

Transformada de Fourier

Gabriel Alexander Valdivia Medina, Giulia Alexa Naval Fernandez, Rodrigo Alonso
Torres Sotomayor

Universidad Católica San Pablo, Arequipa.

Abstract

Este trabajo está orientado en presentar la Transformada de Fourier (TF), explicar su funcionamiento, capacidades y versiones, así como proponer un algoritmo en el lenguaje de programación C++ para su uso práctico. Se implementará una aplicación típica del algoritmo a modo de ejemplificación de lo que se puede lograr con este.

Keywords: Transformada de Fourier, Transformada Rápida de Fourier, espectro de frecuencia, números complejos, dominio de tiempo, inversión de bits, roots, cubics, diagramas de mariposa

1. Introducción a la transformada de Fourier

1.1. Números complejos

Un número complejo se puede representar de 2 maneras: de la forma binomial o de la forma polar, también llamada trigonométrica.

Forma binomial

Usualmente se expresa de la forma $a + bi$, donde bi es la componente imaginaria e $i = \sqrt{-1}$. Esta forma sirve a modo de coordenadas para el plano complejo.

Forma polar o trigonométrica

A diferencia de la forma binomial, que representa coordenadas para un punto específico en el plano complejo, la forma polar da información de la distancia del origen de dicho punto, así como el ángulo de inclinación del vector que va desde el origen hasta el punto. La forma polar se puede mostrar con ecuaciones trigonométricas o con una ecuación exponencial. Esto basándose en la ecuación de Euler para funciones trigonométricas complejas:

$$e^{\pm ix} = \cos(x) \pm i \operatorname{sen}(x) \quad (1)$$

Forma polar:

$$r = \sqrt{a^2 + b^2}$$

$$a = r \cos(\alpha), b = r \operatorname{sen}(\alpha)$$

$$a + bi = r \cos(\alpha) + r \operatorname{sen}(\alpha) i$$

$$a + bi = r e^{i\alpha}$$

1.2. Serie de Fourier

Serie trigonométrica

La Serie de Fourier define cualquier función periódica como una suma de *sen* y *cos* con amplitudes determinadas y frecuencias iguales a múltiplos de la frecuencia de la función.

La ecuación de la serie de Fourier se define de la siguiente forma:

$$P(t) = \frac{a_0}{2} + \sum_{n=1}^{\infty} [a_n \cos(2\pi n \frac{1}{T} t) + b_n \operatorname{sen}(2\pi n \frac{1}{T} t)]$$

Entendiendo que $\frac{1}{T}$ es la frecuencia de $P(t)$. Si representamos esta frecuencia como f_0 , podemos definir:

$$\omega = 2\pi f_0$$

Por lo tanto la ecuación pasa a ser:

$$P(t) = \frac{a_0}{2} + \sum_{n=1}^{\infty} [a_n \cos(\omega n t) + b_n \operatorname{sen}(\omega n t)] \quad (2)$$

Donde se presentan distintas funciones *sen* y *cos* cuya frecuencia depende de n y cuyos coeficientes a_n y b_n muestran qué tanto influye dicha función *sen* o *cos* en la representación de la función $P(t)$. En el caso de a_0 , este valor representa dos veces el valor medio de la función $P(t)$, que normalmente es 0.

Serie exponencial

La serie exponencial representa la serie de Fourier desde otro punto de vista, pero sus resultados no son diferentes a los de la serie trigonométrica. El punto de partida de esta serie es la ecuación de Euler para funciones trigonométricas complejas. La serie exponencial de Fourier se define como:

$$P(t) = \sum_{n=-\infty}^{\infty} C_n e^{i\omega n t} \quad (3)$$

Coefficientes de Fourier

Para hallar los coeficientes C_n , a_n , b_n se hará uso del **producto interno** entre funciones. En este caso, el producto interno se obtiene de la integral del producto de dos funciones, y nos da información del parentesco de estas.

La definición del producto interno para funciones periódicas con discontinuidades finitas se da de la siguiente forma:

$$\langle X(t), Y(t) \rangle = \frac{1}{T} \int_T X(t) \cdot Y^*(t) dt$$

Donde $Y^*(t)$ es la conjugada de $Y(t)$, en caso esta sea compleja.

