Universidad de Sonora

División de Ciencias Exactas y Naturales

Física computacional I

Reconstruyendo la señal

Moreno Chávez Jesús Rodolfo
Profesor: Carlos Lizárraga Celaya

30 de Abril del 2017

Resumen

El presente texto contiene el procedimiento para reconstruír una señal de marea a aprtir de los modos obtenidos al realizar una transoformada de Fourier a un conjunto de datos .

Introducción

En la práctica 6 encontramos los principales constituyentes de las mareas de dos ciudades, recurriendo a la biblioteca de SciPy fftpack, utilizando la Transformada de Fourier Discreta (DFT). El objetivo de esta práctica es utilizar la información obtenida de la DFT para construír una función que nos permita generar una gráfica del comportamiento de las mareas, así como la que se mostró en la práctica 5.
En este texto se mostrará el cógido empleado para la reconstrucción y se mostrarán los resultados obtenidos, es decir, se mostrarán las gráficas de la reconstrucción comparadas con las gráficas originales del comportamiento de las mareas, así como el error de la aproximación.
Para esta práctica nuevamente son utilizados los datos del sitio de ÇICESE"para la ciudad de Huatulco, Oaxaca y el sitio de "Tides and Currents-NOAA"para Santa Barbara, Californio.

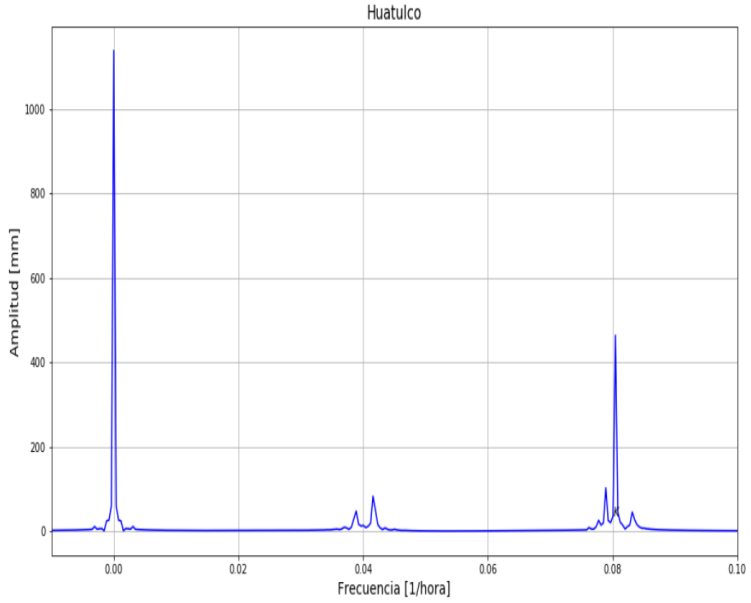
Reconstrucción de marea de Huatulco, Oaxaca

En la práctica anterior obtuvimos los siguintes modos al utilizar la transformada de Fourier discreta, usando el siguiente código

```
from scipy.fftpack import fft, fftfreq, fftshift
import numpy as np
# numero de datos
N = 2597
# Separacion de tiempo entre cada medicion
T = 1.0

#aplicacion de la transformada de fourier a los datos
y = df[u"Altura(mm)"]
yf = fft(y)
 #Cambio de variable de tiempo a uno de frecuencias.
xf = fftfreq(N, T)
xf = fftshift(xf)
# Conjunto de datos dados por la transformada de fourier para graficar
yplot = fftshift(yf)
```

Y obtuvimos lo siguinte:



Al generar una gráfica con DFT se buscaron los picos más notorios, por ejemplo, para amplitudes mayores a 25mm. Para ello utilizamos el siguiente comando:

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

print(np.where(a[:,]>25))
b= a[a[:,]>25]
b
```

y obtuvimos la siguiente información

