

LA TRANSFORMADA DE FOURIER Y SUS APLICACIONES EN LA FÍSICA

J. E. Loera,¹ C. E. Valladares¹ E. A. Leija¹

RESUMEN

En este proyecto se busca la obtención de conocimiento acerca del diagrama H-R, debido a su gran importancia en la Astronomía. Cada una de las características de las estrellas que pueden proporcionarnos los tipos de diagramas ha sido de gran importancia y aportación para acercarse al entendimiento de la vida de las estrellas. Esto se debe a que las estrellas han sido un gran misterio desde que se conoce que son como nuestro Sol, tan enigmático como todos las demás. Aquí hablaremos de los tipos de diagramas H-R que suelen utilizarse, cómo se usan y qué información podemos conocer con ellos. Además, se construirá con una base de datos un propio diagrama H-R para la sencilla explicación de cada una de las partes que lo dividen.

ABSTRACT

Aquí va el abstract

Key Words: DFT — FFT — Transformada de Fourier — Espectro — Algoritmo

1. INTRODUCCIÓN

Conforme se avanza en el estudio de la física, llega un momento donde es inevitable encontrarse con la Transformada de Fourier para la simplificación y solución de problemas, es una herramienta muy útil dentro de la física. Sus aplicaciones van desde el análisis armónico de señales periódicas hasta la solución de ecuaciones diferenciales y la descripción de fenómenos ondulatorios en mecánica clásica, electrodinámica, mecánica cuántica y muchas más.

En este trabajo la estudiaremos e implementaremos computacionalmente para resolver problemas de la física.

2. LA TRANSFORMADA DE FOURIER

Las transformadas integrales como lo son la transformada de Laplace o la transformada de Fourier, tienen una gran aplicación en muchas ramas de la ciencia, en este trabajo nos centraremos en la transformada de Fourier que puede ser usada para simplificar complejos problemas matemáticos. De manera poco rigurosa podemos decir que nos lleva una función del espacio en el que está definida al espacio de las frecuencias.

Sea $f(x)$ una función que satisface las condiciones de Dirichlet, entonces se dice que tiene una representación en series de Fourier.

Teorema 2.1 (Teorema de Dirichlet) Si se cumple que:

1. Si $f(x)$ tiene periodo $2L$ y es univaluada entre $-L$ y L .
2. Tiene un número finito de máximos y mínimos.
3. Tiene un número finito de discontinuidades.
4. Y si $\int_{-L}^L \|f(x)\| dx$ es finita.

Entonces $f(x)$ tiene representación en series de Fourier y converge a ella en todos los puntos donde $f(x)$ es continua y al punto medio donde $f(x)$ es discontinua.

Definición 2.1 (Series de Fourier) Si $f(x)$ satisface las condiciones de Dirichlet entonces su representación en series de Fourier está dada por:

$$f(x) = \sum_{n=-\infty}^{\infty} c_n e^{i(n\pi x/L)} \quad (1)$$

Donde los coeficientes $\{c_n\}$ suelen ser llamados como el *espectro de $f(x)$* .

El espectro anterior es un espectro discreto que puede representar ciertas funciones periódicas, sin embargo, existen otras funciones periódicas o ciertos fenómenos físicos en los que se necesita de un espectro continuo. Notemos como la serie de Fourier está dada por una

¹Universidad Autónoma de Nuevo León, Facultad de Ciencias Físico Matemáticas, San Nicolás de los Garza, Nuevo León, México

suma infinita, por lo cual no debería sorprendernos que bajo ciertas condiciones podamos reemplazarla por la integral de Fourier.

Definición 2.2 (La transformada de Fourier)

Si una función $f(x)$ satisface las condiciones de Dirichlet y si $\int_{-\infty}^{\infty} \|f(x)\| dx$ es finita entonces:

$$F(k) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} f(x) e^{ikx} dx \quad (2)$$

Y la transformada inversa de Fourier está dada por:

$$f(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} F(k) e^{-ikx} dk \quad (3)$$

3. LA TRANSFORMADA DE DISCRETA DE FOURIER DFT

Notemos que una transformada de Fourier implica la evaluación de integrales y cierto manejo del álgebra, sin embargo, es posible implementar un algoritmo que nos permita encontrar la transformada de Fourier a un conjunto de datos discreto que contenga la información de la función a transformar. A este algoritmo le llamamos la *Transformada Discreta de Fourier o DFT*.