Por lo tanto, el coeficiente C_n nos queda:

$$\langle P(t), e^{j\omega n t} \rangle = \frac{1}{T} \int_T P(t) \cdot e^{-j\omega n t} dt \quad (4)$$

En el caso de a_n y b_n :

$$a_n = \frac{2}{T} \int_0^T \cos(2\pi n \frac{1}{T} t) \cdot P(t) dt$$
$$b_n = \frac{2}{T} \int_0^T \sen(2\pi n \frac{1}{T} t) \cdot P(t) dt$$

Los límites de integración se definen como un periodo T de la función $P(t)$.

Relación entre serie exponencial y trigonométrica

$$p(t) = \frac{a_0}{2} + \sum_{n=1}^{\infty} [a_n \cos(n\omega t) + b_n \text{sen}(n\omega t)] =$$

$$\frac{a_0}{2} + a_1 \cos(\omega t) + b_1 \text{sen}(\omega t) + a_2 \cos(2\omega t) + b_2 \text{sen}(2\omega t) + \dots$$

$$p(t) = \sum_{n=-\infty}^{\infty} c_n e^{j\omega n t} =$$

$$\dots + c_{-2} e^{-j2\omega t} + c_{-1} e^{-j\omega t} + c_0 + c_1 e^{j\omega t} + c_2 e^{j2\omega t} + \dots$$

Figura1. Relación entre serie exponencial de Fourier y serie trigonométrica

2. Transformada de Fourier

La transformada de Fourier nos permite, entre muchas cosas, cambiar la forma en la que miramos una señal o potencia, pasando de mirar su comportamiento en base de tiempo a verla en base a su frecuencia.

Se relaciona con la Serie de Fourier, ya que puede representar todas las amplitudes de las curvas encontradas en esta serie.

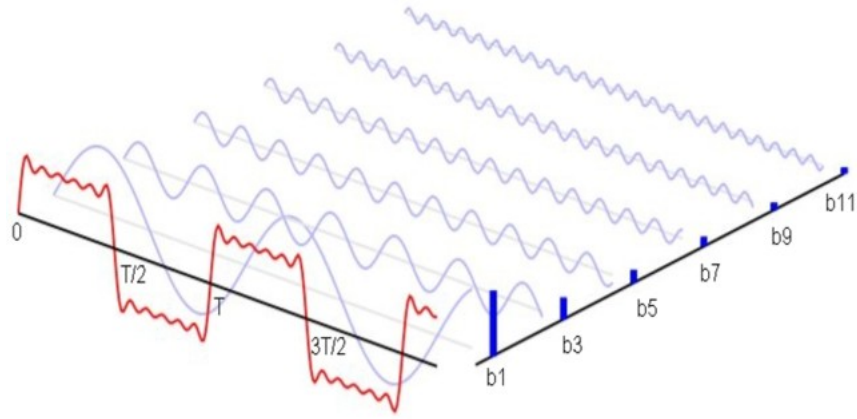


Figura2. En rojo, la señal recibida. En azul intenso, la transformada de Fourier. En morado, las series de Fourier. Fuente: wikimedia commons.

La transformada se halla con el producto interno entre la señal recibida y una exponencial compleja de frecuencia fija:

$$X(f) = \int_{-\infty}^{\infty} X(t) \cdot e^{-2\pi i f t} dt \quad (5)$$

Donde $X(t)$ es la señal en base tiempo definida, $X(f)$ la vista en base frecuencia que se busca. El cambio más común es de tiempo a frecuencia, pero no es el único que la transformada puede manejar.

2.1. Ejemplo 1

Hallaremos la transformada de la siguiente función:

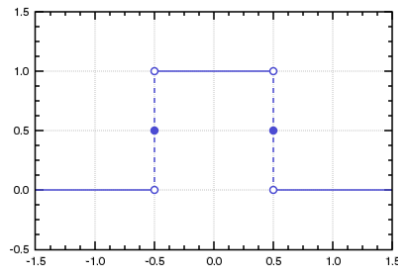


Figura 3. Función rectangular, función cajón, o pulso unitario.

