UNIVERSIDAD DE SONORA

DIVISIÓN DE CIENCIAS EXACTAS Y NATURALES

DEPARTAMENTO DE FÍSICA

LICENCIATURA EN FÍSICA

FÍSICA COMPUTACIONAL I

Reconstruyendo la señal

Martinez López Lizbeth Vanessa

Profesor del curso
Dr. Carlos Lizárraga Celaya

26 Abril del 2017

Resumen

En la práctica anterior hicimos una transformada de Fourier para las mareas en las costas de Atlantic City y Cabo San Lucas, de esta manera podíamos hacer un análisis armónico y ver el comportamiento de dichas mareas. En esta actividad se pretende partir de la información proporcionada por la transformada de Fourier, para así reconstruir la señal y obtener una gráfica similar a las obtenidas en la actividad 5.

1. Introducción

En esta práctica seguimos trabajando con la transformada de Fourier realizada en la práctica anterior, para así poder hacer una reconstrucción de la señal. Como ya sabemos, las series de Fourier describen señales periódicas como una combinación de señales armónicas (sinusoides). En este caso, la transformada de Fourier tuvo como función descomponer la señal de las amplitudes de las mareas, en un intervalo de tiempo; con la transformada obtuvimos los modos y logramos extraer la información de las amplitudes de las frecuencias de cada tipo de marea. La transformada de Fourier nos permite aproximar las señales de las mareas por medio de senos y/o cosenos, los valores que nos proporciona la transformada son los que nos permitirán reconstruir la señal.

La reconstrucción de la señal de la marea debe darnos como resultado, gráficas muy similares a las obtenidas en la actividad 5, con 'similares' nos referimos a que tendrán un margen de error, pues los valores de amplitud y frecuencias son aproximaciones a los valores reales. A continuación se habla del desarrollo para llevar a cabo esta práctica.

2. Desarrollo

Así como en las actividades 5 y 6, volvemos a hacer uso de los archivos previamente descargados (del sitio web de la NOAA y CICESE) y preparados, los cuales tienen la información del nivel del mar y las fechas en las que el mar contaba con tal amplitud, que es lo que necesitamos principalmente.

Como ya se mencionó, la transformada de Fourier permite aproximar las señales de las mareas por medio de senos y/o cosenos (además de otros parámetros necesarios), es por ello que para la reconstrucción de la señal se hace uso de la siguiente ecuación:

$$S_N(t) = \frac{A_0}{2} + \sum_{n=1}^{N} A_n \cdot \cos\left(2\pi n f t + \phi_n\right) \tag{1}$$

Dicha ecuación depende de la frecuencia, pues con la transformada obtenemos los valores de ésta. Aquí vemos que también depende de la amplitud y la fase. La razón por la que se usa coseno, es porque en esta práctica se observó que la mejor aproximación de mareas era con dicha función.

2.1. Atlantic City, New Jersey

Para la reconstruccción de la señal de mareas en Atlantic City, primero inicié a obtener la transformada de Fourier, empleando el código de la actividad pasada, es decir, se importaron las bibliotecas necesarias, se hizo lectura del archivo, se definieron las columnas de la tabla, se procedió con la graficación, usando el siguiente código:

```
from scipy.fftpack import fft, fftfreq, fftshift
# number of signal points
N = 744
# sample spacing
```

```
T = 1
x = df['Date']
y = df['Water Level']
yf = fft(y)
xf = fftfreq(N, T)
xf = fftshift(xf)
yplot = fftshift(yf)
import matplotlib.pyplot as plt
plt.plot(xf, 1.0/N * np.abs(yplot))
plt.xlim(0, 0.1)
fig=plt.gcf()
fig.set_size_inches(7,7)

plt.xlabel("Frecuencia (1/hr)")
plt.ylabel("Amplitud (m)")



plt.grid()
plt.show()
```

Lo adicional a este código es lo siguiente:

```
v = np.arange(0.0, 744.0, 1.0)

df['T'] =  pd.Series(v, index =None)
```

Esto nos permite agregar otra columna en los datos, las cuales son de horas transcurridas, en este caso los datos se tomaron cada hora. Mientras que df['T'] nos permitirá evaluar la función que utilizaremos para la reconstrucción de la señal, pues la ecua-

ción que usaremos para reconstruir la señal es dependiente del tiempo transcurrido, es por ello que empleamos este código, pues si no lo consideramos, la aproximación de las mareas no será buena.