Definición 3.1 (DFT) Sea X_n un arreglo con N terminos que contiene la información de una función, la Transformada Discreta de Fourier (DFT) de cada elemento del arreglo numérico está por:

$$X_k = \sum_{n=1}^{N-1} x_n e^{-i2\pi kn/N} \quad (4)$$

Y la Transformada Inversa Discreta de Fourier (también denotada por DFT inversa o IDFT) está dada por:

$$x_n = \frac{1}{N} \sum_{k=1}^{N-1} X_k e^{i2\pi kn/N} \quad (5)$$

Resaltamos que tanto x_n como X_k pueden tomar valores complejos.

4. LA TRANSFORMADA RAPIDA DE FOURIER DFT

Por lo general, tenemos que lidiar con una gran cantidad de puntos de datos, y la velocidad del algoritmo para la transformada de Fourier se convierte en un tema muy importante. El algoritmo de la DFT tiene un costo computacional muy grande, conforme crece el

tamaño del conjunto de datos, de hecho su complejidad es de $O(n^2)$ (donde n es el tamaño del conjunto de datos).

La Transformada Rápida de Fourier (FFT) es un algoritmo eficiente para calcular la DFT de una secuencia. Se describe por primera vez en el artículo clásico de Cooley y Tukey en 1965, aunque se hizo referencia a este por primera vez en un trabajo no publicado de Gauss en 1805. Este algoritmo reduce considerablemente los costos computacionales pues su complejidad es de $O(n \log(n))$ (donde n es el tamaño del conjunto de datos).

Este algoritmo se aprovecha de varias simetrías a la hora de calcular la DFT, y será implementado en Python para el desarrollo de este trabajo.

5. EL ATRACTOR DE LORENTZ

Uno de los sistemas más icónicos cuando se empieza el estudio de la teoría del caos es el de las *ecuaciones de Lorenz*.

Definición 5.1 (Ecuaciones de Lorenz)

$$\dot{x} = \sigma(y - x) \quad (6)$$

$$\dot{y} = x(\rho - z) - y \quad (7)$$

$$\dot{z} = xy - \beta z \quad (8)$$

El atractor de Lorenz es un concepto introducido por Edward Lorenz en 1963. Se trata de un sistema dinámico determinista tridimensional no lineal derivado de las ecuaciones simplificadas de rolos de convección que se producen en las ecuaciones dinámicas de la atmósfera terrestre.

Al parametro σ se le suele denominar el número de Prandtl y ρ se llama el número de Rayleigh.

Fijemos los parametros $\sigma = 10$ y $\beta = 8/3$ y variemos el número de Rayleigh entre 1 y 100 para hacer una inspección sobre de la sensibilidad a las condiciones iniciales del atractor de Lorenz. Emplearemos la siguiente metodología:

1. Resolver numéricamente el sistema de ecuaciones diferenciales.
2. Aplicar la transformada de Fourier a la solución de la ecuación diferencial.
3. Analizar gráficamente el espectro y la fase.

Hemos encontrado algunos puntos donde la sensibilidad a las condiciones iniciales de hace muy presente y estos son algunos de ellos:

1. $\rho \in (14.5500, 14.5599)$
2. $\rho \in (24.97, 24.98)$
3. $\rho \in (27.08, 27.09)$

En cada uno de estos cambios se observó una considerable variación en el comportamiento del sistema, ante mínimas variaciones del número de Rayleigh, hay otros casos en los cuales también se observan perturbaciones y estos son solo algunos de dichos casos.