Para hallar la transformada de una función por partes, se hallan las transformadas de las partes por separado:

$$X(f) = \int_{-\infty}^{-0.5} rect(t) \cdot e^{-2\pi i f t} dt + \int_{-0.5}^{0.5} rect(t) \cdot e^{-2\pi i f t} dt + \int_{0.5}^{\infty} rect(t) \cdot e^{-2\pi i f t} dt$$

Sin embargo, dado que $rect(t)$ es igual a 0 tanto en la primera como en la tercera parte, nos quedamos sólo con la transformada del medio.

$$X(f) = \int_{-0.5}^{0.5} 1 \cdot e^{-2\pi i f t} dt$$

Desarrollamos la integral:

$$X(f) = \left[\frac{e^{-i2\pi f t}}{-i2\pi f} \right]_{-0.5}^{0.5}$$

$$X(f) = \frac{e^{-i\pi f} - e^{i\pi f}}{-2\pi f i}$$

Por la fórmula de Euler, sabemos que las funciones *sen* y *cos* se relacionan con ecuaciones exponenciales de la forma:

$$\cos(\theta) = \frac{1}{2}(e^{+i\theta} + e^{-i\theta}) \quad (6)$$

$$\sin(\theta) = \frac{1}{2i}(e^{+i\theta} - e^{-i\theta}) \quad (7)$$

Entonces, podemos acomodar nuestra ecuación para que satisfaga alguna:

$$X(f) = \frac{-e^{-i\pi f} + e^{i\pi f}}{2i} \cdot \frac{1}{\pi f}$$

$$X(f) = \frac{\sin(\pi f)}{\pi f}$$

$$X(f) = \text{sinc}(\pi f)$$

3. Transformada Inversa de Fourier

La transformada inversa de Fourier revierte el estado en que se encuentra la representación de una onda luego de aplicarle la Transformada de Fourier normal. De forma más específica, convierte una serie de tamaño potencia de 2 números complejos, puntos en el espectro de frecuencia, en una serie del mismo tamaño en un dominio de tiempo.

$$x(t) = \int_{-\infty}^{\infty} X(f) \cdot e^{i2\pi ft} df \quad (8)$$

4. Propiedades de la transformada de Fourier

4.1. Traslación

Tenemos una función que se transforma de modo

$$X(t) \supset X(f)$$

Entonces, si modificamos la variable t por $t \pm b$, siendo b una constante:

$$X(t \pm b) \supset e^{\pm i2\pi fb} \cdot X(f)$$

4.2. Escalamiento

Si modificamos la variable t por at , siendo a una constante:

$$X(at) \supset \frac{1}{|a|} \cdot X\left(\frac{f}{a}\right)$$

Si el tiempo se multiplica por a , la frecuencia se reducirá por a .

Combinando Traslación y Escalamiento

Al combinar ambas propiedades es importante una correcta interpretación. Si tenemos $X(at - b)$, se puede interpretar de la forma:

$$\begin{aligned} &= X(at - b) \\ &= X(a(t - \frac{b}{a})) \end{aligned}$$

Primero se realiza la traslación en base a $-\frac{b}{a}$. Luego, el Escalamiento en base a a . Suponemos que la función $X(at)$ sin traslación se transforma en $Y(f)$

$$\begin{aligned} X(at) &\supset Y(f) \\ X(a(t - \frac{b}{a})) &\supset e^{-i2\pi f \frac{b}{a}} \cdot Y(f) \\ X(a(t - \frac{b}{a})) &\supset e^{-i2\pi f \frac{b}{a}} \cdot \frac{1}{|a|} X(\frac{f}{a}) \end{aligned}$$

5. Transformada de Fourier discreta

Para muchas de las aplicaciones de la transformada de Fourier, vamos a necesitar usar la transformada de Fourier Discreta (DFT) en su lugar, pues los elementos computacionales no son continuos, es decir, no son infinitos.

La DFT trabaja con un conjunto finito de puntos de una función, y los transforma a puntos complejos que nos darán información de la función.

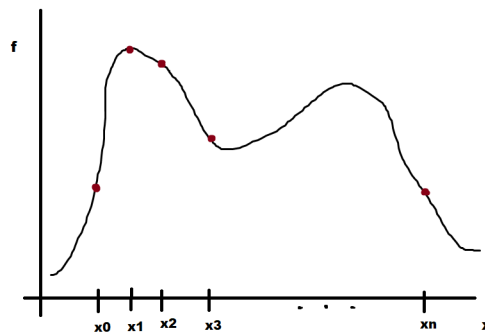


Figura 4. Puntos finitos para la DFT.

A pesar del nombre, la DFT se relaciona mucho con la serie de Fourier, ya que sus resultados indican el grado de relación de una determinada función sinusoidal con la función principal.

