UNIVERSIDAD DE SONORA
DIVISIÓN DE CIENCIAS EXACTAS Y NATURALES
DEPARTAMENTO DE FÍSICA
LICENCIATURA EN FÍSICA

Actividad 7
Reconstrucción de las mareas

Jesús Ernesto Torres Burruel

25 de abril de 2017

Física Computacional I
Profr. Carlos Lizárraga Celaya

Reconstrucción de las mareas

Hemos visto que el análisis armónico de Fourier aplicado a las mareas genera resultados que ayudan a aproximar las ondas o señales mediante combinación de senos y cosenos con diferentes amplitudes y fases. En la actividad 6 de este curso hicimos un análisis de Fourier para encontrar las componentes principales de las mareas registradas por las estaciones de El Sauzal, Ensenada, B.C, Méx.y por la estación de Monterey, CA., EE.UU.

La transformada de fourier nos devuelve el valor de las amplitudes y las fases asociadas a funciones de onda, las cuales tienen una frecuencia específica, que también es obtenida a partir del análisis de Fourier.

En esta práctica se elaboró una recreación de la marea gracias a un análisis armónico previo utilizando transformada de Fourier con ayuda del paquete fftpack de la biblioteca textsfSciPi de python. Con esto obtendremos una recreación de las principales de las mareas que se presentan en las regiones de El Sauzal, Ensenada, B.C., México y Monetrey, CA., EE. UU., a partir de los datos obtenidos de sus correspondientes estaciones, que pueden obtenerse desde las páginas web de CICESE (Centro de Investigación Científica y de Educación Superior de Ensenada, `http://predmar.cicese.mx/calendarios`) y CO-OPS de la NOAA (Administración Nacional Oceánica y Atmosférica, `https://tidesandcurrents.noaa.gov/`).[1]

1. Descripción de la marea recreada

La transformada de Fourier descompone una función que depende del tiempo en una función de frecuencias que generan a la función. Las series de Fourier son una forma de representar una función de tipo ondulatorio como la suma de funciones oscilatorias de senos y cosenos.

Una onda como las que estudiamos en las mareas son el producto de la superposición de ondas son funciones de senos y cosenos con con sus respectivas amplitudes de onda amplitud de onda del tipo:

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

[1]Los códigos utilizados en esta práctica fueron elaborados en lenguaje Python.

O bien como

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

Consideremos que se tiene una de la forma de 1

$$= \frac{a_0}{2} + \sum_{n=1}^{N} \left(A_n \sin(\phi_n) \cos\left(\frac{2\pi n}{P}\right) + A_n \cos(\phi_n) \cdot \sin\left(\frac{2\pi nt}{P}\right) \right)$$

$$= \frac{a_0}{2} + \sum_{n=1}^{N} \left(a_n \cos\left(\frac{2\pi n}{P}\right) + b_n \cdot \sin\left(\frac{2\pi nt}{P}\right) \right)$$

Donde $A_0/2$ es el valor medio en el cual se presenta la onda, en nuestro caso es el nivel del mar promedio de la región estudiad, t es el tiempo y P es el periodo en el cual se presenta la onda.

Al aplicar la transformada de Fourier a nuestros datos, se obtendrá una función en términos de la frecuencia de las ondas, en lugar del tiempo. Para nuestra función se obtendría lo siguiente.

$$S_N(t) = \frac{a_0}{2} + \sum_{n=1}^{N} \left(A_n \cos(\phi_n) \cdot \sin\left(\frac{2\pi nt}{P}\right) + A_n \sin(\phi_n) \cos\left(\frac{2\pi n}{P}\right) \right)$$

$$S_N = \sum_{n=-N}^{N} c_n \cdot e^{i\frac{2\pi nx}{P}} \tag{3}$$

Donde

$$c_n = \begin{cases} \frac{A_n}{2i} e^{\phi_n} = \frac{1}{2}(a_n) - ib_n) & n > 0 \\ \frac{1}{2} A_0 = \frac{1}{2} a_0 & n = 0 \\ c^*_{|n|} & n < 0 \end{cases}$$

