95.13

ANALISIS NUMERICO

TPM2: Predicciones de las mareas.

1er.Cuatrimestre de 2022

Nicolas Facciano 105859
Juan Biancuzzo 106005

Docentes a cargo del curso:

- Rodriguez Daniel.F
- Machiunas Valeria

ANÁLISIS NUMÉRICO I - 75.12 – 95.04 – 95.13
Curso: Rodríguez - Machiunas

Primer Cuatrimestre 2022

TRABAJO PRÁCTICO DE MÁQUINA Nº 2

Las altura de las mareas se pueden modelar utilizando una sumatoria de componentes armónicas de acuerdo a la siguiente ecuación:

$$altura = a_0 + \sum_{k=1}^{n} a_k cos\left(\omega_k t + \alpha_k\right) \quad (1)$$

donde:

a_0 es el nivel medio de referencia

n: número de componentes armónicas consideradas

a_k: amplitud de la componente armónica k-ésima

w_K: Frecuencia angular de la componente armónica k-ésima

α_k: Fase de la componente armónicas k-ésima

Los parámetros de la ecuación (1) se calculan a partir de la serie temporal de datos, obtenida por mareógrafos en años anteriores, conocida como marea astronómica o predicha.
Por lo general; los responsables de la toma de datos de los mareógrafos son organismos que dependen de los gobiernos. Históricamente realizaban una publicación con en el anuario de mareas y en ocasiones también las últimas constantes armónicas obtenidas en los puertos principales.
Cada puerto tiene asociado una tabla de mareas y en ocasiones hay que realizar correcciones debido a la influencia de los factores meteorológicos.
Actualmente parte de la información es de acceso público mediante páginas web, por ejemplo en Argentina el organismo responsable es el Servicio de Hidrografía Naval: http://www.hidro.gov.ar.
En Estados Unidos, la Administración Nacional Oceánica y Atmosférica (National Oceanic and Atmospheric Administration - NOAA) permite el acceso publico de datos (https://oceanservice.noaa.gov/)

El objetivo de este trabajo es estimar las constantes armónicas que permiten caracterizar las mareas de un puerto del cual se dispone de los siguientes datos: la altura de la marea durante el mes de enero cada seis minutos y la altura de la marea durante todo el año en forma horaria.

Para cumplir adecuadamente el objetivo propuesto, cada grupo elegirá un puerto diferente, de los suministrados en el aula virtual del curso (anotando en el foro asociado al TPM2, el puerto elegido), y se deberán realizar los siguientes ítems:

Nicolas Facciano 105859
Juan Biancuzzo 106005

Parte 1

(a) Primero grafique la altura de las mareas utilizando sólo los datos del mes de enero. Realice una estimación del tiempo entre dos máximos consecutivos y/o dos mínimos consecutivos.

(b) Determine la frecuencia angular más importante ($w_{1,}$). Para ello deberá utilizar la transformada discreta de Fourier. Podrá utilizar la implementación de la transformada rápida de Fourier (fft) en Octave o Python. Compare con el valor obtenido en el punto anterior.

(c) Utilizando el método de mínimos cuadrados (lineales) obtenga el nivel medio de referencia (a_0) y la amplitud de la componente armónica (a_1) como así también la fase. Indicar el ECM del ajuste.

Parte 2

(a) Grafique la altura de las mareas utilizando los datos de todo el año. Realice una estimación del tiempo entre dos máximos consecutivos y/o dos mínimos consecutivos.

(b) Determine la frecuencia angular más importante ($w_{1,}$). Para ello deberá utilizar la transformada discreta de Fourier. Compare con el valor obtenido en el punto anterior.

(c) Utilizando el método de mínimos cuadrados (lineales) obtenga el nivel medio de referencia (a_0) y la amplitud de la componente armónica (a_1) como así también la fase correspondiente. Indicar el ECM del ajuste.

(d) Determine las frecuencia angulares más importantes ($w_{1,}$ $w_{2,...,}$ w_n). Para ello deberá utilizar la transformada discreta de Fourier. De ser posible, interprete las mismas. (no utilice más de 20 frecuencias)

(e) Utilizando el método de mínimos cuadrados deberá obtener el nivel medio de referencia (a_0) y la amplitud de dos componentes armónicas (a_1, a_2) como así también las fases respectivas. Indicar el ECM del ajuste.

(f) Realice nuevamente el punto anterior agregando una nueva componente armónica. Calcule el ECM del ajuste. Repita el proceso, agregando una nueva componente armónica, hasta que el ECM no varíe más de un cinco por ciento en cada paso o hasta que llegue a 20 frecuencias.

Nicolas Facciano 105859
Juan Biancuzzo 106005

Objetivo:

Estimar las constantes armónicas que permiten caracterizar las mareas de un puerto del cual se dispone de los siguientes datos: la altura de la marea durante el mes de enero cada seis minutos y la altura de la marea durante todo el año en forma horaria.

Introducción:

El ciclo de marea se produce habitualmente dos veces al día (dos pleamares y dos bajamares).

Las mareas vivas se producen cuando la altura es máxima y las mareas muertas cuando la altura es mínima. De acuerdo con estas alturas, se han registrados datos de los mareógrafos, los cuales nos van a servir para poder calcular las constantes armónicas que caracterizan la función determinada por los mismos datos.

También hay otros factores que complican la descripción de las mareas como lo son:

1. El eje de la Tierra está inclinado 23.5° respecto de la perpendicular al plano de la eclíptica y esto hace que el ángulo formado por el plano ecuatorial y el plano de la eclíptica varíe en ±23.5° a lo largo del año.
2. La órbita de la Luna alrededor de la Tierra es elíptica, por lo que la distancia entre la Tierra y la Luna no es constante
3. La órbita de la Tierra alrededor del Sol es elíptica, por lo que la distancia entre la Tierra y el Sol no es constante.
4. El viento.

Uno de los puntos a analizar es la frecuencia angular más importante, que la averiguamos mediante la transformada rápida de Fourier. Y por otra parte podríamos calcular el nivel medio de referencia, la amplitud de la componente armónica y su fase. Esto último lo calcularemos mediante el método de cuadrados mínimo.

A la hora de analizar los datos, se decidió evaluar las alturas del mes de enero y las de un año completo.

Desarrollo:

Arrancando por el mes de enero, lo primero que hicimos fue cargar los datos de las alturas en Python para graficarlos y poder observar, para hacer esto debimos indicarle en el código cuales son los caracteres separadores del csv.