Con esto definido, seguimos con el código. Para reconstruir la señal, necesitamos obtener los parámetros necesarios, con el código anterior se obtiene el tiempo, ahora faltan las amplitudes, frecuencias y fases, para ello hacemos lo siguiente:

```
a = 2*np.absolute(yf)/N


print(np.where(a[:,]>0.03))
b= a[a[:,]>0.03]
b
```

Con ésto obtenemos los picos de la transformada, lo cual es necesario para el cálculo de los parámetros ya mencionados. El 0.03 es un aproximado de la amplitud del pico más pequeño, la condición es que me proporcione todos los picos que cumplen con amplitudes mayores a 0.03 m.

Ya con el arreglo de picos obtenido, seleccionamos los que están en el lado positivo de la transformada y calculamos los parámetros con el siguiente código:

```
#Amplitud
A0 = np.absolute(yf[int(0),]/N)
A1= 2*np.absolute(yf[1,]/N)
A2= 2*np.absolute(yf[2,]/N)
A3= 2*np.absolute(yf[3,]/N)
A4= 2*np.absolute(yf[4,]/N)
A5= 2*np.absolute(yf[5,]/N)
A6= 2*np.absolute(yf[6,]/N)
A7= 2*np.absolute(yf[7,]/N)
```

```
A8= 2*np.absolute(yf[8,]/N)

A9= 2*np.absolute(yf[9,]/N)

A10= 2*np.absolute(yf[11,]/N)

A11= 2*np.absolute(yf[12,]/N)

A12= 2*np.absolute(yf[14,]/N)

A13= 2*np.absolute(yf[28,]/N)

A14= 2*np.absolute(yf[29,]/N)

A15= 2*np.absolute(yf[31,]/N)

A16= 2*np.absolute(yf[58,]/N)

A17= 2*np.absolute(yf[59,]/N)

A18= 2*np.absolute(yf[60,]/N)

A19= 2*np.absolute(yf[61,]/N)

A20= 2*np.absolute(yf[62,]/N)


#Frecuencia

f1= xf[int(N/2 +1),]

f2= xf[int(N/2 +2),]

f3= xf[int(N/2 +3),]

f4= xf[int(N/2 +4),]

f5= xf[int(N/2 +5),]

f6= xf[int(N/2 +6),]

f7= xf[int(N/2 +7),]

f8= xf[int(N/2 +8),]

f9= xf[int(N/2 +9),]

f10= xf[int(N/2 +11),]

f11= xf[int(N/2 +12),]

f12= xf[int(N/2 +14),]

f13= xf[int(N/2 +28),]
```

```
f14= xf[int(N/2 +29),]

f15= xf[int(N/2 +31),]

f16= xf[int(N/2 +58),]

f17= xf[int(N/2 +59),]

f18= xf[int(N/2 + 60),]

f19= xf[int(N/2 + 61),]

f20= xf[int(N/2 + 62),]



#Fases

O1= np.angle(yf[1,])

O2= np.angle(yf[2,])

O3= np.angle(yf[3,])

O4= np.angle(yf[4,])

O5= np.angle(yf[5,])

O6= np.angle(yf[6,])

O7= np.angle(yf[7,])

O8= np.angle(yf[8,])

O9= np.angle(yf[9,])

O10= np.angle(yf[11,])

O11= np.angle(yf[12,])

O12= np.angle(yf[14,])

O13= np.angle(yf[28,])

O14= np.angle(yf[29,])

O15= np.angle(yf[31,])

O16= np.angle(yf[58,])

O17= np.angle(yf[59,])

O18= np.angle(yf[60,])
```

```
O19= np.angle(yf[61,])
O20= np.angle(yf[62,])
```