Para calcular la DFT se usa la siguiente ecuación:

$$\hat{f}_k = \sum_{j=0}^{n-1} f_j \cdot e^{-i2\pi j \frac{k}{n}} \quad (9)$$

Donde \hat{f}_k hace referencia al punto transformado y f_j al punto no transformado.

$$\{f_1, f_2, \dots, f_n\} \xrightarrow{DFT} \{\hat{f}_1, \hat{f}_2, \dots, \hat{f}_n\}$$

La exponencial compleja se puede expresar en términos de *sen* y *cos* de acuerdo a la fórmula de Euler, por lo que representa una función sinusoidal con características dependientes del punto en cuestión (j) que se comparará para encontrar su relación con la función original.

Podemos definir una frecuencia fundamental ω para los valores en nuestros n puntos.

$$\omega_n = e^{\frac{-2\pi i}{n}}$$

Podemos notar que todas las transformadas están añadiendo múltiplos a esta frecuencia fundamental. Usando esto podemos mapear todas las operaciones de la ecuación en una matriz de la siguiente forma:

$$\begin{bmatrix} \hat{f}_1 \\ \hat{f}_2 \\ \hat{f}_3 \\ \dots \\ \hat{f}_n \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & \omega_n & \omega_n^2 & \dots & \omega_n^{n-1} \\ 1 & \omega_n^2 & \omega_n^4 & \dots & \omega_n^{2(n-1)} \\ \dots & \dots & \dots & \dots & \dots \\ 1 & \omega_n^{n-1} & \omega_n^{2(n-1)} & \dots & \omega_n^{(n-1)^2} \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ \dots \\ f_n \end{bmatrix}$$

Como el DFT trabaja con números complejos, sus resultados también serán complejos. Esto es útil para la interpretación de los resultados. La magnitud de cada número representará qué tanto se tiene de esa frecuencia, y su ángulo representará su fase (desplazamiento del origen).

6. Algoritmo FFT (Cooley Tukey)

Pensando en formas de implementar y mejorar la TFD, James W. Cooley y John W. Tukey publicaron en su paper insigne en 1965 una implementación de la transformada utilizando el paradigma "divide y vencerás", denominada como Transformada Rápida de Fourier. En dicho trabajo, se destaca la ventaja que presenta el algoritmo en arreglos de tamaño N potencia de 2 sobre otros con N distintos, al lograr una complejidad de $O(N \log(N))$ para el primer caso.

Este algoritmo consigue reducir el número de cálculos al dividir el arreglo de tamaño N del polinomio en 2 de tamaño $N/2$, uno con el contenido de los índices impares y el otro con el de los índices pares.

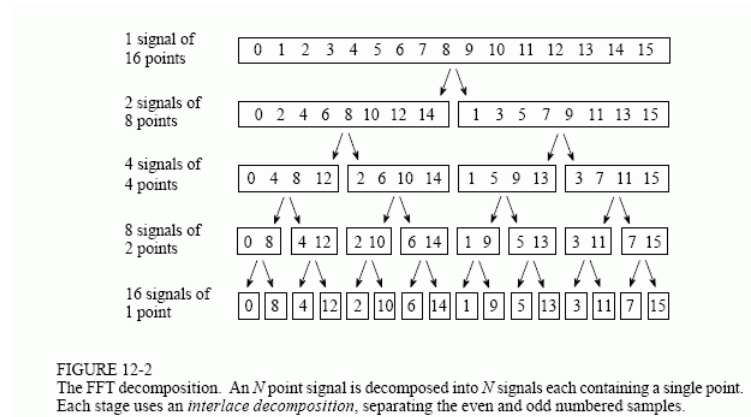


Figura 5. Descomposición de arreglo de coeficientes del polinomio. Fuente: The Scientist and Engineer's Guide to Digital Signal Processing.

De esta forma, se puede simplificar el tiempo que toma la multiplicación de matrices en la DFT segmentando esta matriz en otras más pequeñas. Suponemos que tenemos un $\hat{f} = F_{1024}$. Es decir, una matriz de $1024 * 1024$. Para segmentar esta matriz se usa el criterio de índices pares o impares, quedando lo siguiente:

$$\hat{f} = F_{1024} \cdot f = \begin{bmatrix} I_{512} & D_{512} \\ I_{512} & D_{512} \end{bmatrix} \begin{bmatrix} F_{512} & 0 \\ 0 & F_{512} \end{bmatrix} \begin{bmatrix} f_{even} \\ f_{odd} \end{bmatrix}$$

10

Donde f_{even} son los índices pares, f_{odd} los índices impares. I_{512} es la matriz identidad de tamaño $512 * 512$, y D_{512} está dado por:

$$D_{512} = \begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ 0 & \omega & 0 & \dots & 0 \\ 0 & 0 & \omega^2 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & \omega^{511} \end{bmatrix}$$

Asímismo, la matriz del medio se podrá dividir de nuevo siguiendo el mismo criterio y de manera recursiva.