Para una mejor observación llevamos los datos a horas e hicimos el análisis desde ahí, pudiendo ver aproximadamente los mínimos y máximos de la función.

Nicolas Facciano 105859
Juan Biancuzzo 106005

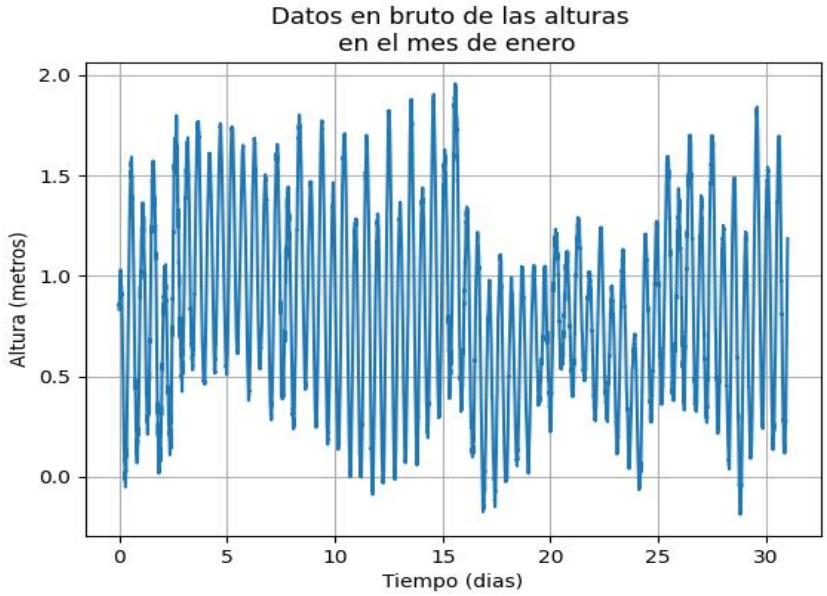

Figura 1: Grafico de las alturas de las mareas en el mes de enero.

Pudiendo ver en la **Figura 1** los datos en bruto de las alturas con los máximos y mínimos señalados.

Luego de visualizar los periodos, buscamos la frecuencia angular más importante utilizando la transformada rápida de Fourier con la librería numpy. Obtuvimos las frecuencias posibles y el peso de cada una.

$$X_k = \sum_{n=0}^{N-1} X_n * e^{-\frac{2x_i}{N} * K_n} \qquad K = 0, \dots, N-1$$

Donde X_k es un conjunto de números complejos.

- Calculamos el módulo de cada frecuencia y lo ubicamos en su posición correspondiente.
- Ordenamos las frecuencias según su orden de importancia. (mayor a menor)
- Devolvemos la cantidad que deseamos de frecuencias.

Pudimos observar los picos más grandes que en si son las frecuencias con mayor peso.

En esta parte tuvimos como inconveniente que nos daba una frecuencia grande y una chica, así sucesivamente. Porque la FFT dependiendo de la cantidad de datos que se usan se pueden duplicar de forma espejada, poniendo los datos del principio en el final. Por lo que solo agarramos la primera mitad.

Para el ítem c, que nos pedía utilizar el método de mínimos cuadrados para obtener el nivel medio de referencia (a0), la amplitud de la componente armónica (a1) y la fase.

$$f^*(x) = \sum_{i=1}^{m} c_i * \varphi_1(x)$$

Siendo $\varphi_1(x)$ la función conocida

Nicolas Facciano 105859
Juan Biancuzzo 106005

A partir de las amplitudes que encontramos con cuadrados mínimos llegamos a la amplitud armónica y fase mediante el siguiente razonamiento:

$$a_k \cdot \cos(w_k \cdot t) + b_k \cdot \operatorname{sen}(w_k \cdot t)$$

$$a_k = \mathrm{A} \cdot \cos(\alpha)$$
$$b_k = \mathrm{A} \cdot \operatorname{sen}(\alpha) \cdot (-1)$$

$$a_k = \frac{-b_k}{\operatorname{sen}(\alpha)} \cdot \cos(\alpha)$$

Unimos el cos y el sen…

$$\operatorname{tang}(\alpha) = \frac{-b_k}{a_k}$$

y despejamos α

$$\alpha = arctang(\frac{-b_k}{a_k})$$

Y sacamos la amplitud $$A = \frac{a_k}{\cos(\alpha)}$$

Y también se nos pide calcular el ECM que se calcula de la siguiente manera:

$$E_{CM}(f) = \sqrt{\frac{\sum_{k=1}^{n}(e_k)^2}{n}}$$

Creamos una función que haga el producto interno entre dos vectores.

$$(f, g) \coloneqq \sum_{k=1}^{m} f(x_k) \cdot g(x_k)$$

Aprovechamos la propiedad que esta matriz es simétrica para no tener que calcular los productos de la diagonal inferior de dicha matriz, decidimos crear una función que haga el producto interno entre dos vectores que vaya calculando cada uno de estos sin volver a hacer el simétrico del anteriormente calculado.

Nicolas Facciano 105859
Juan Biancuzzo 106005

Análogamente este procedimiento lo repetimos con los datos del año entero ya que intentamos programar un código bastante general que nos sirva para calcular cualesquiera sean los datos que les pasemos y la cantidad de frecuencias o componentes que queramos calcular.

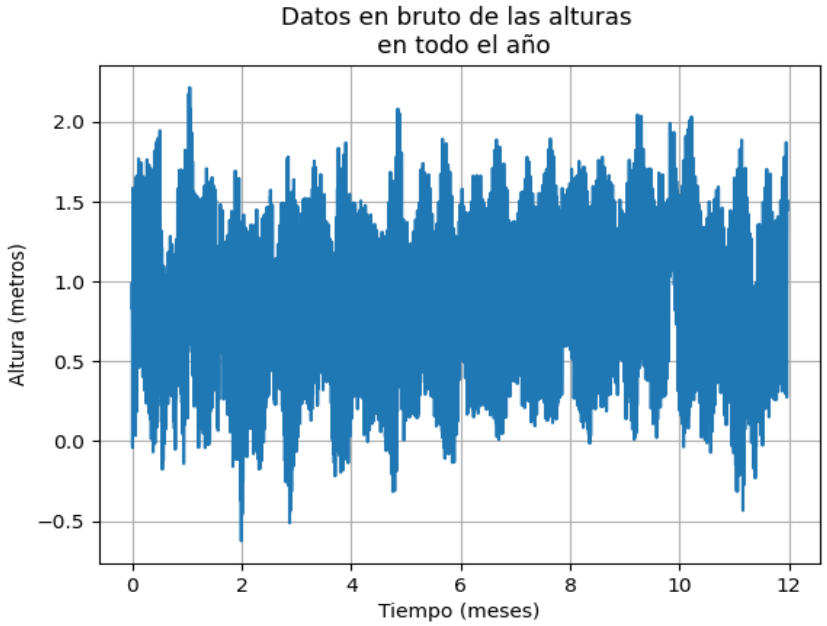

Figura 2: Grafico de las alturas de las mareas en un año.

