Métodos Computacionales 2: Taller 2

Transformadas de Fourier

Por Santiago Henao Castellanos

Puede usar cualquier librería que desee, pero les aconsejo no poner código que no comprendan.

La entrega de este taller será en una carpeta dedicada en su repositorio de github del curso llamada `Taller 2`, con un **único** archivo `.py` que contenga los puntos en orden y señalados. Todo lo que se pida exportar, como gráficas, imágenes o videos, debe estar presente en dicha carpeta.

El código debe correr si se clona el repositorio, sin ninguna acción adicional. Es decir, todos los archivos necesarios para el funcionamiento, como datos para importar, deben estar también en la carpeta.

Si el código alza una excepción, se calificará hasta ese punto, así que prueben el código antes de subirlo.

No incluya ningún `plt.show()` o similares en su código.

1. Transformada general

Utilice la siguiente función para generar datos de prueba:

```
from numpy.typing import NDArray

def datos_prueba(t_max:float, dt:float, amplitudes:NDArray[float],
                 frecuencias:NDArray[float], ruido:float=0.0) -> NDArray[float]:
    ts = np.arange(0.,t_max,dt)
    ys = np.zeros_like(ts,dtype=float)
    for A,f in zip(amplitudes,frecuencias):
        ys += A*np.sin(2*np.pi*f*ts)
    ys += np.random.normal(loc=0,size=len(ys),scale=ruido) if ruido else 0
    return ts,ys
```

1.a. Implementación

Implemente la forma general de la transformada de Fourier:

$$\mathcal{F}\{t_i, y_i\}(f) = \sum_{k=0}^{N-1} y_k e^{-2\pi i t_k f}$$

en la forma de una función con la signatura:

```
def Fourier(t:NDArray[float], y:NDArray[float], f:float) -> complex:
```

o, si lo prefiere, que la calcule para varias frecuencias al mismo tiempo:

```
def Fourier(t:NDArray[float], y:NDArray[float],
            f:NDArray[float]) -> NDArray[complex]:
```

Genere dos señales de prueba con la función proporcionada arriba, ambos con las mismas amplitudes y frecuencias (al menos tres), pero uno con ruido y otro sin ruido.

Calcule la transformada de ambos y grafique su valor absoluto en función de la frecuencia. Elija un intervalo y un paso de frecuencia adecuado para visualizar todos los picos correspondientes a las frecuencias. Guarde la gráfica en un archivo llamado `1.a.pdf`

¿Qué le pasa a la transformada cuando el valor del ruido es similar al valor de las amplitudes?

Imprima su respuesta *en la consola* con el prefacio `"1.a)"`. No más de diez palabras.

1.b. Ancho de los picos

Genere una señal de prueba con una frecuencia conocida, sin ruido, y calcule el valor absoluto de la transformada como en el punto anterior.

Calcule el ancho a media altura (FWHM) del pico correspondiente a la frecuencia conocida. Puede usar `scipy.signal.peak_widths`, pero hay métodos mejores.

Varíe el intervalo de tiempo donde se miden los datos (`t_max`, o use partes del array) desde 10s hasta 300s.

Grafique el FWHM del pico como función de `t_max`, en escala log-log. Ajuste un modelo matemático a la tendencia que observa. Exporte una gráfica llamada `1.b.pdf` con este gráfico, donde indique qué modelo usó.

1.c. Aplicación

Aplique todo esto a datos experimentales reales: el **siguiente archivo** contiene las columnas de tiempo t, intensidad y y su incertidumbre σ_y.

Encuentre la frecuencia de Nyquist de los datos, es decir, la parte en la que empieza a reflejarse el espectrograma. Imprima en consola este valor con el prefacio `"1.c) f Nyquist:"`

Encuentre la frecuencia de oscilación de la señal f_{true}. Imprima en consola este valor con el prefacio `"1.c) f true:"`

Compruebe que su solución está bien calculando la *fase* $\varphi = \text{mod}(f_{\text{true}} t, 1)$, y graficando (scatter) y como función de φ. Guarde esa gráfica como `1.c.pdf`. Debería ser aproximadamente sinusoidal, con no demasiado ruido.

2. Transformada rápida

2.a. Comparativa

Para los datos de histéresis del taller 1 (proporcionado aquí como `H_field.csv`, esta vez con buen formato), encuentre la frecuencia de oscilación f_{fast} usando la transformada rápida `np.fft.rfft`. Recuerde que puede obtener las frecuencias con `np.fft.rfftfreq(n,Δt)`.

Compare este resultado con obtener la frecuencia mediante la transformada general f_{general} del punto 1. Imprima en consola ambas frecuencias de esta manera

```
print(f"2.a) {f_fast = :.5f}; {f_general = }")
```

Para ver la consecuencia de esta diferencia, calcule dos fases como antes:

$$\varphi_{\text{fast}} = \text{mod}(f_{\text{fast}} t, 1) \qquad \varphi_{\text{general}} = \text{mod}\big(f_{\text{general}} t, 1\big)$$

y en una misma gráfica (scatter) ponga H como función de φ_{fast} y H como función de φ_{general}. Guárdela como `2.a.pdf`

2.b. Manchas solares

Descargue el historial de avistamientos de manchas solares de la AAVSO del siguiente enlace.