Esta distribución también se puede lograr a través usando inversión de bits.

Una vez dividido el arreglo en partes más pequeñas, el algoritmo busca encontrar su

Sample numbers in normal order			Sample numbers after bit reversal	
Decimal	Binary		Decimal	Binary
0	0000	⇒	0	0000
1	0001		8	1000
2	0010		4	0100
3	0011		12	1100
4	0100		2	0010
5	0101		10	1010
6	0110		6	0100
7	0111		14	1110
8	1000		1	0001
9	1001		9	1001
10	1010		5	0101
11	1011		13	1101
12	1100		3	0011
13	1101		11	1011
14	1110		7	0111
15	1111		15	1111

FIGURE 12-3
The FFT bit reversal sorting. The FFT time domain decomposition can be implemented by sorting the samples according to bit reversed order.

Figura 6. Ordenamiento por inversión de bits. Fuente: The Scientist and Engineer's Guide to Digital Signal Processing.

valor en el espectro de frecuencia, donde obviamente, la frecuencia de las divisiones más pequeñas, es decir de 1 solo punto, será igual a sí mismo. Luego de esto, tendrá que regresar y juntar los resultados en el orden inverso al que se hicieron las divisiones. Para sumar los puntos de cada una, también se tienen que diluir la señal completando con 0s las posiciones pares o impares, dependiendo del sub-arreglo, lo que equivale a una

duplicación de la data de la señal en el dominio de la frecuencia. Esta discordancia en la separación con 0s, o de forma más precisa, este corrimiento del segundo sub-arreglo corresponde a la multiplicación del mismo por un senoide.

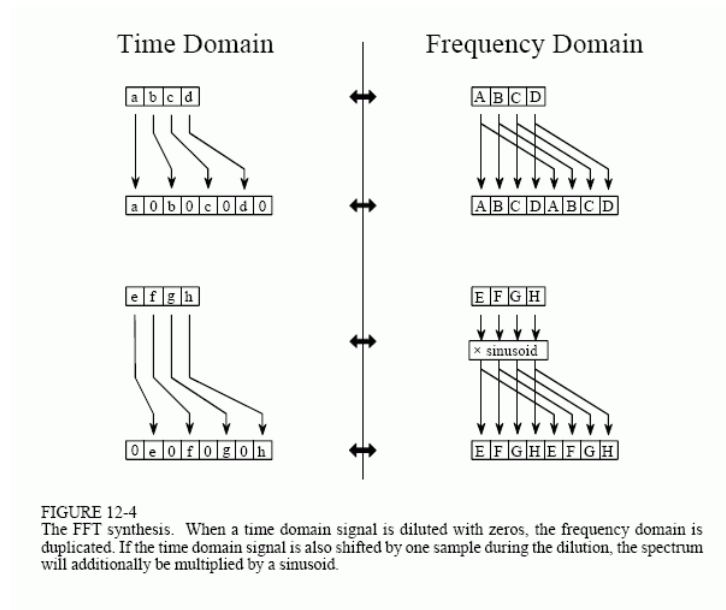


Figura 7. Separación de puntos con 0s y multiplicación por sinusoides. Fuente: The Scientist and Engineer's Guide to Digital Signal Processing.

Esta forma de sumar los puntos del espectro mientras se los reordena, tiene el nombre de **mariposa** y es la operación base de toda la TFR.

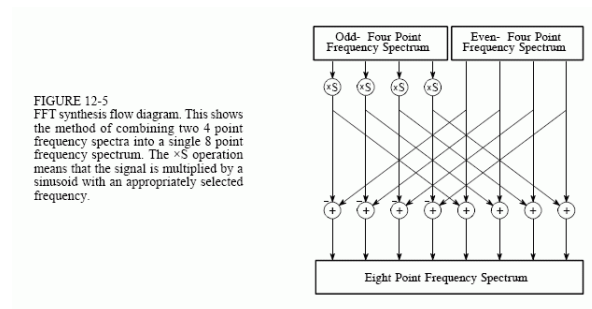


Figura 8. Flujo cruzado de operaciones o *mariposas*. Fuente: The Scientist and Engineer's Guide to Digital Signal Processing.