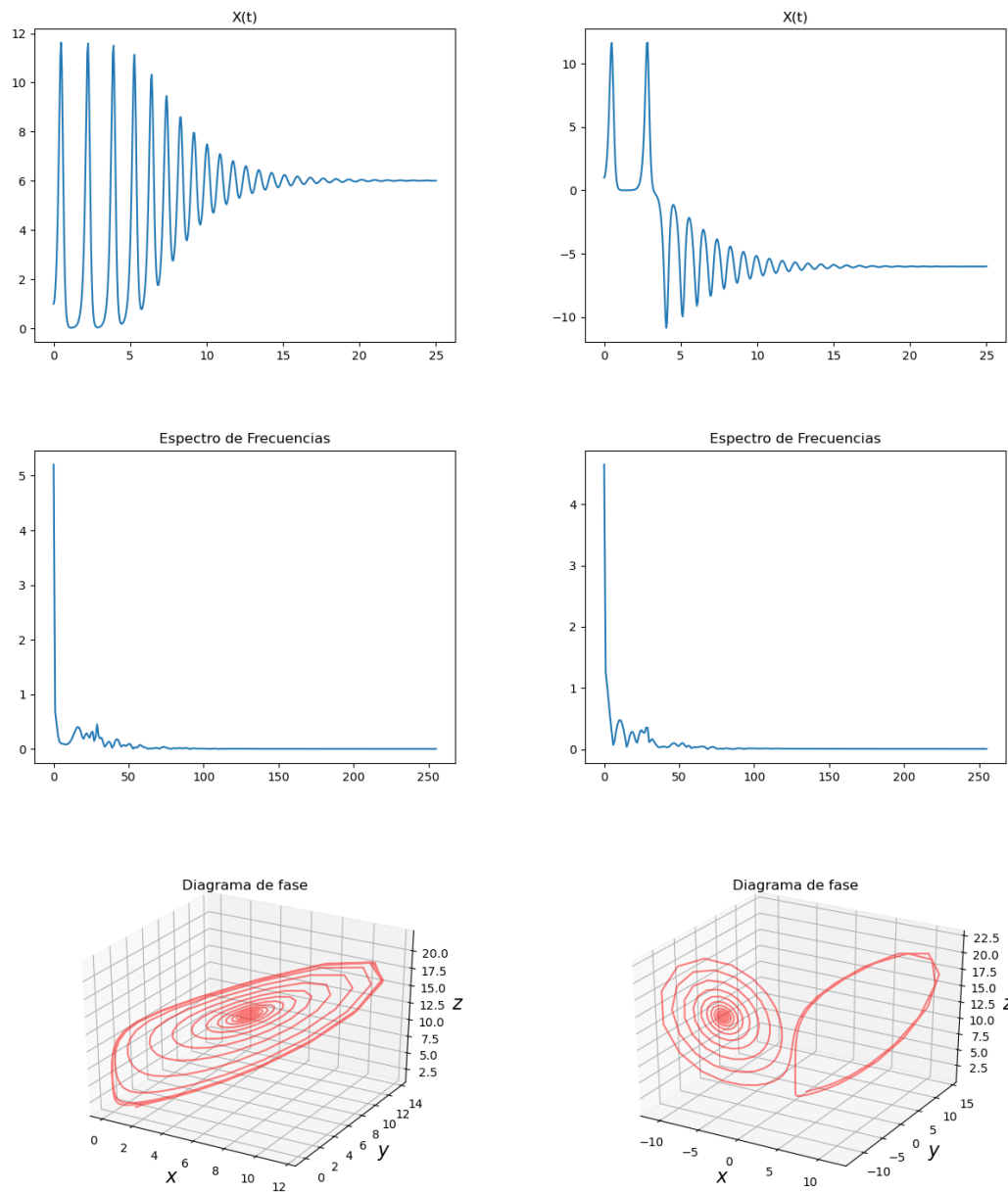


Fig. 1. $\rho = 