Y la relación inversa de estas expresiones son:

$$A_n = \sqrt{a_n^2 + b_n^2} \quad \phi_n = \arctan\left(\frac{a_n}{b_n}\right)$$

1.1. Amplitudes encontradas

En la actividad 6 se hizo uso de la transformada de Fourier para encontrar los armónicos que componen a las mareas que se presentan en cada región, en este caso utilizaremos esos resultados (añadiendo algunos valores que no se

consideraron en dicha actividad ya que no pudieron ser catalogados dentro de las mareas comunes). Con esto se encontró que los armónicos más influyentes en las mareas fueron 12 para El Sauzal y 11 para Monterey. Estos valores son los que se asignan a cada A_n de la ecuación 4.

El método utilizado para encontrar los valores de amplitudes y frecuencias asociadas en esta actividad se pueden consultar en la actividad 6 *Análisis armónico de mareas* [1].

2. Reconstrucción de las mareas

Para reconstruir las mareas primero se deben de poseer los archivos de datos correspondientes a los meses con las fechas y cantidad de datos que se trabajó en la actividad 6 de las regiones de El Sauzal, Ensenada, B.C., México y Monetrey, CA., EE. UU., que pueden obtenerse desde las páginas web de CICESE (Centro de Investigación Científica y de Educación Superior de Ensenada, `http://predmar.cicese.mx/calendarios`) y CO-OPS de la NOAA (Administración Nacional Oceánica y Atmosférica, `https://tidesandcurrents.noaa.gov/`).

Con estos datos podemos trabajar en nuestra aproximación a las mareas presentadas en estas regiones. Para poder construir una función del tipo de la ecuación 1, se deben de conocer primero los parámetros a partir de los cuales se manejará esta función.

2.1. Amplitudes, frecuencias y fases

Para no tener problemas se repetirán los códigos de la transformada de Fourier a los datos, solo que en este caso ya usaremos los resultados obtenidos de la actividad anterior para los nodos encontrados; el fin de repetir la transformada de fourier es con el de objetivo de tener los valores exactos de amplitud, frecuencia y fase para poder reescribir la ecuación 1 para la reproducción de la marea presentada.

2.1.1. Encontrando los parámetros

El siguiente código es el utilizado para asignar las amplitudes que manejaremos en la ecuación:

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

Para comenzar, primero se hace la lectura de los paquetes que serán utilizados y los datos.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from datetime import datetime

df_s = pd.read_csv('szl_data.csv', names = ("Anio", "Mes", "dia",
"Hora", "Altura" ), header = 0) #La altura está en mm
df_m = pd.read_csv('Mtry_CA_hr.csv', names = ("Date Time",
"Water Level","Sigma", "I", "L"), header = 0)
```

Donde `df_s` son los datos de El Sauzal y `df_m` los de Monterey. Como estos datos contienen fechas y horas asignadas, aprovecharemos esto para convertir estos datos en un formato de fechas:

```
df_s['date'] = df_s.apply(lambda x:datetime.strptime("{0} {1} {2}
{3}".format(x[u'Anio'],x[u'Mes'], x[u'dia'], x[u'Hora']), "%Y %m
    %d %H"),axis=1)
df_m["Date Time"]= pd.to_datetime(df_m["Date Time"], format =
'%Y %m %d %H:%M:')
```

Para los datos de El sauzal se debe seleccionar el rango de fechas con el que se quiere trabajar, pues los datos proporcionados por el CICESE son de todo un año, pero solo deseamos trabajar con tres meses (Ene–Mar de 2016). Para esto se usa el siguiente código:

```
szl = df_s[(df_s['date'] >= '2016-01-01 00:00:00') &
(df_s['date'] <= '2016-03-31 23:00:00')]
mty = df_m
```

Ahora nuestros datos de El Sauzal son `szl` y los datos de Monterey `mty` , para estos últimos datos no se seleccionó un rango de fechas pues se puede seleccionar previamente desde la página de la NOAA (se trabajaran los datos de Dic 2016 a Feb 2017, medidos en metros y cada hora).

Lo que sigue ahora es aplicar la transformada de Fourier y con ello poder asignar los valores a los parámetros que utilizaremos para crear una función como 1.

Parámetros para El Sauzal