7. Implementación de los algoritmos

- Fast Fourier Transform

```
void fft(vector<complex<double>>& v)
{
    int N = v.size();
    if (N <= 1) return;

    vector<complex<double>> impar;
    vector<complex<double>> par;
    for (int i=0; i<N-1; i+=2) {
        par.push_back(v[i]);
        impar.push_back(v[i+1]);
    }

    fft(par);
    fft(impar);

    for (int k = 0; k < N/2; ++k)
    {
        complex<double> t = std::polar(1.0, -2 * PI * k / N) * impar[k];
        v[k] = par[k] + t;
        v[k+N/2] = par[k] - t;
    }
}
```

- Inverse Fast Fourier Transform

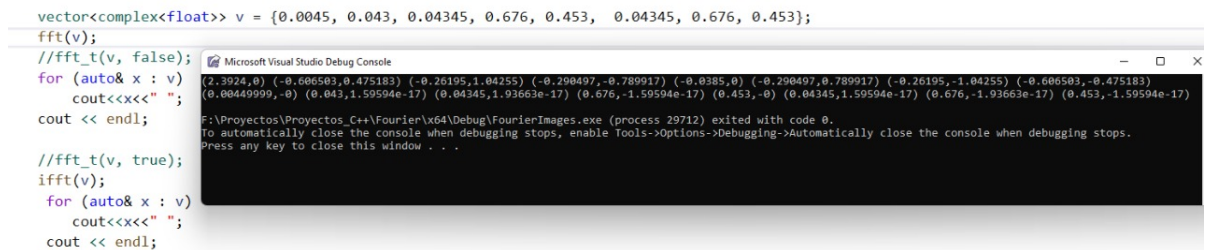
```
void ifft(vector<complex<double>>& v)
{
    for (auto& x : v)
        x = conj(x);
    fft(v);
    complex<double> s(v.size(),0);
    for (auto& x : v)
        x = conj(x) / s;
}
```

Apéndice

7.1. Prueba FFT y IFFT

Se realiza la prueba del buen funcionamiento de las funciones mostradas anteriormente del FFT y su inversa, IFFT.

Para esta prueba hemos creado un vector de tipo complex de datos float, ya que es el tipo de dato con el opera el fft, luego se realiza una impresión de los datos convertidos y finalmente se aplica el ifft, con su respectiva impresión para mostrar que nos devuelve los datos iguales a los ingresados originalmente.



```
vector<complex<float>> v = {0.0045, 0.043, 0.04345, 0.676, 0.453, 0.04345, 0.676, 0.453};
fft(v);
//fft_t(v, false);
for (auto& x : v)
    cout<<x<<" ";
cout << endl;

//fft_t(v, true);
ifft(v);
for (auto& x : v)
    cout<<x<<" ";
cout << endl;
```

Microsoft Visual Studio Debug Console

```
(2.3924,0) (-0.606503,0.475183) (-0.26195,1.04255) (-0.290497,-0.789917) (-0.0385,0) (-0.290497,0.789917) (-0.26195,-1.04255) (-0.606503,-0.475183)
(0.00449999,-0) (0.043,1.59594e-17) (0.04345,1.93663e-17) (0.676,-1.59594e-17) (0.453,-0) (0.04345,1.59594e-17) (0.676,-1.93663e-17) (0.453,-1.59594e-17)
F:\Proyectos\Proyectos_C++\Fourier\64\Debug\FourierImages.exe (process 29712) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .
```

Figura 9. Prueba realizada.

Bibliografía

Bhavani, R. D., Sudhakar, D. (2013). Design and implementation of inverse fast fourier transform for OFDM. International Journal of Science and Engineering Applications, 2(7), 155-157.

Pedraza, D. (06 de 2022). El Traductor de Ingeniería. Obtenido de <https://www.youtube.com/c/ElTraductordeIngeniería>

Steven L. Brunton, J. N. (2017). Data Driven Science Engineering. Obtenido de Machine Learning, Dynamical Systems, and Control: <http://databookuw.com/>

Villar, A. C. (s.f.). Fourier y sus coeficientes. Universidad de Granada.