En el caso de calcular con dos componentes armónicas, decidimos hacerlo a través del mismo proceso que necesitábamos para calcular que el error cuadrático medio no varie más de un 5%. Es decir que generamos un código el cual calcula por el método de cuadrados mínimos agregando de forma iterada una componente armónica más.

Una de las ideas principales que tuvimos al comienzo fue que las frecuencias más importantes deberían ser aquellas que son más lentas, pensando que la estructura de estas les darían la forma semejante a las alturas. Pero al hacer el análisis con las cuentas y aproximaciones correspondientes cambiamos esta hipótesis debido a que en realidad las de mayor relevancia son las frecuencias rápidas ya que dan una mejor aproximación a las alturas en si debido a su cercanía con dichas alturas.

Resultados:

Primera parte (mes de enero):

Tabla con las frecuencias más importantes ordenadas en forma decreciente según su orden.

TABLA 1 FRECUENCIAS PARA EL MES DE ENERO		
Frecuencia angular	**Periodo**	**Tipo de periodo**
60	12,4	hora
1	31,0	día
31	24,0	hora

Nicolas Facciano 105859
Juan Biancuzzo 106005

62	12,0	hora
59	12,6	hora
6	5,17	día
5	6,2	día
3	10,3	día
29	1,07	día
4	7,75	día
13	2,4	día
58	12,8	hora
61	12,2	hora
7	4,43	día
17	1,82	día
8	3,875	día
9	3,44	día
15	2,07	día
10	3,1	día
14	2,21	día

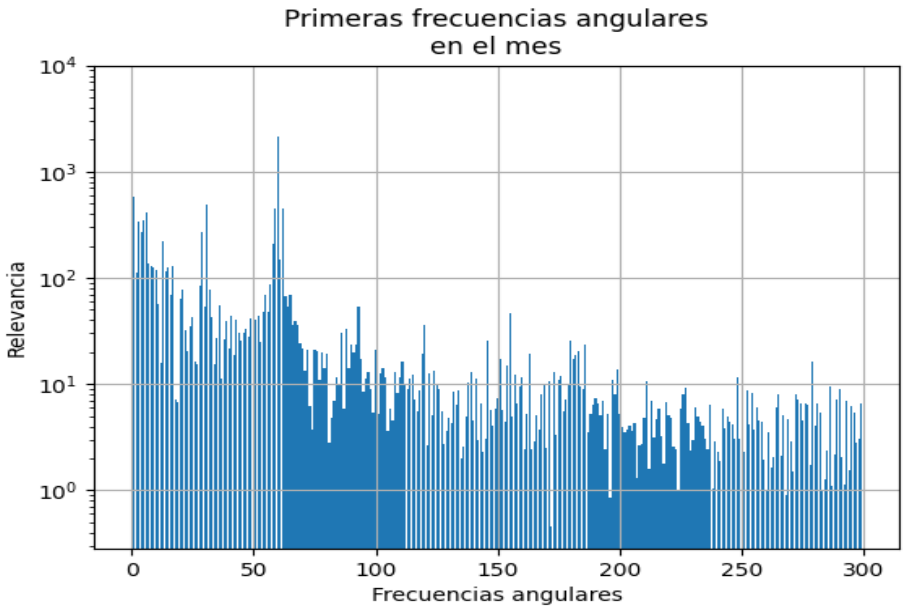

Figura 3: Grafico de las primeras frecuencias angulares del mes de enero.

TABLA 2: TABLA CON UNA FRECUENCIAS DE ENERO			
Posición media: 1.6731			
Amplitud 1	-1.1776	Fase 1	0.5679

Nicolas Facciano 105859
Juan Biancuzzo 106005

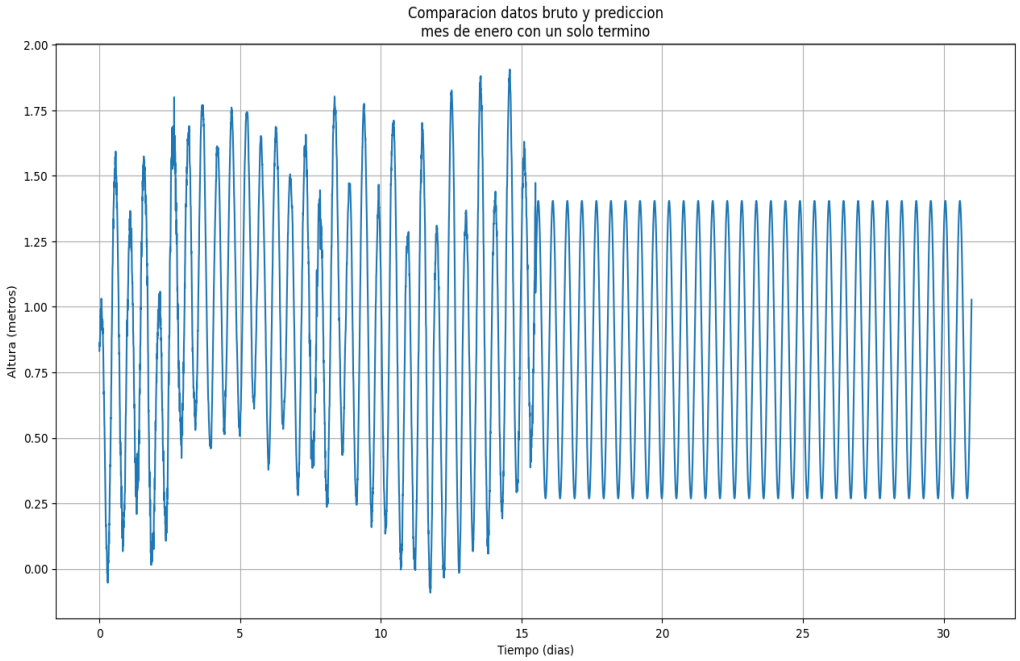

Figura 4: Grafico comparativo entre los datos en bruto de las alturas en el mes de enero y la predicción con un solo termino.