14.5500 \Rightarrow \rho = 14.5899$

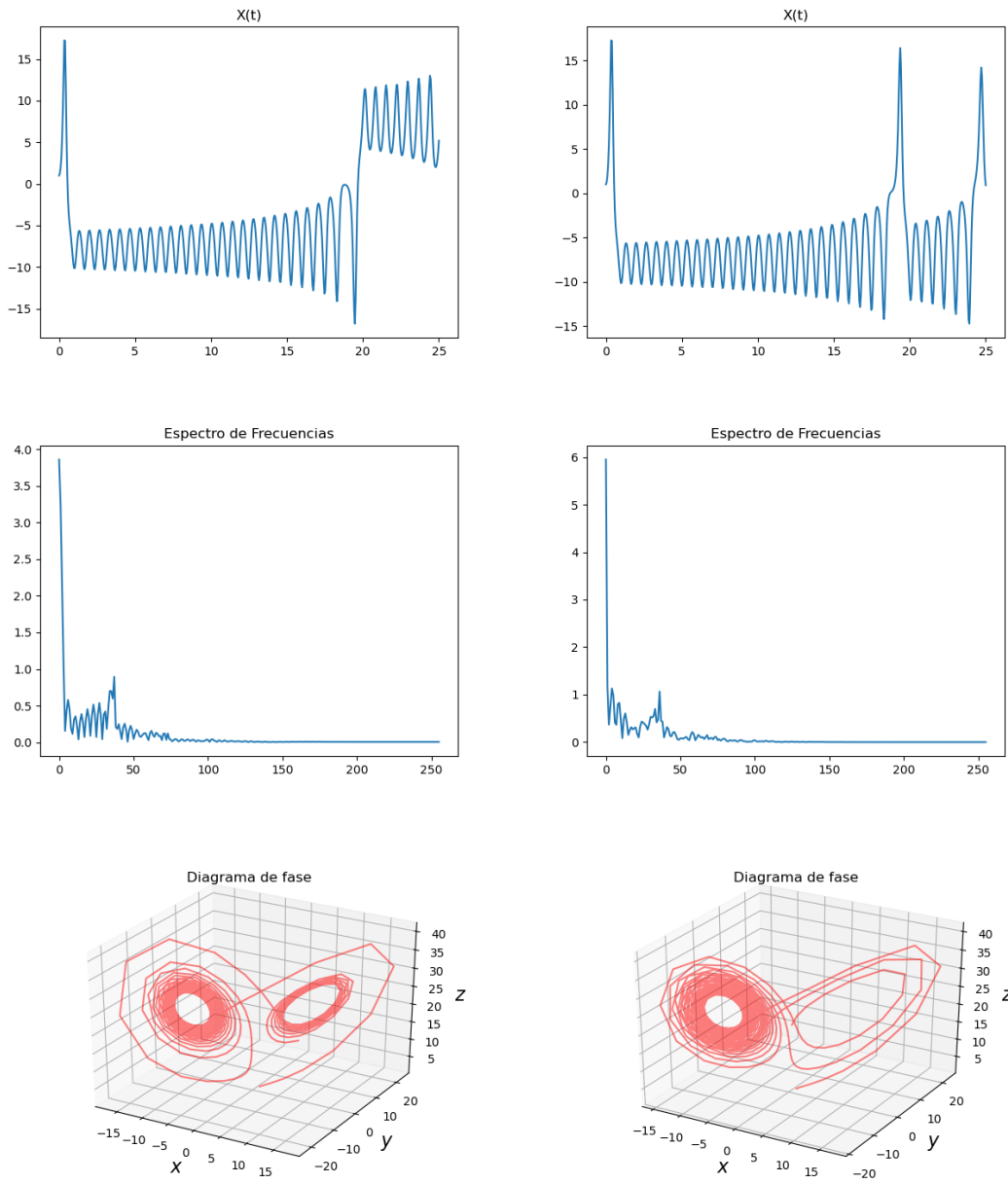
