros casos} \end{cases}$$

para $0 \leq k, j \leq n$. luego:

$$< \phi_j | \phi_k >_n = \frac{1}{n} \sum_{m=0}^{n-1} e^{i(j-k)x_m}$$

sea $p = j - k$, como j y k son enteros, p también es entero.

2.0.1 Caso 1: $j = k$

Si $j = k$, entonces $j - k = 0$ y $e^{i(j-k)x_m} = e^0 = 1$ por ende, nuestra fórmula se convierte:

$$< \phi_j | \phi_k >_n = \frac{1}{n} \sum_{m=0}^{n-1} 1 = \frac{1}{n} n = 1$$

2.0.2 Caso 2: $j \neq k$

Si son distintos, entonces $j - k = p$ donde p es un entero y $p \neq 0$,

$$< \phi_j | \phi_k >_n = \frac{1}{n} \sum_{m=0}^{n-1} e^{ipx_m} = \frac{1}{n} \sum_{m=0}^{n-1} e^{ipx_m}$$

para saber el valor, debemos elegir el valor de x_m. Como estamos discretizando funciones periódicas, deberíamos entonces tomar el período (2π) y dividirlo entre n espacios equidistantes, por lo que:

$$x_m = \frac{2\pi m}{n} \Rightarrow \frac{1}{n} \sum_{m=0}^{n-1} e^{ip\frac{2\pi m}{n}} = \frac{1}{n} \sum_{m=0}^{n-1} e^{(ip\frac{2\pi}{n})^m}$$

De esta forma se transforma en una suma geométrica, la cual será igual a

$$\frac{1 - e^{(ip\frac{2\pi}{n})^n}}{1 - e^{(ip\frac{2\pi}{n})}}$$

Sea $r = e^{ip\frac{2\pi}{n}}$

$$r^n = e^{(ip\frac{2\pi}{n})^n} = e^{(ip2\pi)}$$

Aplicando la fórmula de Euler:

$$e^{(ip2\pi)} = \cos(p2\pi) + i\sin(p2\pi) = 1 + 0 \cdot i = 1$$

Sustituimos de vuelta en la suma geométrica:

$$\frac{1-e^{(ip\frac{2\pi}{n})^n}}{1-e^{(ip\frac{2\pi}{n})}} = \frac{1-1^n}{1-e^{(ip\frac{2\pi}{n})}} = 0$$

por lo que en este caso

$$< \phi_j | \phi_k >_n = 0$$

Este pseudo-producto interno nos sirve mejor ya que para utilizar el producto interno continuo definido en (2) se requiere evaluar una integral, lo que es computacionalmente costoso e incluso puede ser incalculable si solo se tienen muestras discretas de f y g, como es en el caso de aplicaciones que requieren de procesamiento de señales. Este solo es aplicable si f y g son funciones continuas o integrables, lo que no siempre se cumple en datos que son discretos o son experimentales. Debido a esto, buscamos una forma de poder evitar estas limitaciones, lo cual puede ser logrado utilizando el casi-producto interno discreto.

$$< f | g >_n = \frac{1}{n} \sum_{k=0}^{n-1} f(x_k)\overline{g(x_k)}$$

Este se adapta a los datos discretos utilizando sumatorias en lugar de integrales, lo que es ideal para trabajar con muestras discretas. Además de esta adaptación, se obtiene mejor eficiencia computacional ya que las sumatorias son triviales de calcular numéricamente, incluso para grandes n.

Si bien buscamos adaptación a los datos discretos, es importante también tener consistencia con la interpolación trigonométrica, por tanto los nodos

$$x_k = \frac{2\pi k}{n}$$

Están elegidos para que este producto interno discreto mantenga la ortogonalidad de las funciones base. Así, manteniendo las propiedades clave en el contexto discreto.

Podemos concluir que el casi-producto interno discreto sí elimina las limitaciones prácticas del producto interno continuo, ya que: Mantiene las propiedades clave, es computacionalmente viable y no requiere conocer f de forma analítica, solo sus valores en nodos discretos. Esto nos resulta útil, ya que cuando buscamos construir de manera explícita el interpolador, la expresión:

$$p(x) = \sum_{j=0}^{n-1} c_j \phi_j(x) = \sum_{j=0}^{n-1} c_j e^{ijx}$$

Proporciona una fórmula cerrada para $p(x)$, donde los coeficientes c_j se calculan directamente a partir de los datos $f(x_k)$ usando:

$$c_j = \langle f, \phi_j \rangle_n = \frac{1}{n} \sum_{k=0}^{n-1} f(x_k) e^{-ijx_k}, \quad j = 0, 1, \ldots, n-1$$

Esto evita resolver sistemas lineales, como los que se pueden ver en la interpolación polinomial tradicional, reduciendo el costo computacional.

Si analizamos los coeficientes c_j en podemos ver que son exactamente los coeficientes de la transformada discreta de Fourier de la secuencia $\{f(x_k)\}$.

Esto nos permite utilizar algoritmos eficientes como FFT (Fast Fourier Transform) para calcular c_j en $O(nlogn)$ operaciones. Otro punto importante es que, en aplicaciones como las de análisis de señales, los c_j revelan las frecuencias dominantes de f; esto se puede ver sobre todo si f es una combinación de tonos puros, donde los c_j tendrán picos en las frecuencias correspondientes. Esto permite decodificar señales o filtrar componentes no deseados en nuestra aplicación.

3 Transformada Discreta de Fourier

Una buena idea en este punto sería reescribir estas fórmulas como productos matriz-vector, aprovechando el hecho de que son transformaciones lineales. Recordando la transformada discreta de Fourier, para un vector $\mathbf{y} = (y_0, y_1, \ldots, y_{n-1})^T$ se define como:

$$c_j = \frac{1}{n} \sum_{k=0}^{n-1} y_k \omega^{jk}, \quad j = 0, 1, \ldots, n-1,$$

donde $\omega = e^{-\frac{2\pi i}{n}}$.