```
#Transformada de Fourier para datos de El Sauzal
from scipy.fftpack import fft, fftfreq, fftshift
import numpy as np
# number of points
N = 2184
# sample spacing
T = 1.0
y = szl["Altura"]/1000.0
yf = fft(y)
xf = fftfreq(N, T)
xf = fftshift(xf)
yplot = fftshift(yf)

#Amplitudes notorias, fueron ubicadas en la actividad 6
A0_s = np.absolute(yf[int(0),]/N)
A1_s = 2.0*np.absolute(yf[int(84),]/N)
A2_s = 2.0*np.absolute(yf[int(85),]/N)
A3_s= 2.0*np.absolute(yf[int(90),]/N)
A4_s = 2.0*np.absolute(yf[int(91),]/N)
A5_s = 2.0*np.absolute(yf[int(92),]/N)
A6_s = 2.0*np.absolute(yf[int(172),]/N)
A7_s = 2.0*np.absolute(yf[int(173),]/N)
A8_s = 2.0*np.absolute(yf[int(174),]/N)
A9_s = 2.0*np.absolute(yf[int(175),]/N)
A10_s =  2.0*np.absolute(yf[int(176),]/N)
A11_s = 2.0*np.absolute(yf[int(177),]/N)
A12_s = 2.0*np.absolute(yf[int(182),]/N)

#Frecuencias que se presentan en el Sauzal
f_A1s = xf[int(N/2 +84)]
f_A2s = xf[int(N/2 +85),]
f_A3s =  xf[int(N/2 +90),]
f_A4s =  xf[int(N/2 +91),]
```

```
f_A5s =  xf[int(N/2 +92),]
f_A6s =  xf[int(N/2 +172),]
f_A7s =  xf[int(N/2 +173),]
f_A8s =  xf[int(N/2 +174),]
f_A9s =  xf[int(N/2 +175),]
f_A10s =  xf[int(N/2 +176),]
f_A11s =  xf[int(N/2 +177),]
f_A12s =  xf[int(N/2 +182),]

# Fases que se presentan en el Sauzal
qA0s = np.angle(yf[int(0),])
qA1s = np.angle(yf[int(84),])
qA2s = np.angle(yf[int(85),])
qA3s = np.angle(yf[int(90),])
qA4s = np.angle(yf[int(91),])
qA5s = np.angle(yf[int(92),])
qA6s =  np.angle(yf[int(172),])
qA7s =  np.angle(yf[int(173),])
qA8s =  np.angle(yf[int(174),])
qA9s =  np.angle(yf[int(175),])
qA10s =  np.angle(yf[int(176),])
qA11s =  np.angle(yf[int(177),])
qA12s =  np.angle(yf[int(182),])
```

Parámetros para Monterey

```
#Transformada de Fourier para datos de Monterey
from scipy.fftpack import fft, fftfreq, fftshift
import numpy as np
# number of points
N_d = 2160
# sample spacing
T_d = 1.0
y_hr = mty['Water Level']
yf_hr = fft(y_hr)
xf_hr = fftfreq(N_d, T_d)
xf_hr = fftshift(xf_hr)
yplot_hr = fftshift(yf_hr)
```

```
#Amplitudes notorias, fueron ubicadas en la actividad 6
A0_m = np.absolute(yf_hr[0,]/N_d)
A1_m = 2.0*np.absolute(yf_hr[6,]/N_d)
A2_m = 2.0*np.absolute(yf_hr[7,]/N_d)
A3_m = 2.0*np.absolute(yf_hr[83,]/N_d)
A4_m = 2.0*np.absolute(yf_hr[84,]/N_d)
A5_m = 2.0*np.absolute(yf_hr[90,]/N_d)
A6_m = 2.0*np.absolute(yf_hr[91,]/N_d)
A7_m = 2.0*np.absolute(yf_hr[170,]/N_d)
A8_m = 2.0*np.absolute(yf_hr[171,]/N_d)
A9_m = 2.0*np.absolute(yf_hr[173,]/N_d)
A10_m = 2.0*np.absolute(yf_hr[174,]/N_d)
A11_m = 2.0*np.absolute(yf_hr[180,]/N_d)

