# Curso de Analise de Sinais

Analise espectral

# Parte 1

- Módulos fftpack e signal da biblioteca SciPy library.

  In [1]: from scipy import fftpack

  In [2]: from scipy import signal

- Também o módulo io.wavefile do SciPy para ler e escrever arquivos de audio WAV

  In [3]: import scipy.io.wavfile

  In [4]: from scipy import io

# Parte 1

- Para base numéricas e gráficos também vamos necessita das bibliotecas NumPy, Pandas, and Matplotlib:

```
In [5]: import numpy as np

In [6]: import pandas as pd

In [7]: import matplotlib.pyplot as plt

In [8]: import matplotlib as mpl
```

- A transformada de Fourier Discreta (DFT) de uma sequência uniformente espaçada

$$X_k = \sum_{n=0}^{N-1} x_n \mathrm{e}^{-2\pi nk/N}$$

- E a inversa de DFT

$$x_n = \frac{1}{N} \sum_{n=0}^{N-1} X_k \mathrm{e}^{2\pi nk/N}$$

- Onde X é TF discreta da amostra Xn, e k é o número de frequência que pode ser relacionado para frequência real

- O DFT para uma sequência de amostras pode ser calculado de forma muito eficiente utilizando o algoritmo conhecido como ***Fast Fourier Transform*** (FFT). O módulo fftpack do SciPy fornece implementações do algoritmo FFT.
- O módulo fftpack contém funções de FFT para uma variedade de casos:
- Aqui vamos nos concentrar em demonstrar o uso das funções FFT e IFFT, e várias das funções auxiliares no módulo fftpack.. No entanto, o uso geral é semelhante para todas as funções de FFT

| Function | Description |
|---|---|
| `fft`, `ifft` | General FFT and inverse FFT of a real- or complex-valued signal. The resulting frequency spectrum is complex valued. |
| `rfft`, `irfft` | FFT and inverse FFT of a real-valued signal. |
| `dct`, `idct` | The discrete cosine transform (DCT) and its inverse. |
| `dst`, `idst` | The discrete sine transform (DST) and its inverse. |
| `fft2`, `ifft2`, `fftn`, `ifftn` | The 2-dimensional and the n-dimensional FFT for complex-valued signals, and their inverses. |
| `fftshift`, `ifftshift`, `rfftshift`, `irfftshift` | Shift the frequency bins in the result vector produced by `fft` and `rfft`, respectively, so that the spectrum is arranged such that the zero-frequency component is in the middle of the array. |
| `fftfreq` | Calculate the frequencies corresponding to the FFT bins in the result returned by `fft`. |

- Note-se que o DFT toma de amostras discretas como entrada, e gera um espectro de frequência discreto. Para ser capaz de utilizar a DFT para processos que são originalmente contínua que primeiro devem reduzir os sinais para valores discretos utilizando amostragem.
- De acordo com o teorema de amostragem, um sinal contínuo com uma banda de frequência B (isto é, o sinal não conter frequências mais altas do que B), pode ser totalmente reconstruído a partir de amostras discretas com uma frequência de amostragem fs > 2B.

- Vamos simular um sinal com uma componente senoidal de 1 Hz e em 22 Hz, o topo da distribuição normal do ruido.
- Vamos calcular o espectro de frequência de sinal com frequência até 30 Hz.
- Vamos escolher uma frequência de amostragem fs = 60 Hz, e se queremos obter um espectro de frequência com resolução $\Delta f = 0.01$ Hz, nos precisamos coleta no mínimo $N = fs / \Delta f = 6000$ amostras, correspondente a período de amostragem de $T = N /fs = 100$ s
- Spectral_1.py

- Para ver as componente do sinal, vamos usar FFT para calcular o espectro do sinal (domínio da frequencia) Podemos obter a transformada de Fourier discreta do sinal aplicando a função fft de f_t
- F = fftpack.fft(f_t)

- F contem as frequências no espectro de frequências que são dados pela taxa de amostragem e número de amostras. Quando calculamos essas frequências, podemos usar a função fftfreq, que pega o número de amostras e o tempo de duração entre amostras como parâmetros e retorna um vetor de dados com o mesmo tamanho.
- f = fftpack.fftfreq(N, 1.0/f_s)

- A faixa de frequência por amplitude retorna valores pela função fft tanto positiva quanto negativa até a metade da taxa de frequência fs/2. Para valor do sinal real, o espectro é simétrico e então podemos checar somente o lado positivo da componente de frequência. Vamos criar uma mascara para usar somente a frequência positiva.
- Mask = np.where(f >=0)

# Parte 2

- spectral_2.py

# Filtro no domínio da frequência

- Para calcula o sinal do domínio do tempo para o domínio da frequência usamos a FFT inversa, a função **ifft.** Vamos aplicar a função ifft para o vetor F para reconstruir f_t. Modificando o espectro antes com o filtro no domínio da frequência. Por exemplo vamos separar somente as frequências abaixo de 2 Hz na soma espectral aplicando um filtro passa baixa que elimina alta frequências do sinal