Este es un archivo con año, mes, día y número de manchas solares en ese día. Se recomienda usar la función `pd.to_datetime` para graficar con las fechas.

Inlcuya en su análisis solamente los datos *hasta* el 1 de enero de 2010.

2.b.a. Período del ciclo solar

Encuentre el período del ciclo solar en años. Imprima `f'2.b.a) {P_solar = }'`

Recomiendo visualizar la gráfica de la transformada en ejes log-log.

2.b.b. Extrapolación

Use los primeros $n \approx 50$ armónicos de la señal (puede incluir más o menos, es su decisión) para predecir en qué parte del ciclo solar estamos. Concretamente, prediga el número de manchas solares desde 2012 hasta la fecha de entrega de esta tarea (10 Feb 2025)

Para esto, usando la transformada X_k obtenida con `rfft` y sus frecuencias f_k obtenidas con `rfftfreq`, use la transformada inversa:

$$y(t) = \mathfrak{Real}\left(\frac{1}{N} \sum_{k=0}^{M-1} \frac{X_k}{2} e^{2\pi i f_k t} \right)$$

Nótese que sumamos hasta $M = 50$ número de armónicos a considerar, pero dividimos por N, que es la longitud de los datos originales.

Aquí, t es el número (entero) de días desde el primer dato.

Imprima su predicción con: `f'2.b.b) {n_manchas_hoy = }'`

Grafique los datos en bruto junto a su predicción en `2.b.pdf`, donde el eje x debe tener las fechas.

Para saber si lo hizo bien, compare con las predicciones de otros modelos más complicados: http://solarcyclescience.com/forecasts.html y con los datos actuales https://www.spaceweatherlive.com/en/solar-activity.html

3. Filtros

3.a. Filtro gaussiano

Para los datos de manchas solares, implemente un filtro tipo Gaussiano, es decir:

$$\text{señal filtrada} = \text{IFFT}(\ \text{FFT}(\text{señal original}) \times \text{filtro}(f)\ )$$

El filtro en este caso debe ser

$$\text{filtro}(f, \alpha) = \exp(-(f\alpha)^2)$$

Grafique en varios subplots diferentes valores de α comparados con la señal original. Intente con varios valores, desde uno que no haga nada hasta uno que casi borre la señal.

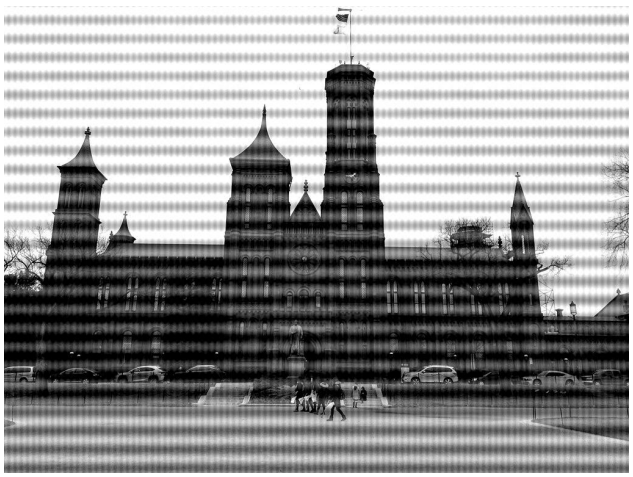
Los subplots deben ser:
- columna 1 la señal
- columna 2 la transformada

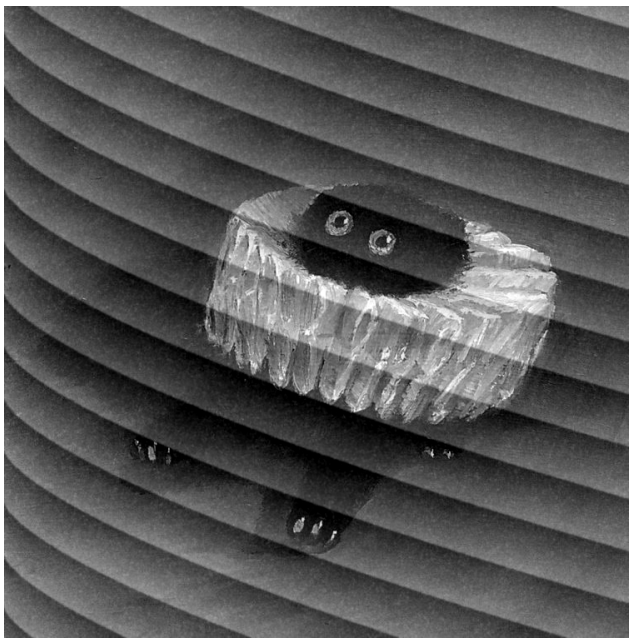
y en cada fila se va variando el α, indicando cuál valor se usa con `plt.text`. En todas las gráficas deben estar los datos de la señal original y de la señal filtrada.

Guarde esta gráfica como `3.1.pdf`

3.b. Limpieza de ruido periódico

Imagine que está tomando fotografías de baja calidad desde atrás de una persiana. Obtiene las siguientes imágenes, desde dentro hacia afuera de la ventana y viceversa:





Elimine el ruido periódico de ambas imágenes, y guárdelas como `3.b.a.png` y `3.b.b.png` respectivamente.