#Frecuencias que se presentan en el Monterey
f_A1m = xf_hr[int(N_d/2 +6),]
f_A2m = xf_hr[int(N_d/2 +7),]
f_A3m = xf_hr[int(N_d/2 +83),]
f_A4m = xf_hr[int(N_d/2 +84),]
f_A5m = xf_hr[int(N_d/2 +90),]
f_A6m = xf_hr[int(N_d/2 +91),]
f_A7m = xf_hr[int(N_d/2 +170),]
f_A8m = xf_hr[int(N_d/2 +171),]
f_A9m = xf_hr[int(N_d/2 +173),]
f_A10m = xf_hr[int(N_d/2 +174),]
f_A11m = xf_hr[int(N_d/2 +180),]

#fases de cada componente encontrada en Monterey
qA0m = np.angle(yf_hr[0,])
qA1m = np.angle(yf_hr[6,])
qA2m = np.angle(yf_hr[7,])
qA3m = np.angle(yf_hr[83,])
qA4m = np.angle(yf_hr[84,])
qA5m = np.angle(yf_hr[90,])
qA6m = np.angle(yf_hr[91,])
qA7m = np.angle(yf_hr[170,])
qA8m = np.angle(yf_hr[171,])
qA9m = np.angle(yf_hr[173,])
```

```
qA10m = np.angle(yf_hr[174,])
qA11m = np.angle(yf_hr[180,])
```

Con estos parámetros ya seremos capaces de construir una función como la ecuación 1 o 2:

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

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

Como encontramos el valor de la frecuencia, utilizando la relación $P = \frac{1}{f}$:

$$S_N(t) = \frac{A_0}{2} + \sum_{n=1}^{N} A_n \cdot \sin\left(2\pi nft + \phi_n\right) \tag{4}$$

Para la serie de cosenos:

$$S_N(t) = \frac{A_0}{2} + \sum_{n=1}^{N} A_n \cdot \cos\left(\frac{2\pi nt}{P} + \phi_n\right) \tag{5}$$

2.2. El tiempo como variable

Sabemos que nuestra función es dependiente del tiempo que transcurre, por esto debemos de tener en cuenta esta variable para poder hacer nuestra aproximación de la marea a partir de la función $S_N(t)$

Para poder analizar nuestra función se definirá una nueva columna, la cual serán las horas transcurridas, desde las 0.0 horas transcurridas hasta las horas transcurridas para nuestro número de datos, los cuales se realizaron cada hora. El siguiente código es para agregar a nuestros datos este otro término, que nos servirá para evaluar la función que utilizaremos para la marea.

```
z = np.arange(0.0, 2184.0, 1.0)
k = np.arange(0.0, 2160.0, 1.0)

szl['T'] =  pd.Series(z, index =None)
mty['T'] = pd.Series(k, index=None)
```

Ahora hay una columna que contiene los valores del tiempo con los que trabajaremos para nuestra aproximación. Cada conjunto de datos tiene diferente cantidad de mediciones, por eso es que los datos de El Sauzal terminan en tiempo diferente al de Monterey.

2.3. Construcción de la función para la marea

Hemos encontrado ya todos los parámetros bajo los cuales se define nuestra función que aproximará la marea presentada en Monterey y El Sauzal. Para saber si nuestra función es del tipo de 1 o del tipo de 2 debemos observar que hay un valor para la amplitud media del mar en la frecuencia con valor cero y fase cero, para que exista esta (A_0 en realidad es también parte de la serie de cosenos y senos), debido a los desfaces que se presentan al tratar de hacer una evaluación con la serie de senos, se utilizará la ecuación 2 para aproximar los valores de la amplitud y eliminar este desface.

EL siguiente código es la función con la que a partir de los parámetros de las amplitudes, frecuencias y fases se utilizará para reconstruir las mareas de la región correspondiente, para obtener los valores de las amplitudes se debe evaluar en el tiempo transcurrido que corresponde cada medición con respecto al primer dato (columna de tiempo en los datos).