3x3	ECM	0.263

Segunda parte (todo el año):

TABLA 3: FRECUENCIAS PARA EL AÑO		
Frecuencia angular	**Periodo**	**Tipo de periodo**
705	12,43	hora
706	12,41	hora
692	12,66	hora
366	23,93	hora
730	12	hora
704	12,44	hora
707	12,39	hora
1	12,17	mes
339	1,077	dia
4	3,042	mes
703	12,46	hora
16	22,81	dia
708	12,37	hora
5	2,433	mes
702	12,48	hora
709	12,36	hora
19	19,21	dia

Nicolas Facciano 105859
Juan Biancuzzo 106005

7	1,738	mes
732	11,97	hora
26	14,04	dia

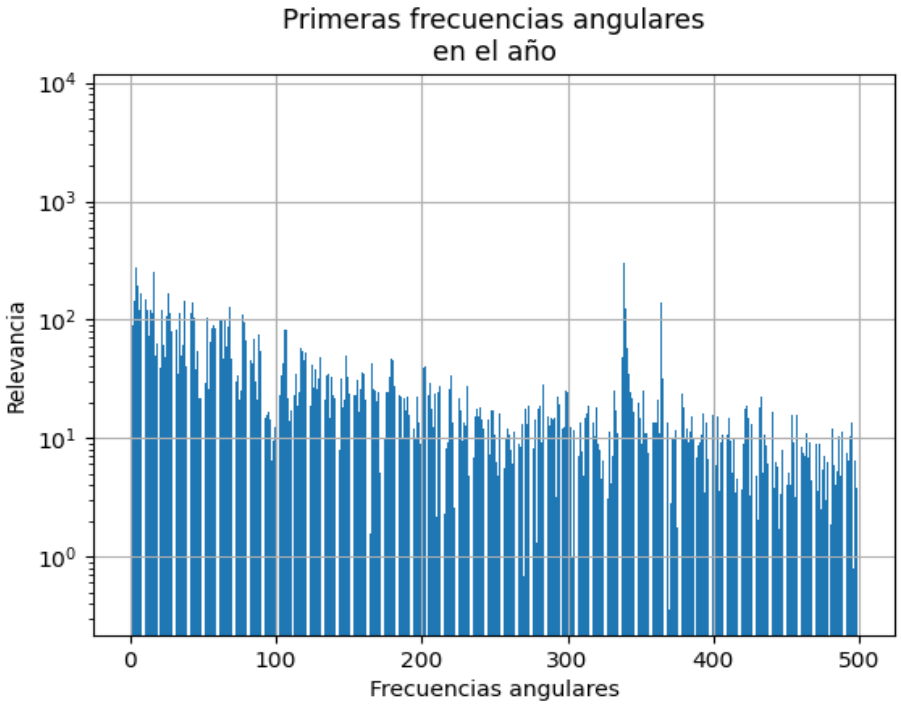

Figura 5: Grafico de las primeras frecuencias angulares de todo el año.

TABLA 4: TABLA CON UNA FRECUENCIAS EN EL AÑO			
Posición media: 1.662			
Amplitud 1	-0.032	Fase 1	0.501

TABLA 5: TABLA CON 2 FRECUENCIAS EN EL AÑO			
Posición media: 1.662			
Amplitud 1	-0.032	Fase 1	0.501
Amplitud 2	-0.038	Fase 2	-0.194

TABLA 6: TABLA CON 7 FRECUENCIAS EN EL AÑO			
Posición media: 1.662			
Amplitud 1	-0.032	Fase 1	0.501
Amplitud 2	-0.038	Fase 2	-0.194
Amplitud 3	0.970	Fase 3	0.144
Amplitud 4	0.022	Fase 4	-0.116
Amplitud 5	-0.382	Fase 5	0.109
Amplitud 6	0.021	Fase 6	0.106
Amplitud 7	0.064	Fase 7	-0.086

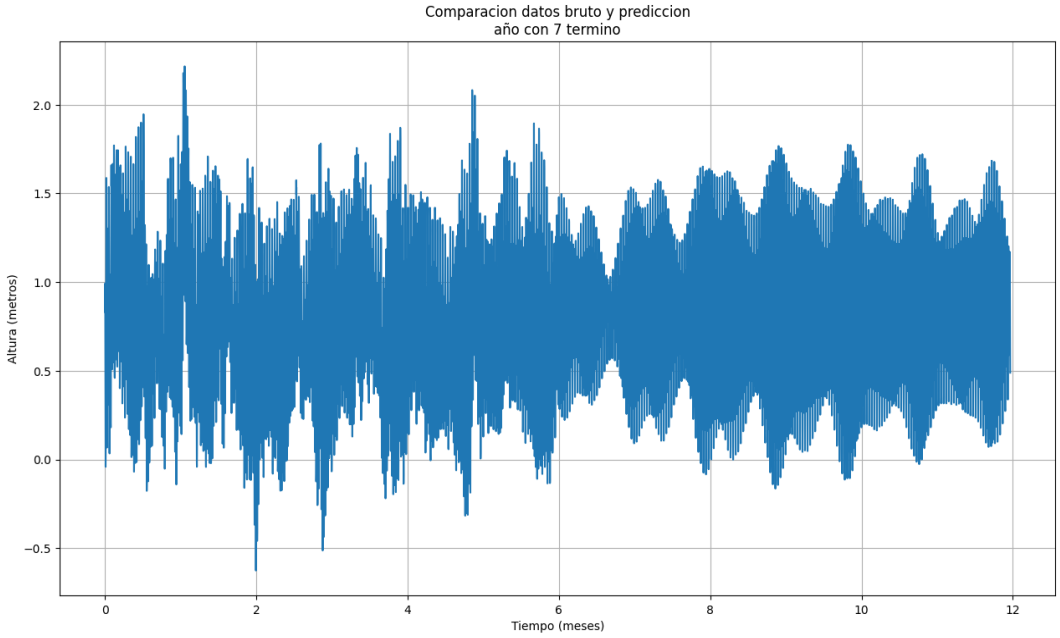

Figura 6: Grafico comparativo entre los datos en bruto y la predicción de las alturas en el año con 7 términos.