  F_filtered = F * (abs(f) < 2)

  f_t_filtered = fftpack.ifft(F_filtered)
- Spectral_3.py

# janelamento

- Sempre aplicamos diretamente a FFT no sinal. Isto pode dar resultados aceitáveis, mas muitas vezes é possível melhorar ainda mais a qualidade e o contraste do espectro de frequência aplicando uma chamada função de janela ao sinal antes de aplicar a FFT.
- **Uma função de janela é uma função que, quando multiplicada pelo sinal, modula sua magnitude para que se aproxime de zero no início e no final**
- Duração da amostragem. Existem muitas funções possíveis que podem ser usadas como função de janela e o módulo de sinal SciPy fornece implementações de muitas funções de janela comuns, incluindo a função Blackman, a função Hann, a função Hamming, as funções de janela Gaussiana (com desvio padrão variável) e A função de janela Kaiser. Estas funções estão todas traçadas no script **spectral_4.py.**
- O Gráfico mostra que, embora todas essas funções janela são ligeiramente diferentes tem forma geral é muito semelhante.

# janelamento

- O objetivo principal das funções de janela é **reduzir o vazamento espectral** entre as frequências próximas, que ocorrem em computação discreta de transformada de Fourier quando o sinal contém componentes com períodos que não são exatamente divisíveis com o período de amostragem. Componentes de sinal com tais frequências podem, portanto, não se ajustar a um número completo de ciclos durante o período de amostragem,
- E uma vez que a TF discreta assume que o sinal é periódico, a descontinuidade resultante no limite do período pode dar origem a uma fuga espectral. Multiplicar o sinal com uma função de janela reduz esse problema. Alternativamente, poderíamos também aumentar o número de pontos de amostragem (aumentar o período de amostragem) para obter uma maior resolução de frequência, mas isso pode não ser sempre prático.
- Para ver como podemos usar uma função de janela antes de aplicar o FFT a um sinal de série de tempo, consideremos as medidas de temperatura ao ar livre disponíveis. Primeiro, usamos a biblioteca Pandas para carregar o conjunto de dados, remostrando-o para espaçamento uniforme. Também aplicamos o método fillna para eliminar quaisquer valores NaN no conjunto de dados.

# Janelamento

- Uma vez que o quadro de dados do Pandas foi criado e processado, precisamos dos arrays do NumPy subjacentes para poder processar os dados da série de tempo usando o módulo fftpack.
- Vamos aplicar uma função de janela aos dados na temperatura da matriz antes da FFT.
- Vamos usar função de janela Blackman, que é uma função de janela que é adequada para reduzir o vazamento espectral.
- Está disponível como a função blackman no módulo de sinal no SciPy. Como argumento para a função de janela, precisamos passar o comprimento da matriz de amostra, e ele retorna uma matriz com o mesmo comprimento:

# Janelamento

- Para aplicar a função janela simplesmente multiplicá-lo com a matriz contendo o sinal do domínio do tempo, e usar o resultado na computação FFT subsequente.
- No entanto, antes de prosseguir com a FFT para o sinal janelado de temperatura, primeiro traçar a série de tempo de temperatura original e a versão com janelas.
- O resultado é spectral_5.py. O resultado da multiplicação da série de tempo com a função de janela é um sinal que se aproxima de zero próximo dos limites do período de amostragem e, portanto, pode ser visto como uma função periódica com transições suaves entre limites de período e como tal a FFT do sinal em janela tem Mais bem-comportado propriedades.

# Janelamento

- Depois de ter preparado o sinal de janela, o resto da análise espectral prossegue como antes: Podemos usar a função **fft** para calcular o espectro e a função **fftfreq** para calcular as frequências correspondentes a cada intervalo de frequência.
- Selecionamos as frequências positivas criando uma matriz de máscaras a partir da matriz f e traçamos o espectro de frequência positiva resultante como mostrado no script spectral_6.py. O espectro apresentado mostra claramente picos na frequência que corresponde a um dia (1/86400 Hz) e seus harmônicos mais altos (2/86400 Hz, 3/86400 Hz, etc.).

# comentários

- Para obter o espectro mais preciso de um dado conjunto de amostras, é geralmente aconselhável aplicar uma função de janela ao sinal da série temporal antes de aplicar uma FFT.
- A maioria das funções de janela disponíveis no SciPy podem ser usadas de forma intercambiável, e a escolha da função de janela geralmente não é crítica.
- Uma opção popular é a função de janela Blackman, que é projetada para minimizar o vazamento espectral.

- Tudo o texto e scripts são traduzidos e adaptados do Livro

- Robert Johansson-Numerical Python. A Practical Techniques Approach for Industry-Apress (2015)