```
(array([   0,    1,    2,  100,  101,  108,  109,  202,  205,  208,  209,
        210,  216,  217, 2380, 2381, 2387, 2388, 2389, 2392, 2395, 2488,
       2489, 2496, 2497, 2595, 2596], dtype=int64),)
Out[13]:
array([ 1137.79283789,    59.90169929,    26.06367895,    28.92596781,
          47.49734193,    82.76279772,    49.69035015,    25.71511853,
         102.37608803,    35.97289977,   463.70108706,    44.79733118,
          45.24414595,    27.11388926,    27.11388926,    45.24414595,
          44.79733118,   463.70108706,    35.97289977,   102.37608803,
          25.71511853,    49.69035015,    82.76279772,    47.49734193,
          28.92596781,    26.06367895,    59.90169929])
```

La cual nos da la correspondencia del número de dato y su amplitud correspondiente.

Esta información la utilizamos para encontrar con gran precisión más información asociada a la amplitud.
Por ejemplo, el dato 208 tiene asociada la siguiente información :

```
#
print( 'Armonico de prueba')
print('Amplitud=',2.0*np.absolute(yf[208,]/N))
print('frecuencia=', xf[int(N/2 +208),])
print('periodo=', 1/xf[int(N/2 +208),])
print()

Armonico de prueba
Amplitud= 35.9728997701
frecuencia= 0.0800924143242
periodo= 12.4855769231
```

Posteriormente se buscaron las amplitudes, frecuencias y faes para los primeros 13 picos (ya que a paritr de los primeos 13 comenzaban a repetirse).
Se obtuvieron las amplitudes con el siguiente código:

```
A0 = np.absolute(yf[int(0),]/N)
A1 = 2.0*np.absolute(yf[int(1),]/N)
A2 = 2.0*np.absolute(yf[int(2),]/N)
A3= 2.0*np.absolute(yf[int(100),]/N)
A4 = 2.0*np.absolute(yf[int(101),]/N)
A5 = 2.0*np.absolute(yf[int(108),]/N)
A6 = 2.0*np.absolute(yf[int(109),]/N)
A7 = 2.0*np.absolute(yf[int(202),]/N)
A8 = 2.0*np.absolute(yf[int(205),]/N)
A9 = 2.0*np.absolute(yf[int(208),]/N)
A10 =  2.0*np.absolute(yf[int(209),]/N)
A11 = 2.0*np.absolute(yf[int(210),]/N)
A12 = 2.0*np.absolute(yf[int(216),]/N)
A13 = 2.0*np.absolute(yf[int(217),]/N)
```

Posteriormente se obtuvieron las correspondientes frecuencias haciendo lo siguiente:

```
#frecuencias (N/2=1298.5)

fA1 = xf[int(1298.5 +1)]
fA2 = xf[int(1298.5 +2),]
fA3 =  xf[int(1298.5 +100),]
fA4 =  xf[int(1298.5 +101),]
```