TABLA 7: TABLA DEL ERROR CUADRATICO MEDIO		
3x3	ECM	0.311
5x5	ECM	0.279
7x7	ECM	0.260
9x9	ECM	0.247
11x11	ECM	0.234

Nicolas Facciano 105859
Juan Biancuzzo 106005

13x13	ECM	0.222
15x15	ECM	0.213

Conclusiones:

En la **Figura 1** analizamos los máximos y mínimos de la representación de dichas alturas, pudiendo ver como el primer periodo esta aproximadamente entre 0 y 12 días y luego se repiten sucesivamente periodos de 5 días aproximadamente.

En cuanto al cálculo de las frecuencias del mes de enero (mostrados en la primera tabla), pudimos ver que, aunque las frecuencias que observamos en la imagen están aproximadamente dentro de las 20 más importantes, hay muchas que no pudimos interpretar y son aun de mayor frecuencia que las que vimos a simple vista.

Como vemos en la **Tabla 1**, los periodos más importantes no son los que mencionamos anteriormente, aunque aparecen (como el periodo de 5 días) pero es la más relevante. Esto podemos pensarlo como la capacidad de la transformada rápida de Fourier en analizar con mayor detalle y mayor precisión esta periodicidad que tienen las mareas que nosotros a simple viste no podríamos observar.

También podemos notar como los periodos más importantes no bajan de las 12 horas, por lo que podemos entender que en el periodo más chico es de esas 12 horas. Con una cierta variación dada por factor externos como lo puede ser el viento.

Algo a mencionar, es que podemos decir que, según la serie de Fourier, esta es la mejor predicción con 3 términos. Esto nos lo asegura la Transformada de Fourier que busca la relevancia de los coeficientes de cada termino, en nuestro caso sobre las alturas de las mareas.

En cuanto a la segunda parte del desarrollo, relacionada a las mediciones hechas durante un año completo, pudimos asimilar de igual manera que en el análisis del mes de enero, que hay ciertos periodos que siguen sosteniendo su relevancia dentro del análisis anual.

Siendo los periodos con más relevancia que pudimos observar en la **Figura 2** el de 1 mes y 8 meses aproximadamente.

Al calcular y ordenar el peso de las frecuencias notamos que la correspondiente al periodo de un mes se encuentra en la posición 18 (mostrado en la tabla 3); y la correspondiente al periodo de 8 meses no se encuentra entre las primeras 20 frecuencias. Al igual que mencionamos anteriormente, nuestra habilidad para encontrar el periodo con mayor relevancia es inferior a la que nos brinda la Transformada de Fourier.

Una de las cosas que pudimos concluir mediante la resolución del ítem c, e y f, es que, al agregar frecuencias para generar otra componente armónica, vemos que la Amplitud y la Fase correspondientes no varían por dicha modificación.

Para concluir, remarcaremos como la Transformada de Fourier nos permite obtener los términos más relevantes para la aproximación, ahorrándonos no solo el cálculo de los coeficientes de Fourier, sino que también tener una mejor aproximación con la menor cantidad de términos. Esto lo fuimos mencionando, en la parte 1 con la mejor aproximación de 3 términos, y para el final con la mejor aproximación donde tiene un error menor al 5% en relación con la iteración anterior.

Bibliografía:

Nicolas Facciano 105859
Juan Biancuzzo 106005

García A.F. (2016). Análisis armónico de las mareas. [Articulo]. Análisis de datos. Recuperado de: http://www.sc.ehu.es/sbweb/fisica3/datos/mareas/mareas_1.html [11/06/2022].

Anexo I:

Main

```
from numpy import arange

        from Configuracion import datosSeisMinutosArchivo, datosUnaHoraArchivo,
        minutosPorDatoEnArchivoSeisMinutos, minutosPorDatoenArchivoUnaHora,
        porcentajeMinimo
        from Transformacion import FrecuenciasAngularesOrdenadasPorImportancia,
        FuncinesPhi
        from Utilidades import LeerArchivo
        from CuadradosMinimos import MinimosCuadrados, CalculoDeAmplitudYFase,
        ErrorCuadraticoMedio

        def CalcularPeriodo(periodoEnMinutos):
            tipoDeDato = "min"
            if periodoEnMinutos / 60 < 1:
                return periodoEnMinutos, tipoDeDato

            periodoEnHora = periodoEnMinutos / 60
            tipoDeDato = "hora"
            if periodoEnHora / 24 < 1:
                return periodoEnHora, tipoDeDato

            periodoEnDia = periodoEnHora / 24
            tipoDeDato = "dia"
            if periodoEnDia / 30 < 1:
                return periodoEnDia, tipoDeDato

            periodoEnMes = periodoEnDia / 30
            tipoDeDato = "mes"
            if periodoEnMes / 12 < 1:
                return periodoEnMes, tipoDeDato

            periodoEnAnio = periodoEnMes / 12
            tipoDeDato = "año"
            return periodoEnAnio, tipoDeDato

        def MostrarFrecuenciasImportantes(frecuenciasImportantes,
        cantidadDeFrecuencias, cantidadDeDatos, minutosPorDato):
            frecuencias = frecuenciasImportantes[:cantidadDeFrecuencias]
            for frecuencia in frecuencias:
                frecuenciaAngular = int(frecuencia[1])
```

```
        periodo = (cantidadDeDatos * minutosPorDato) /
frecuenciaAngular
        periodo, tipoDePeriodo = CalcularPeriodo(periodo)

        periodoCon3Digitos = "{0:.3f}".format(periodo)
        print(f"Frecuencia angular: {frecuenciaAngular}, con un periodo
de {periodoCon3Digitos} {tipoDePeriodo}")

def DatosDeLaSerie(amplitudesDeLaSerie):

    datosDeLaSerie = [amplitudesDeLaSerie[0]]
    for i in range(1, len(amplitudesDeLaSerie[1:]), 2):
        amplitudCoseno = amplitudesDeLaSerie[i]
        amplitudSeno = amplitudesDeLaSerie[i + 1]

        amplitud, fase = CalculoDeAmplitudYFase(amplitudCoseno,
amplitudSeno)
        datosDeLaSerie.append(amplitud)
        datosDeLaSerie.append(fase)

    return datosDeLaSerie

def MostrarDatosDeLaSerie(datosDeLaSerie):
    posicionMedia3Digitos = "{0:.3f}".format(datosDeLaSerie[0])
    print(f"Posicion media: {posicionMedia3Digitos}")
    for i in range(1, len(datosDeLaSerie[1:]), 2):
        amplitud = datosDeLaSerie[i]
        fase = datosDeLaSerie[i + 1]
        posicion = (i + 1) // 2
        amplitudCon3Digitos = "{0:.3f}".format(amplitud)
        faseCon3Digitos = "{0:.3f}".format(fase)
        print(f"Amplitud {posicion}: {amplitudCon3Digitos}, Fase
{posicion}: {faseCon3Digitos}")

def CalculoDelError(cantidadDeTerminos, datosTiempo, datosAltura):
    cantidadDeDatos = len(datosAltura)
    frecuenciasImportantes =
FrecuenciasAngularesOrdenadasPorImportancia(datosAltura)
    funcinesPhi =
FuncinesPhi(frecuenciasImportantes[:cantidadDeTerminos],
cantidadDeDatos)
    amplitudesSerie = MinimosCuadrados(funcinesPhi, datosTiempo,
datosAltura)
    return ErrorCuadraticoMedio(funcinesPhi, amplitudesSerie,
datosTiempo, datosAltura)
```

```
def MostrarErrorCuadraticoMedio(cantidadDeTerminos,
errorCuadraticoMedio):
    tamanioDeLaMatriz = 1 + 2 * cantidadDeTerminos
    errorCon3Decimales = "{0:.3f}".format(errorCuadraticoMedio)
    print(f"{tamanioDeLaMatriz}x{tamanioDeLaMatriz}: ECM:
{errorCon3Decimales}")

def SeguirIterando(errorActual, errorAnterior, porcentaje):
    return abs(errorActual - errorAnterior) / abs(errorActual) >
porcentaje / 100

def Main():
    print("Parte 1) \n\tDatos del mes\n")

    datosAltura = LeerArchivo(datosSeisMinutosArchivo)
    cantidadDeDatos = len(datosAltura)
    datosTiempo = arange(cantidadDeDatos)
    cantidadDeTerminos = 1

    frecuenciasImportantes =
FrecuenciasAngularesOrdenadasPorImportancia(datosAltura)

    MostrarFrecuenciasImportantes(frecuenciasImportantes,
cantidadDeTerminos, cantidadDeDatos,
minutosPorDatoEnArchivoSeisMinutos)

    funcionesPhi =
FuncinesPhi(frecuenciasImportantes[:cantidadDeTerminos],
cantidadDeDatos)
    amplitudesSerie = MinimosCuadrados(funcionesPhi, datosTiempo,
datosAltura)

    datosDeLaSerie = DatosDeLaSerie(amplitudesSerie)
    MostrarDatosDeLaSerie(datosDeLaSerie)

    errorCuadraticoMedio = CalculoDelError(cantidadDeTerminos,
datosTiempo, datosAltura)
    MostrarErrorCuadraticoMedio(cantidadDeTerminos,
errorCuadraticoMedio)

    print("\nParte 2) \n\tDatos del año\n")

    datosAltura = LeerArchivo(datosUnaHoraArchivo)
    cantidadDeDatos = len(datosAltura)
    datosTiempo = arange(cantidadDeDatos)
```

```
    frecuenciasImportantes =
FrecuenciasAngularesOrdenadasPorImportancia(datosAltura)

    MostrarFrecuenciasImportantes(frecuenciasImportantes,
cantidadDeTerminos, cantidadDeDatos, minutosPorDatoenArchivoUnaHora)

    funcionesPhi =
FuncinesPhi(frecuenciasImportantes[:cantidadDeTerminos],
cantidadDeDatos)
    amplitudesSerie = MinimosCuadrados(funcionesPhi, datosTiempo,
datosAltura)

    datosDeLaSerie = DatosDeLaSerie(amplitudesSerie)
    MostrarDatosDeLaSerie(datosDeLaSerie)

    errorCuadraticoMedioAnterior = CalculoDelError(cantidadDeTerminos,
datosTiempo, datosAltura)
    MostrarErrorCuadraticoMedio(cantidadDeTerminos,
errorCuadraticoMedioAnterior)

    cantidadDeTerminos += 1
    errorCuadraticoMedioActual = CalculoDelError(cantidadDeTerminos,
datosTiempo, datosAltura)
    MostrarErrorCuadraticoMedio(cantidadDeTerminos,
errorCuadraticoMedioActual)

    while SeguirIterando(errorCuadraticoMedioActual,
errorCuadraticoMedioAnterior, porcentajeMinimo) and cantidadDeTerminos
<= 20:

        cantidadDeTerminos += 1
        errorCuadraticoMedioAnterior = errorCuadraticoMedioActual
        errorCuadraticoMedioActual =
CalculoDelError(cantidadDeTerminos, datosTiempo, datosAltura)
        MostrarErrorCuadraticoMedio(cantidadDeTerminos,
errorCuadraticoMedioActual)

    print(f"El error cuadratico medio de ", end = "")
    MostrarErrorCuadraticoMedio(cantidadDeTerminos,
errorCuadraticoMedioActual)

    print("\nLa informacion final dio: ")
    frecuenciasImportantes =
FrecuenciasAngularesOrdenadasPorImportancia(datosAltura)
    funcinesPhi =
FuncinesPhi(frecuenciasImportantes[:cantidadDeTerminos],
cantidadDeDatos)
```

```
        amplitudesSerie = MinimosCuadrados(funcinesPhi, datosTiempo,
    datosAltura)
        datosDeLaSerie = DatosDeLaSerie(amplitudesSerie)
        MostrarDatosDeLaSerie(datosDeLaSerie)
        MostrarFrecuenciasImportantes(frecuenciasImportantes,
    cantidadDeTerminos, cantidadDeDatos, minutosPorDatoenArchivoUnaHora)

    if __name__ == "__main__":
        Main()
```