Con esto calculado, se procede a hacer la aproximación, utilizando la ecuación mencionada anteriormente:

```
y= df['Water Level']
w= 2.0*np.pi
a=0
def f(t):
    return A0+ (A1*np.cos(w*f1*t+O1) + A2*np.cos(w*f2 *t+O2)
                    + A3*np.cos(w*f3*t+O3) + A4*np.cos(w*f4*t + O4)
                    + A5*np.cos(w*f5*t+O5) + A6*np.cos(w*6*t + O6)
                    + A7*np.cos(w*f7*t+O7) + A8*np.cos(w*f8*t+ O8)
                    + A9*np.cos(w*f9*t+ O9) + A10*np.cos(w*f10*t+ O10)
                    + A11*np.cos(w*f11*t+ O11) + A12*np.cos(w*f12*t+ O12)
                    + A13*np.cos(w*f13*t+ O13) + A14*np.cos(w*f14*t+ O14)
                    + A15*np.cos(w*f15*t+ O15) + A16*np.cos(w*f16*t+ O16)
                    + A17*np.cos(w*f17*t+ O17) + A18*np.cos(w*f18*t+ O18) +
                    A19*np.cos(w*f12*t+ O19) + A20*np.cos(w*f20*t+ O20) )
```

Ya habiendo hecho la aproximación, procedemos a graficar por medio del siguiente código:

```
plt.plot(df[u'Date'], df[u'Water Level'], 'k', label ="Altura")
plt.plot(df['Date'], f(df['T']), 'b-', label='Altura reconstruida')
plt.ylabel('Altura de marea [m]')
plt.xlabel('Fecha')
plt.title('Marea en Atlantic City, NJ. Enero 2017')
plt.grid(True)
```

```
locs, labels = plt.xticks()
plt.setp(labels, rotation=45)


fig = plt.gcf()
fig.set_size_inches(10, 6)
plt.show()
```

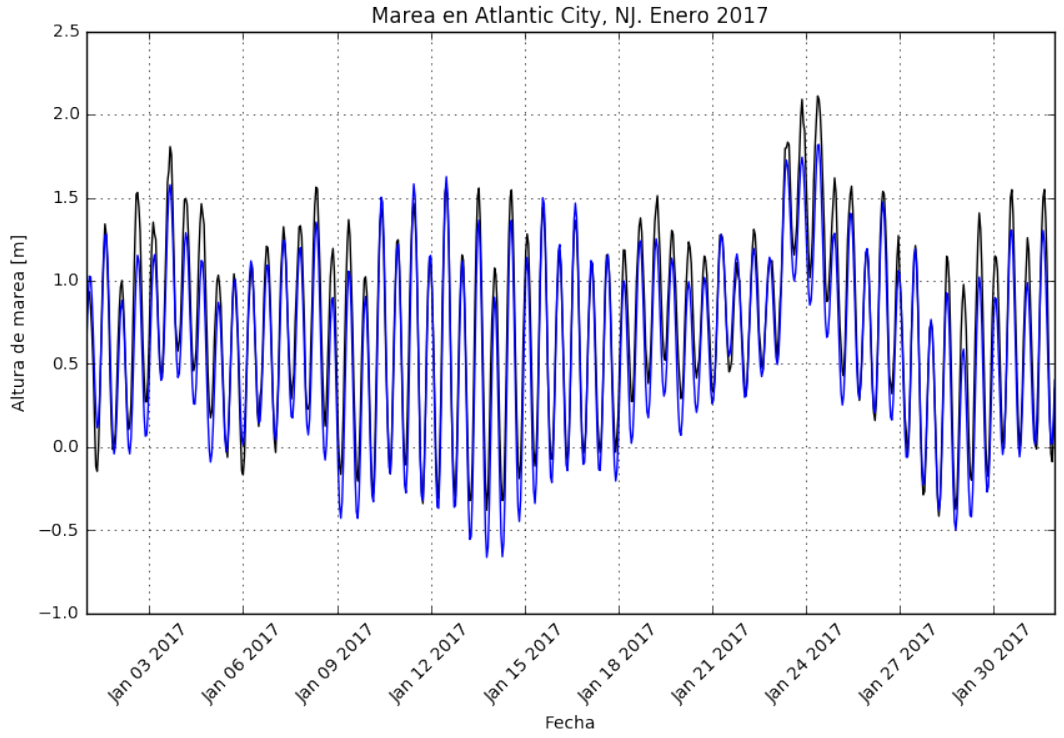
Aquí es donde usamos la evaluación de la función respecto al tiempo.