```
#Aproximacion a la marea de El Sauzal
y= szl['Altura']/1000
w= 2.0*np.pi
a=0
def f(t):
    return A0_s + (A1_s*np.cos(w*f_A1s*t+qA1s) + A2_s*np.cos(w*f_A2s *t+qA2s)
+ A3_s*np.cos(w*f_A3s*t+qA3s) + A4_s*np.cos(w*f_A4s*t + qA4s)
+ A5_s*np.cos(w*f_A5s*t+qA5s) + A6_s*np.cos(w*f_A6s*t + qA6s)
+ A7_s*np.cos(w*f_A7s*t+qA7s) + A8_s*np.cos(w*f_A8s*t+ qA8s)
+ A9_s*np.cos(w*f_A9s*t+ qA9s) + A10_s*np.cos(w*f_A10s*t+ qA10s)
+ A11_s*np.cos(w*f_A11s*t+ qA11s) + A12_s*np.cos(w*f_A12s*t+ qA12s))

#Aproximacion para la marea en Monterey
w= 2.0*np.pi
def g(t):
    return A0_m    + (A1_m*np.cos(w*f_A1m*t+qA1m) + A2_m*np.cos(w*f_A2m *t+qA2m)
+ A3_m*np.cos(w*f_A3m*t+qA3m) + A4_m*np.cos(w*f_A4m*t+qA4m)
```

```
+ A5_m*np.cos(w*f_A5m*t+qA5m) + A6_m*np.cos(w*f_A6m*t+qA6m)
+ A7_m*np.cos(w*f_A7m *t+qA7m) + A8_m*np.cos(w*f_A8m*t+qA8m)
+ A9_m*np.cos(w*f_A9m*t+qA9m) + A5_m*np.cos(w*f_A10m*t+qA10m)
+ A5_m*np.cos(w*f_A10m*t+qA10m))
```

2.4. Obtención gráfica de la marea reconstruida y la real

Una vez que ya construimos nuestra serie o función de cosenos de la cuál proviene nuestra marea, lo que resta es trabajar las gráficas que representan a las mareas. Con esto podemos observar cómo fue el comportamiento de nuestra aproximación con respecto a los valores reales obtenidos.

Con los siguientes códigos se realiza la gráfica de la marea real medida y la reconstruida por nosotros para cada región.

```
#Gráfica de la aproximación de la marea en El Sauzal
import matplotlib.pyplot as plt
#Marea real registrada
plt.plot(szl['date'], szl['Altura']/1000, 'b-',
label ="Altura registrada")

#Marea reconstruida
plt.plot(szl['date'], f(szl['T']), 'r-', label='Altura recosntruida')



plt.xlim(pd.Timestamp("2016-01-01 00:00:00"),
pd.Timestamp('2016-03-31 23:00:00'))
plt.ylabel('Altura de marea [m]', fontsize = 13)
plt.xlabel('Día', fontsize = 13)
plt.title('Marea registrada por la estación en El Sauzal, Ensenada
durante Enero-Marzo 2016', fontsize= 19)
plt.legend()
plt.grid(True)

fig = plt.gcf()

fig.set_size_inches(20, 10)
plt.show()
```

```
#gráfica de la aproximación de la marea en Monterey
import matplotlibtlib.pyplot as plt
#Marea reconstruida
plt.plot(mty['Date Time'], g(mty['T']), 'y-',
label='Altura recosntruida')

#Marea real registrada
plt.plot(mty[u'Date Time'], mty[u'Water Level'], 'g',
label ="Altura Real")

plt.xlim(pd.Timestamp('2016-12-01 00:00:00'),
pd.Timestamp('2017-02-28 00:23:00'))
plt.ylabel('Altura de marea [m]', fontsize = 12)
plt.xlabel('Día', fontsize=12)
plt.title('Marea en Monterey, CA,  durante Diciembre 2016 -
Febrero 2017', fontsize = 19)
plt.legend()
plt.grid(True)
plt.grid(True)

fig = plt.gcf()
fig.set_size_inches(20, 8)
plt.show()
```

Con esto obtenemos las siguientes gráficas:

Como se puede observar en las gráficas obtenidas 1 y 2 se tiene una muy buena aproximación a la marea real registrada por parte de la marea reconstruida a partir del análisis armónico de los datos.

Para apreciar mejor lo obtenido con la aproximación se puede graficar una semana:

```
#Gráfica aumentada de la aproximación de la marea en El Sauzal
import matplotlib.pyplot as plt

#Marea reconstruida
plt.plot(szl['date'], f(szl['T']), 'r-', label='Altura recosntruida')
```

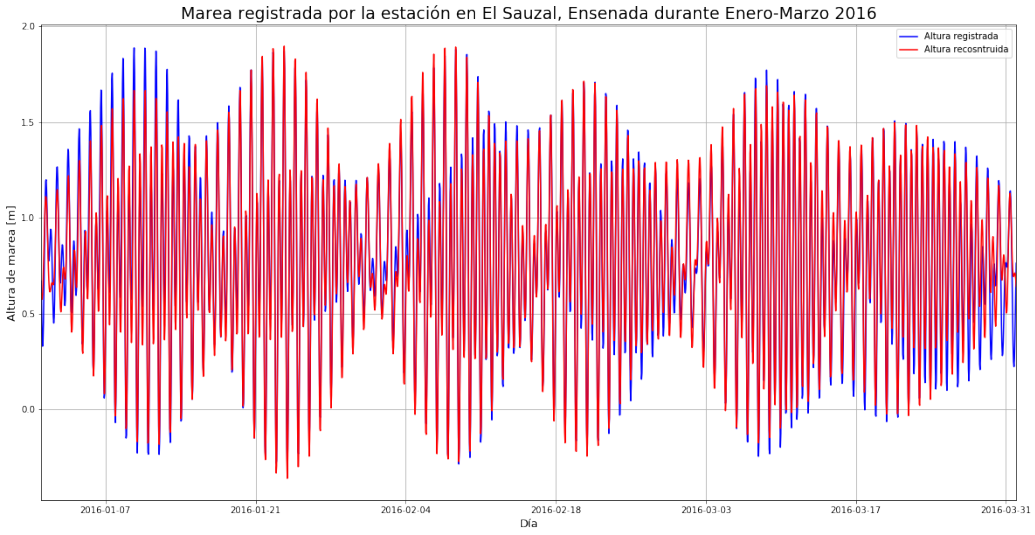

Figura 1: Gráfica obtenida al tener la marea reconstruida (color rojo) y la marea que en realidad se presentó (color azul) en El Sauzal, Ensenada, B.C.

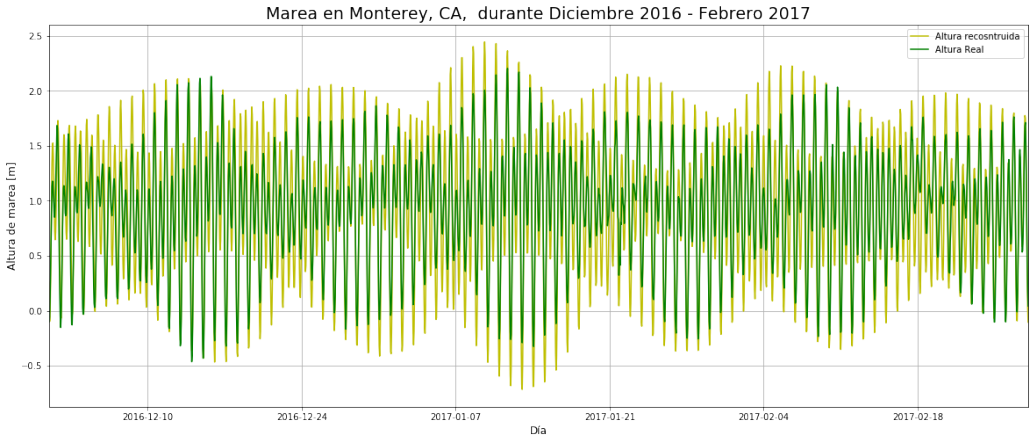

Figura 2: Gráfica obtenida al tener la marea reconstruida (color amarillo) y la marea que en realidad se presentó (color verde) en Monterey, CA.