Esta se puede escribir como un producto matriz-vector $c = Fy$, donde F es una matriz de nxn y sus elementos están dados por:

$$F_{j,k} = \frac{1}{n} \omega^{jk}, \quad j, k = 0, 1, \ldots, n-1.$$

Es decir, la matriz F tiene la forma:

$$F = \frac{1}{n} \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega & \omega^2 & \cdots & \omega^{n-1} \\ 1 & \omega^2 & \omega^4 & \cdots & \omega^{2(n-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{n-1} & \omega^{2(n-1)} & \cdots & \omega^{(n-1)(n-1)} \end{pmatrix}.$$

Esta matriz es una matriz de Vandermonde compleja, esto implica que es invertible, y su inversa está relacionada con la transformada inversa de Fourier.

Por supuesto, calcular una matriz inversa siempre es algorítmicamente costoso y problemático, así que requerimos una alternativa fácil. Afortunadamente para nosotros, esta alternativa existe:

$$F^H F = \frac{1}{n} I \tag{4}$$

Ahora, es importante mostrar una demostración de (4), para esto nos vamos a apoyar de la matriz de Fourier F definida anteriormente, teniendo:

$$F_{j,k} = \frac{1}{n} \omega^{jk}, \quad \omega = e^{-\frac{2\pi i}{n}},$$

y su transpuesta conjugada F^H con elementos:

$$F^H_{k,j} = \overline{F_{j,k}} = \frac{1}{n}\omega^{-jk},$$

Entonces la demostración de $F^H F = \frac{1}{n} I$ inicia tomando el producto $F^H F$ para el cual el elemento (m, l) de $F^H F$ es:

$$(F^H F)_{m,l} = \sum_{k=0}^{n-1} F^H_{m,k} F_{k,l} = \frac{1}{n^2} \sum_{k=0}^{n-1} \omega^{k(l-m)}.$$

Una vez desarrollado, podemos evaluar la suma en los siguientes casos:

Caso 1: $m = l$

$$\sum_{k=0}^{n-1} \omega^0 = n \implies (F^H F)_{m,m} = \frac{1}{n}.$$

Caso 2: $m \neq l$

La suma es una serie geométrica:

$$\sum_{k=0}^{n-1} \omega^{k(l-m)} = \frac{1 - \omega^{n(l-m)}}{1 - \omega^{(l-m)}} = 0,$$

dado que $\omega^n = 1$. Por tanto:

$$(F^H F)_{m,l} = 0.$$

Con estos casos, nos damos cuenta de que:

- Los elementos diagonales de $F^H F$ valen $1/n$.
- Los elementos no diagonales valen 0.

Por lo tanto: $F^H F = \frac{1}{n} I$ (4)

Dada la relación $F^H F = \frac{1}{n} I$ demostrada en el punto anterior, se puede deducir que la matriz inversa de F es

$$F^{-1} = nF^H.$$

Ahora, probaremos que la transformada inversa de Fourier definida como:

$$y_k = \sum_{j=0}^{n-1} c_j \overline{\omega}^{jk}, \quad \overline{\omega} = e^{\frac{2\pi i}{n}}.$$

Hay que aclarar que reescribimos $\overline{\omega}^{jk} = \omega^{-jk}$:

$$y_k = \sum_{j=0}^{n-1} c_j \omega^{-jk}.$$

Una vez teniendo nuestros parámetros a utilizar, vamos a realizar la expresión matricial. La expresión $y_k = \sum_{j=0}^{n-1} c_j \omega^{-jk}$. corresponde al producto matricial $y = nF^H c$, donde F^H es la matriz transpuesta conjugada de F, con elementos $F^H_{k,j} = \frac{1}{n}\omega^{-jk}$.

Dado que $F^{-1} = nF^H$, se cumple:

$$y = F^{-1}c.$$

Esto demuestra que la transformada discreta de Fourier inversa se puede expresar como el producto de la matriz inversa de Fourier F^{-1} por el vector de coeficientes C.

4 Consideraciones finales

El desarrollo de nuestro programa para la codificación y decodificación de estas señales DTMF se basa en estos principios matemáticos e informáticos estables, utilizando la conversión discreta de Fourier y su implementación efectiva para analizar y crear estas frecuencias. Casos como la ortogonalidad de las funciones sinusoidales, que se encuentra en casos continuos y discretos, permiten identificar con precisión y eficacia las frecuencias DTMF incluso en presencia de distorsiones. El pseudo-producto interno discreto nos facilita el procesamiento de señales de muestreo con el programa realizado, evitando la complejidad de las integrales y tomando los beneficios de los algoritmos rápidos.

Estos conceptos teóricos no solo refuerzan el enfoque adoptado, sino que también optimizan el rendimiento del programa, lo que le permite traducir con precisión los tonos DTMF utilizando aplicaciones prácticas en el procesamiento de señales.

References

[1] Stephen Roberts. SIGNAL PROCESSING & FILTER DESIGN. Lecture 7 - The Discrete Fourier Transform. https://www.robots.ox.ac.uk/ sjrob/Teaching/SP/l7.pdf Oxford Department of Engineering Science

[2] Paul A Lynn. AN INTRODUCTION TO THE ANALYSIS AND PROCESSING OF SIGNALS. 1989

[3] DTMF signaling. https://en.wikipedia.org/wiki/DTMF_signaling

[4] IMPLEMENTATION AND EVALUATION OF THE VANDERMONDE TRANSFORM. https://eurasip.org/Proceedings/Eusipco/Eusipco2014/HTML/papers/1569906583.pdf