2.2. Cabo San Lucas, Baja California Sur

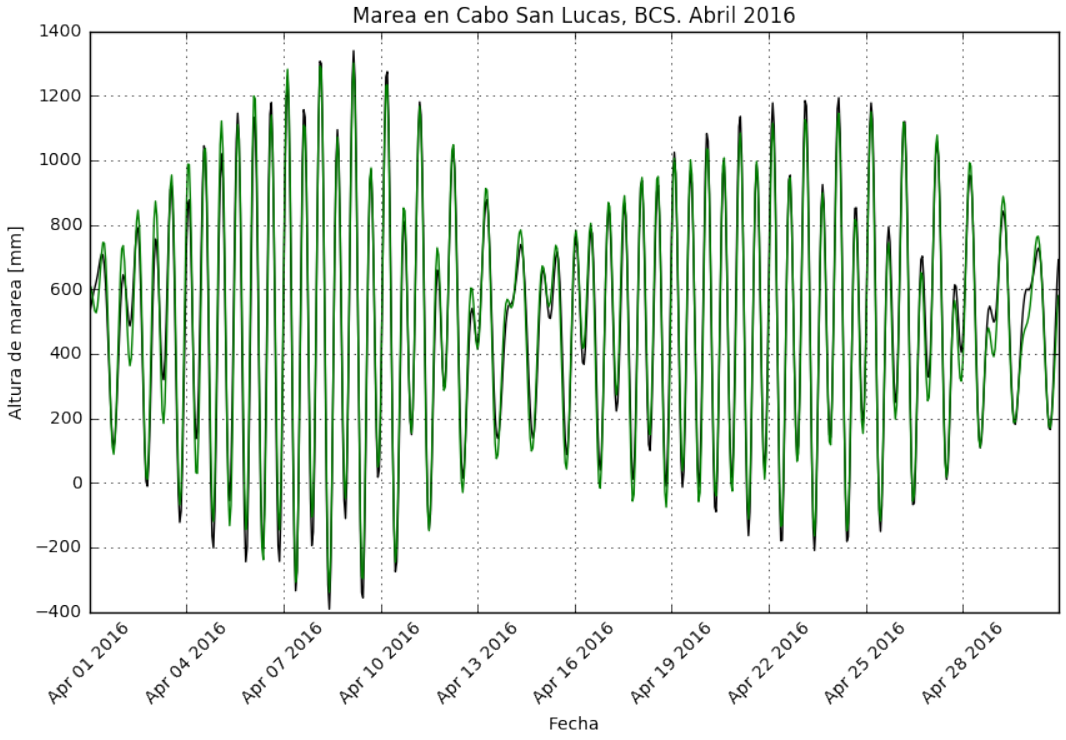
Para la reconstrucción de la señal en Cabo San Lucas, se hizo exactamente le mismo procedimiento que para Atlantic City, lo diferente en éste caso es el hecho de tener que juntar las columnas de Año, Mes, Día y Hora para crear una de "Fecha", en éste caso se encontraron menos picos.

3. Resultados

Reconstrucción de señal en Atlantic City



Reconstrucción de señal en Cabo San Lucas



Podemos observar que en ambos casos la reconstrucción de la señal es bastante aproximada, pues la gráfica obtenida en esta práctica es muy parecida a la obtenida en la actividad 5, podemos verla en las imágenes al ser superpuestas. El error obtenido en ambos casos fue de 0.04 para Atlantic City y 0.006 en Cabo San Lucas.

4. Conclusión

Como hemos visto, la transformada de Fourier es una excelente herramienta que nos ayuda a la aproximación de señales periódicas, en este caso fue de gran utilidad para reconstruir la señal de las mareas en las costas seleccionadas. Con ésto, vemos que se pueden hacer predicciones del comportamiento de las mareas.

En la práctica hemos observado que la reconstrucción de la señal fue bastante buena, siendo la de Cabo San Lucas la más aproximada.

5. Referencias

[1] $http://www4.tecnun.es/asignaturas/tratamiento\%20digital/tema3.pdf$

[2] $https://docs.scipy.org/doc/scipy-0,18,1/reference/tutorial/fftpack.html$

[3] $https://tidesandcurrents.noaa.gov/stations.html?type=Water+Levels\%29$

[4] $http://predmar.cicese.mx/$