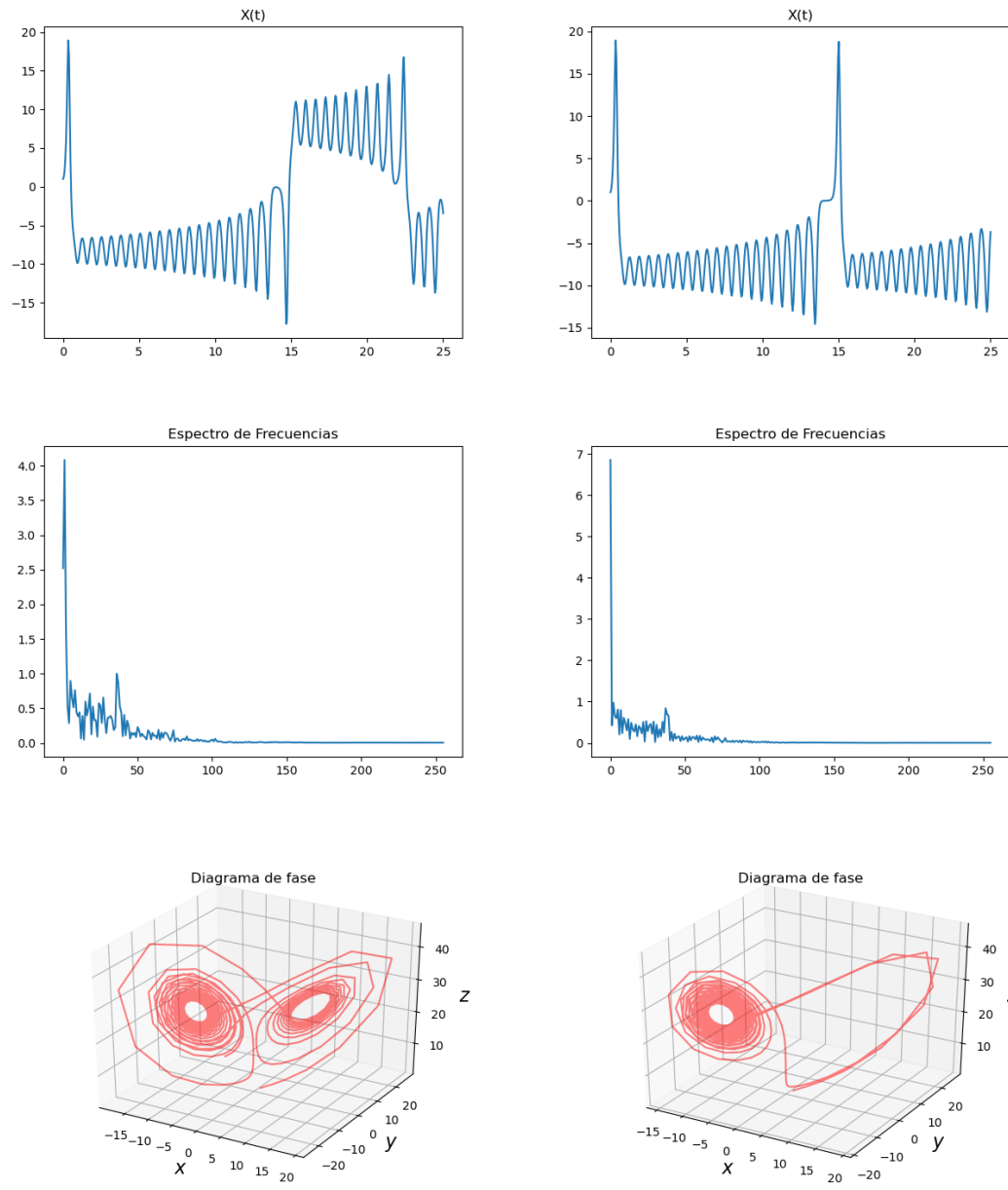