Cuadradosminimos

```
From numpy import float64, dot, ndarray, linalg, array, multiply, subtract, add,
arctan, cos

            from math import sqrt

            def MinimosCuadrados(funcionesPhi, datosX, datosY):
                matriz = CalculoDeLaMatriz(funcionesPhi, datosX)
                vector = CalculoDeFunciones(funcionesPhi, datosX, datosY)
                return ResolucionDeEcuacionesLineares(matriz, vector)

            def ResolucionDeEcuacionesLineares(matriz, vector):
                return linalg.solve(matriz, vector)

            def ErrorCuadraticoMedio(funcionesPhi, pesos, datosX : ndarray,
            datosY : ndarray) -> float:
                cantidadDeDatos = len(datosX)
                datosProcesados = FuncionEstrella(funcionesPhi, pesos, datosX)
                diferencia = subtract(datosY, datosProcesados)
                return sqrt(dot(diferencia, diferencia) / cantidadDeDatos)

            def FuncionEstrella(funcionesPhi, pesos, datosX) -> ndarray:
                datosProcesados = array([])
                for i, funcionPhi in enumerate(funcionesPhi):
                    resultado = multiply(funcionPhi(datosX), pesos[i])
                    datosProcesados = resultado if datosProcesados.size == 0
            else add(datosProcesados, resultado)
                return datosProcesados

            def CalculoDeAmplitudYFase(amplitudCoseno, amplitudSeno):
                fase = arctan(-amplitudSeno/amplitudCoseno)
                amplitud = amplitudCoseno/cos(fase)

                return fase, amplitud

            def CalculoDeLaMatriz(funcionesPhi, datosX):
                matriz = []
```

```
                for i, funcionF in enumerate(funcionesPhi):
                    fila = []
                    resultadoF = funcionF(datosX)
                    for j, funcionG in enumerate(funcionesPhi):
                        resultadoG = funcionG(datosX)
                        resultado = matriz[j][i] if j < i else
            ProductoInterno(resultadoF, resultadoG)
                        fila.append(resultado)
                    matriz.append(fila)
                return matriz

            def CalculoDeFunciones(funcionesPhi, datosX, datosY):
                vector = []
                for funcionPhi in funcionesPhi:
                    resultado = ProductoInterno(funcionPhi(datosX), datosY)
                    vector.append(resultado)
                return vector

            def ProductoInterno(funcionF : ndarray, funcionG : ndarray) ->
            float64:
                return dot(funcionF, funcionG)
```

Transformación

```
from numpy import fft, ndarray, linalg, array, multiply, ones, cos, sin, pi


    def FrecuenciasAngulares(datos : ndarray) -> ndarray:
        vectoresComplejos = fft.fft(datos)
        frecuencias = []

        for posicion, vector in enumerate(vectoresComplejos):
            modulo = linalg.norm(vector)
            frecuencias.append((modulo, posicion))

        return frecuencias[:len(frecuencias)//2]

    def FrecuenciasAngularesOrdenadasPorImportancia(datos : ndarray) -> ndarray:
        frecuenciasImportantes = FrecuenciasAngulares(datos)[1:]
        frecuenciasImportantes.sort(reverse = True, key = lambda valor: valor[0])
        return array(frecuenciasImportantes)

    def FuncinesPhi(frecuenciasImportantes, cantidadDeDatos):
        funcionesPhi = [lambda x : multiply(ones(cantidadDeDatos), 1/2)]
        for frecuencia in frecuenciasImportantes:
            orden = int(frecuencia[1])
```

```
            funcionCos = lambda x, orden = orden : cos(multiply((2 * pi * orden) /
        cantidadDeDatos, x))
            funcionSin = lambda x, orden = orden : sin(multiply((2 * pi * orden) /
        cantidadDeDatos, x))

            funcionesPhi.append(funcionCos)
            funcionesPhi.append(funcionSin)
        return funcionesPhi
```

Pruebadecuadradosminimos

```
from csv import writer

        from numpy import genfromtxt, ndarray, float64
        from Configuracion import caracterSeparador, lineasASaltear,
        columaAUsar

        def LeerArchivo(nombreArchivo) -> ndarray:
            return genfromtxt(nombreArchivo, dtype = float64, delimiter =
        caracterSeparador, skip_header = lineasASaltear, usecols = columaAUsar)

        def GuardarCSV(nombreArchivo, valores, nombresDeValores):
            with open(nombreArchivo, 'w', newline='') as f:
                escritor = writer(f)
                escritor.writerow(nombresDeValores)
                for valor in valores:
                    escritor.writerow(valor)
```

Configuración

```
datosUnaHoraArchivo = "Datos\CO-OPS_8534720_met-1hora.csv"

        datosSeisMinutosArchivo = "Datos\CO-OPS_8534720_met-6minutos.csv"
        periodoConArchivoDeSeisMinutos =
        "Datos\PeriodoConArchivoDeSeisMinutos.csv"
        periodoConArchivoDeUnaHora = "Datos\PeriodoConArchivoDeUnaHora.csv"
        caracterSeparador = ","
        lineasASaltear = 1
        columaAUsar = 4
        minutosPorDatoEnArchivoSeisMinutos = 6
        minutosPorDatoenArchivoUnaHora = 60
        porcentajeMinimo = 5
```

Utilidades

```
from csv import writer

        from numpy import genfromtxt, ndarray, float64
```

Nicolas Facciano 105859
Juan Biancuzzo 106005

```
from Configuracion import caracterSeparador, lineasASaltear,
columaAUsar

def LeerArchivo(nombreArchivo) -> ndarray:
    return genfromtxt(nombreArchivo, dtype = float64, delimiter =
caracterSeparador, skip_header = lineasASaltear, usecols = columaAUsar)

def GuardarCSV(nombreArchivo, valores, nombresDeValores):
    with open(nombreArchivo, 'w', newline='') as f:
        escritor = writer(f)
        escritor.writerow(nombresDeValores)
        for valor in valores:
            escritor.writerow(valor)
```

Plot:

```
from matplotlib import pyplot

from numpy import float64, ndarray, array, arange
from Configuracion import datosSeisMinutosArchivo, datosUnaHoraArchivo
from Transformacion import FrecuenciasAngulares,
FrecuenciasAngularesOrdenadasPorImportancia, FuncinesPhi
from CuadradosMinimos import MinimosCuadrados, FuncionEstrella
from Utilidades import LeerArchivo

def Rango(inicio : float64, separacion : float64, cantidad : int) -> ndarray:
    resultado = []
    for i in range(cantidad):
        resultado.append(inicio + separacion * i)
    return array(resultado)


def GraficarDatos(datos : ndarray, separacion : float64, titulo = "", ejeX = "", ejeY
= "", inicio = 0, logX = False, logY = False):
    rango = Rango(inicio, separacion, len(datos))
    GraficoContinuo(datos, rango, titulo, ejeX, ejeY, logX, logY)

def GraficarDatosBarra(datos : ndarray, separacion : float64, titulo = "", ejeX = "",
ejeY = "", inicio = 0, logX = False, logY = False):
    rango = Rango(inicio, separacion, len(datos))
    GraficoDeBarra(datos, rango, titulo, ejeX, ejeY, logX, logY)


def GraficoContinuo(datos : ndarray, rango : ndarray, titulo, ejeX, ejeY, logX, logY):
    fig = pyplot.figure()
    ax = fig.add_subplot(1, 1, 1)
    if logX: ax.set_xscale('log')
    if logY: ax.set_yscale('log')
```