```
fA5 =  xf[int(1298.5 +108),]
fA6 =  xf[int(1298.5 +109),]
fA7 =  xf[int(1298.5 +202),]
fA8 =  xf[int(1298.5 +205),]
fA9 =  xf[int(1298.5 +208),]
fA10 =  xf[int(1298.5 +209),]
fA11 =  xf[int(1298.5 +210),]
fA12 =  xf[int(1298.5 +216),]
fA13 =  xf[int(1298.5 +217),]
```

Y finalmente las correspondientes fases:

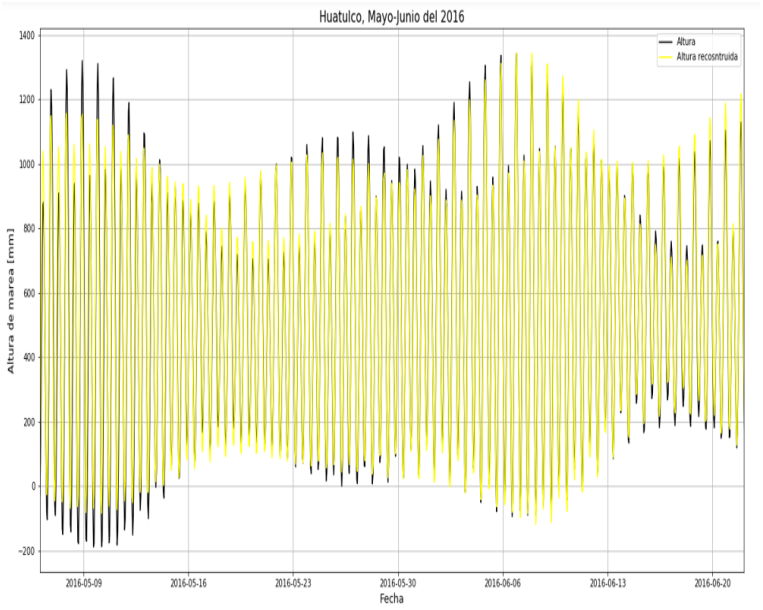
```
# Fases
FSA1 = np.angle(yf[int(1),])
FSA2 = np.angle(yf[int(2),])
FSA0 = np.angle(yf[int(0),])
FSA1 = np.angle(yf[int(1),])
FSA2 = np.angle(yf[int(2),])
FSA3 = np.angle(yf[int(100),])
FSA4 = np.angle(yf[int(101),])
FSA5 = np.angle(yf[int(108),])
FSA6 =  np.angle(yf[int(109),])
FSA7 =  np.angle(yf[int(202),])
FSA8 =  np.angle(yf[int(205),])
FSA9 =  np.angle(yf[int(208),])
FSA10 =  np.angle(yf[int(209),])
FSA11 =  np.angle(yf[int(210),])
FSA12 =  np.angle(yf[int(216),])
FSA13 =  np.angle(yf[int(217),])
```

Una vez obtenidas las amplitudes, frecuencias y fases, podemos hacer uso de esta información para reconstruir la señal por medio de una función de sumas de cosenos

```
#Obteniendo la marea
w= 2.0*np.pi
def f(t):
    return A0 + (A1*np.cos(w*fA1*t+FSA1) + A2*np.cos(w*fA2 *t+FSA2)
                    + A3*np.cos(w*fA3*t+FSA3) + A4*np.cos(w*fA4*t +FSA4)
                    + A5*np.cos(w*fA5*t+FSA5) + A6*np.cos(w*fA6*t +FSA6)
                    + A7*np.cos(w*fA7*t+FSA7) + A8*np.cos(w*fA8*t+ FSA8)
                    + A9*np.cos(w*fA9*t+ FSA9) + A10*np.cos(w*fA10*t+ FSA10)
```

```
+ A11*np.cos(w*fA11*t+ FSA11) + A12*np.cos(w*fA12*t+ FSA12)
+ A13*np.cos(w*fA13*t+ FSA13))
```

Al obtener la función que nos permite recosntruír la señal, la graficamos y comparamos con la gráfica original del comportamiento de las mareas. El resultado que se obtuvo es el siguiente:



La reconstrucción de marea obtenida tiene un error de 0.75 %. Dicho error fue calculado mediante le siguiente código:

```
# error:
y_0=df['Altura(mm)']
y_1=f(df['T'])

E= np.sum(abs(y_0-y_1)**2) / np.sum(np.abs(y_0)**2)
E

0.007511263079838983
```

Reconstrucción de marea de Santa Barbara, California

El procedimiento para la reconstrucción de la señal es el mismo que se usó para Huatulco, la única diferencia es que se utilizaron más términos en la función de reconstrucción de la señal.

La reconstrucción obtenida fue la siguiente:

El error fue de 1.05 %, el cual es un poco mayor al error obtenido para la reconstrucción de marea de huatulco. Esta diferencia se debe a que para Huatuclo se usaron más datos para la reconstrucción, y por ende la aproximación es mejor.

Bibliografía

[1] NOAA

Tides and currents, https://tidesandcurrents.noaa.gov/waterlevels.html?id=9410660

[2] CICESE

Calendario de mareas, http://predmar.cicese.mx/calendarios/

[3] FFTPACK

Transformada discreta de Fourier, https://docs.scipy.org/doc/scipy-0.18.1/reference/tutorial/fftpack.html