Fig. 2. $\rho = 24.97 \Rightarrow \rho = 24.98$

Fig. 3. $\rho = 27.08 \Rightarrow \rho = 27.09$

Apéndice A

Método en python con el algoritmo de la DFT

```
def DFT(F, N):
    Im_F = F[0]
    Re_F = F[1]
    Im_G = []
    Re_G = []
    G = []
    Freq = []

    for k in range(N):
        im_g = 0
        re_g = 0
        for n in range(N):
            re_g = (
                np.cos((2 * np.pi * k * n) / (N)) * Re_F[n]
                + np.sin((2 * np.pi * k * n) / (N)) * Im_F[n]
            )
            im_g = (
                np.cos((2 * np.pi * k * n) / (N)) * Im_F[n]
                - np.sin((2 * np.pi * k * n) / (N)) * Re_F[n]
            )
            Im_G = np.append(Im_G, im_g)
            Re_G = np.append(Re_G, re_g)
        G = np.append(G, [re_g, im_g])
        Freq = np.append(Freq, k)

    return G, Re_G, Im_G, Freq
```

Apéndice B

Método en python con el algoritmo de la FFT

```
def FFT(f):
    N = len(f)
    if N <= 1:
        return f

    # division
    even = FFT(f[0::2])
    odd = FFT(f[1::2])

    # store combination of results
    G = np.zeros(N).astype(np.complex64)

    # only required to compute for half the frequencies
    # since u+N/2 can be obtained from the symmetry property
    for u in range(N // 2):
        G[u] = even[u] + np.exp(-2j * np.pi * u / N) * odd[u] # conquer
        G[u + N // 2] = even[u] - np.exp(-2j * np.pi * u / N) * odd[u] # conquer

    return G
```

Apéndice C

Código donde se simuló el atractor de Lorenz

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.integrate import odeint
from Methods import FFT

"""
El atractor de Lorenz
"""

def Lorenz_ED(xyz, t, sigma, rho, beta):
    x, y, z = xyz
    return [sigma * (y - x), x * (rho - z) - y, x * y - beta * z]

"""
Discretizamos la variable dependiente en N=2**m terminos,
el cual es un requerimiento de la Transformada Rapida de Fourier
"""
m = 9
N = 2 ** m

"""
Condiciones iniciales y parametros del Atractor de Lorenz
"""

# Asignamos valores a los parametros
sigma, Rho, beta = 10, np.linspace(24, 25, 100), 8.0 / 3.0

# Condicion inicial y valores de t sobre los que calcular
xyz0 = [1.0, 1.0, 1.0]
xi = 0.0
xf = 25.0
t = np.linspace(xi, xf, N)
iter = 1

for rho in Rho:

    # Resolvemos las ecuaciones
    xyz = odeint(Lorenz_ED, xyz0, t, args=(sigma, rho, beta))

    # Graficamos las soluciones
    from mpl_toolkits.mplot3d.axes3d import Axes3D

    fig, (ax1) = plt.subplots(1, 1, subplot_kw={"projection": "3d"})
    ax1.plot(xyz[:, 0], xyz[:, 1], xyz[:, 2], "r", alpha=0.5)
    ax1.set_xlabel("$x$", fontsize=16)
    ax1.set_ylabel("$y$", fontsize=16)
    ax1.set_zlabel("$z$", fontsize=16)
    plt.title("Diagrama de fase")
    plt.savefig("images_numpy_24_25/fase/fase" + str(iter) + ".png")
    plt.close()

"""
Calculamos la FFT de X(t) de las ecns de Lorenz
"""
G_Lorenz_X = FFT(xyz[:, 0])

"""
Creamos un arreglo con los espectros de potencia de la FFT
"""
S_fft = []
for i in range(N):
    s = (1.0 / N) * abs(G_Lorenz_X)
    S_fft = np.append(S_fft, s)
```

```
"""
Graficamos la funcion Fx_fft
"""
plt.plot(t, xyz[:, 0])
plt.title("X(t)")
plt.savefig("images_numpy_24_25/eje_x/eje_x" + str(iter) + ".png")
plt.close()

"""
Ahora Graficamos S vs frecuencia (Espectro)
"""
Freq = np.arange(0, N // 2)
plt.plot(Freq, S_fft[: N // 2])
plt.title("Espectro de Frecuencias")
plt.savefig("images_numpy_24_25/espectro/espectro" + str(iter) + ".png")
plt.close()

iter = iter + 1
```


REFERENCES

- Scherer J. (2010) Computational Physics
Boas M. (2006) Mathematical methods in the
physical sciences
Butkov E. (1988) Física Matemática
Stickler, D. (2016) Basic concepts in computa-
tional physics
Pang T. (1999) An introduction to computational
physics