Nicolas Facciano 105859
Juan Biancuzzo 106005

```
    pyplot.title(titulo)
    pyplot.xlabel(ejeX)
    pyplot.ylabel(ejeY)
    pyplot.plot(rango, datos)
    pyplot.grid(True)

def GraficoDeBarra(datos : ndarray, rango : ndarray, titulo, ejeX, ejeY, logX, logY):
    fig = pyplot.figure()
    ax = fig.add_subplot(1, 1, 1)
    if logX: ax.set_xscale('log')
    if logY: ax.set_yscale('log')

    pyplot.title(titulo)
    pyplot.xlabel(ejeX)
    pyplot.ylabel(ejeY)
    ax.bar(rango, datos)
    pyplot.grid(True)

def MostrarGraficos():
    pyplot.show()

def PrimerasFrecuenciasAngulares(datosDeAltura, cantidadMaxima):
    frecuenciasAngulares = FrecuenciasAngulares(datosDeAltura)
    frecuenciasAngularesConElModulo = []
    for frencuanciaAngular in frecuenciasAngulares:
        modulo = frencuanciaAngular[0]
        if len(frecuenciasAngularesConElModulo) < cantidadMaxima:
            frecuenciasAngularesConElModulo.append(modulo)
    return frecuenciasAngularesConElModulo

def Main():

    # datos en bruto

    datosAlturasMinutos = LeerArchivo(datosSeisMinutosArchivo)
    separacionMinutos = 1/(10 * 24)

    GraficarDatos(datosAlturasMinutos, separacionMinutos, "Datos en bruto de las
alturas \n en el mes de enero", "Tiempo (dias)", "Altura (metros)")

    datosAlturasHora = LeerArchivo(datosUnaHoraArchivo)
    separacionHora = 1/(24 * 30.5)

    GraficarDatos(datosAlturasHora, separacionHora, "Datos en bruto de las alturas \n
en todo el año", "Tiempo (meses)", "Altura (metros)")

    MostrarGraficos()
```

```
    # frecuencias que se obtienen del fft

    frecuenciasDelMes = PrimerasFrecuenciasAngulares(datosAlturasMinutos, 300)
    frecuenciasDelAño = PrimerasFrecuenciasAngulares(datosAlturasHora, 500)

    GraficarDatosBarra(frecuenciasDelMes, 1, "Primeras frecuencias angulares\nen el
mes", "Frecuencias angulares", "Relevancia", logY = True)
    GraficarDatosBarra(frecuenciasDelAño, 1, "Primeras frecuencias angulares\nen el
año", "Frecuencias angulares", "Relevancia", logY = True)

    MostrarGraficos()

    # Comparacion entre datos y prediccion

    cantidadDeDatos = len(datosAlturasMinutos)
    datosTiempo = arange(cantidadDeDatos)

    frecuenciasImportantes =
FrecuenciasAngularesOrdenadasPorImportancia(datosAlturasMinutos)

    funcionesPhi = FuncinesPhi(frecuenciasImportantes[:1], cantidadDeDatos)
    amplitudesSerie = MinimosCuadrados(funcionesPhi, datosTiempo, datosAlturasMinutos)

    prediccionAlturasMinutos = FuncionEstrella(funcionesPhi, amplitudesSerie,
datosTiempo)
    comparacionAlturasDelMes = [*(datosAlturasMinutos[:cantidadDeDatos//2]),
*(prediccionAlturasMinutos[:cantidadDeDatos//2])]

    GraficarDatos(comparacionAlturasDelMes, separacionMinutos, "Comparacion datos
bruto y prediccion\nmes de enero con un solo termino", "Tiempo (dias)", "Altura
(metros)")

    MostrarGraficos()

    cantidadDeDatos = len(datosAlturasHora)
    datosTiempo = arange(cantidadDeDatos)

    frecuenciasImportantes =
FrecuenciasAngularesOrdenadasPorImportancia(datosAlturasHora)

    funcionesPhi = FuncinesPhi(frecuenciasImportantes[:7], cantidadDeDatos)
    amplitudesSerie = MinimosCuadrados(funcionesPhi, datosTiempo, datosAlturasHora)

    prediccionAlturasHoras = FuncionEstrella(funcionesPhi, amplitudesSerie,
datosTiempo)
```

```
            comparacionAlturasDelAnio = [*(datosAlturasHora[:cantidadDeDatos//2]),
        *(prediccionAlturasHoras[:cantidadDeDatos//2])]

            GraficarDatos(comparacionAlturasDelAnio, separacionHora, "Comparacion datos bruto
        y prediccion\naño con 7 termino", "Tiempo (meses)", "Altura (metros)")

            MostrarGraficos()


        if __name__ == "__main__":
            Main()
```

Anexo II:

```
Parte 1)
        Datos del mes
Frecuencia angular: 60, con un periodo de 12.400 hora
Posicion media: 1.673
Amplitud 1: -1.178, Fase 1: 0.568
3x3: ECM: 0.263

Parte 2)
        Datos del año

Frecuencia angular: 705, con un periodo de 12.426 hora
Posicion media: 1.662
Amplitud 1: -0.032, Fase 1: 0.501
3x3: ECM: 0.311
5x5: ECM: 0.279
7x7: ECM: 0.260
9x9: ECM: 0.247
11x11: ECM: 0.234
13x13: ECM: 0.222
15x15: ECM: 0.213
El error cuadratico medio de 15x15: ECM: 0.213

La informacion final dio:
Posicion media: 1.662
```

Nicolas Facciano 105859
Juan Biancuzzo 106005

```
Amplitud 1: -0.032, Fase 1: 0.501
Amplitud 2: -0.038, Fase 2: -0.194
Amplitud 3: 0.970, Fase 3: 0.144
Amplitud 4: 0.022, Fase 4: -0.116
Amplitud 5: -0.382, Fase 5: 0.109
Amplitud 6: 0.021, Fase 6: 0.106
Amplitud 7: 0.064, Fase 7: -0.086
Frecuencia angular: 705, con un periodo de 12.426 hora
Frecuencia angular: 706, con un periodo de 12.408 hora
Frecuencia angular: 692, con un periodo de 12.659 hora
Frecuencia angular: 366, con un periodo de 23.934 hora
Frecuencia angular: 730, con un periodo de 12.000 hora
Frecuencia angular: 704, con un periodo de 12.443 hora
Frecuencia angular: 707, con un periodo de 12.390 hora
```

Nicolas Facciano 105859
Juan Biancuzzo 106005