```
#Marea real registrada
plt.plot(szl['date'], szl['Altura']/1000, 'b-', label ="Altura registrada")
```

```
plt.xlim(pd.Timestamp("2016-01-01 00:00:00"),
pd.Timestamp('2016-01-07 23:00:00'))
plt.ylabel('Altura de marea [m]', fontsize = 13)
plt.xlabel('Día', fontsize = 13)
plt.title('Marea registrada por la estación en El Sauzal, Ensenada
durante Enero-Marzo 2016', fontsize= 19)
plt.legend()
plt.grid(True)

fig = plt.gcf()

fig.set_size_inches(20, 10)
plt.show()


#gráfica aumentada de la aproximación de la marea en Monterey
import matplotlibtlib.pyplot as plt
#Marea reconstruida
plt.plot(mty['Date Time'], g(mty['T']), 'y-',
label='Altura recosntruida')

#Marea real registrada
plt.plot(mty[u'Date Time'], mty[u'Water Level'], 'g',
label ="Altura Real")

plt.xlim(pd.Timestamp('2017-01-01 00:00:00'),
pd.Timestamp('2017-01-07 00:23:00'))
plt.ylabel('Altura de marea [m]', fontsize = 12)
plt.xlabel('Día', fontsize=12)
plt.title('Marea en Monterey, CA,  durante Diciembre 2016 - Febrero
2017', fontsize = 19)
plt.legend()
plt.grid(True)
plt.grid(True)

fig = plt.gcf()
fig.set_size_inches(20, 8)
plt.show()
```

Con esto se obtienen gráficas en las cuales se aprecia mejor cómo se comporta la reconstrucción de la marea con respecto a la registrada en los datos de las estaciones.

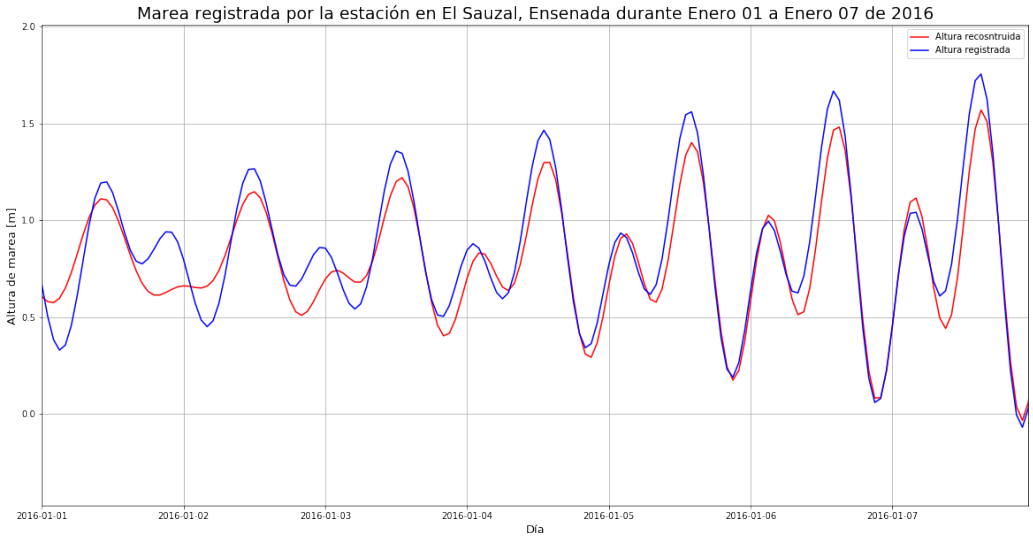

Figura 3: Gráfica obtenida para una semana sobre la marea reconstruida (color rojo) y la marea que en realidad se presentó (color azul) en Monterey, CA.

Como se aprecia, en las gráficas 3 y 4 los valores aproximados y los reales para las mareas presentan resultados bastante similares entre sí. Esto indica que nuestra reconstrucción da un buen resultado. Pero para estar más seguros se debe de considerar el error de las mediciones.

2.5. Error en la reconstrucción

Para encontrar el error dado entre la marea reconstruida y su valor real se hará un análisis del error relativo usando las diferencias cuadradas de la aproximación y el valor verdadero para cada medición.

El siguiente código es el utilizado para encontrar los errores relativos para cada lugar:

```
#Error relativo para El Sauzal
y1 = f(szl['T']) #reconstrucción
y = szl['Altura']/1000 #valor real
```

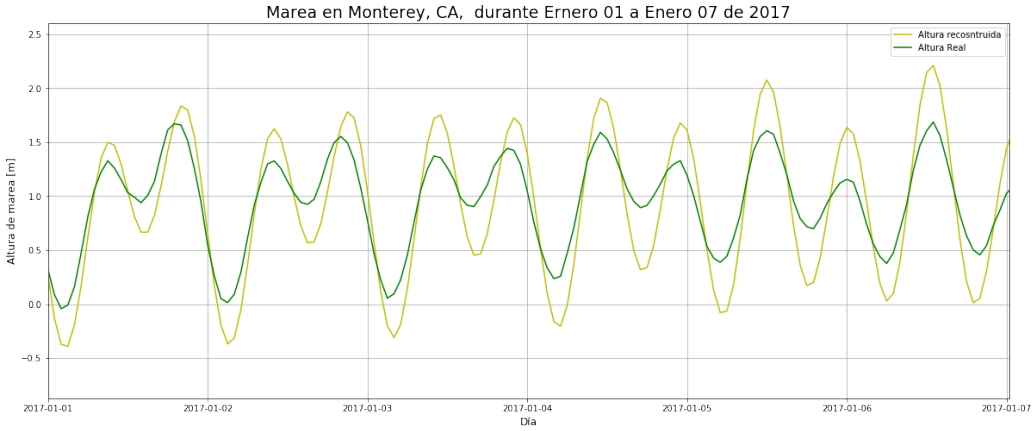

Figura 4: Gráfica obtenida para una semana de la marea reconstruida (color amarillo) y la marea que en realidad se presentó (color verde) en Monterey, CA.

```
err_s = np.sum(np.abs(y1-y)**2)/np.sum(np.abs(y)**2),
err_s

#Error relativo para Monterey
y_m = mty['Water Level'] #valor real
y1_m = g(mty['T']) #reconstrucción

err_s = np.sum(np.abs(y1_m-y_m)**2)/np.sum(np.abs(y_m)**2),
err_s
```

Con este código se obtiene que los errores para las aproximaciones de cada lugar fueron:

El Sauzal: 0.011133759857637149

Monterey: 0.05863425834523114

Estos errores son bastante aceptables, pues estamos hablando de un error del 1.1 % y del 5.9 %, respectivamente. Estos errores son bastante aceptables y nuestra aproximación cae en un rango de aceptación bastante bueno.

3. Conclusiones

En esta actividad hemos hecho una aproximación a las mareas presentadas en dos regiones, con los resultados obtenidos de esta práctica podemos tener un modelo para predicciones de mareas, todo esto a partir del uso de la transformada de Fourier que nos da información sobre comportamientos de movimientos oscilatorios, como lo son las mareas.

En nuestros resultados se puede ver que hay discrepancias entre las mareas reales y las aproximadas pues esto depende de las condiciones naturales que se tienen en cada día y que generan más variaciones, además, no podemos considerar todos los nodos que devuelve la transformada de Fourier, lo cual nos daría una aproximación más exacta a las mareas que se presentan realmente.

4. Referencias

[1] J. E. T. Burruel, "Actividad 6: Análisis armónico de mareas," 2017. [Online]. Available: https://github.com/ErnestoTb/Computacional1/blob/master/Actividad%206/actividad-6-analisis.pdf

[2] Wikipedia, "Tide," Wikipedia Foundation, Inc. [Online]. Available: https://en.wikipedia.org/wiki/Tide

[3] C. for Operational Oceanographic Products and Services, "Tides and currents," 2013. [Online]. Available: https://tidesandcurrents.noaa.gov/

[4] C. de Investigación Científica y Educación Superior de Ensenada, "Calendarios de mareas en méxico," 2015.