Лабораторная работа 8

Луняк Николай

19 апреля 2021 г.

Оглавление

Список иллюстраций

Листинги

Глава 1

Запуск `chap08.ipynb`

При увеличении `std` без увеличения `M` можно наблюдать, что наши «скачки» снова вернулись, как если бы мы взяли Boxcar-фильтр на `M` элементов. Действительно, при большом значении `std` нормальное распределение растягивается в time domain, сжимая при этом нормальное распределение, которое соответствует ему в frequency domain, делая его все более и более похожим на Boxcar-фильтр.

Глава 2

Исследование DFT нормального распределения

Сначала, как всегда, импорты.

```
from thinkdsp import Signal, Sinusoid, SquareSignal,
    TriangleSignal, SawtoothSignal, ParabolicSignal
from thinkdsp import normalize, unbias, PI2, decorate
from thinkdsp import Chirp
from thinkdsp import read_wave
from thinkdsp import Spectrum, Wave,
    UncorrelatedGaussianNoise, Spectrogram
from thinkdsp import Noise

import numpy as np
import pandas as pd

from matplotlib import pyplot as plt

import thinkstats2

from scipy.stats import linregress

import scipy
import scipy.fftpack

import scipy.signal

from ipywidgets import interact, interactive, fixed
import ipywidgets as widgets

loglog = dict(xscale='log', yscale='log')

```

```
PI2 = np.pi * 2
```

Листинг 2.1: Импорты

Теперь будем строить рядом нормальное распределение (сигнал «во времени») и его DFT. DFT будет как будто бы разрезано и разнесено в разные стороны, что решается при помощи `np.roll()`.

```
def plot_gaussian(std):
    M = 32
    gaussian = scipy.signal.gaussian(M=M, std=std)
    gaussian /= sum(gaussian)

    plt.subplot(1, 2, 1)
    plt.plot(gaussian)
    decorate(xlabel='Time')

    fft_gaussian = np.fft.fft(gaussian)
    fft_rolled = np.roll(fft_gaussian, M//2)

    plt.subplot(1, 2, 2)
    plt.plot(np.abs(fft_rolled))
    decorate(xlabel='Frequency')
    plt.show()

slider = widgets.FloatSlider(min=0.1, max=10, value=2)
interact(plot_gaussian, std=slider)
```

Листинг 2.2: Импорты

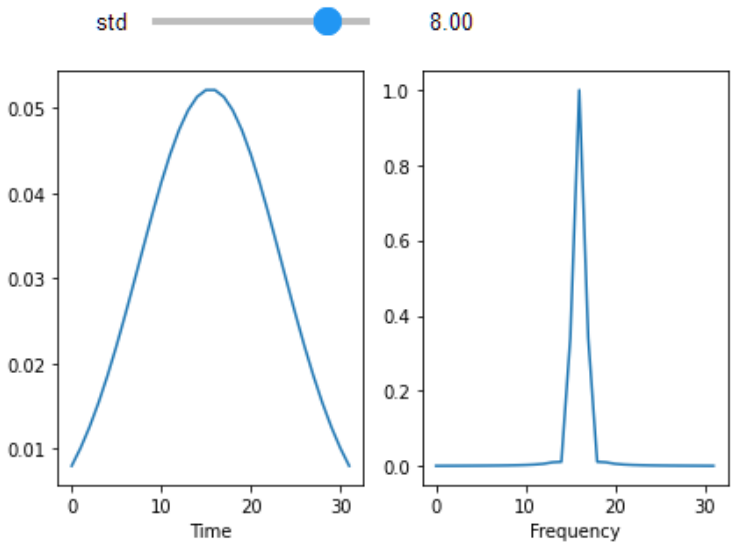

Рис. 2.1: Сравнение

Как я и писал в прошлом разделе, сужение одного «колокола» ведет к расширению другого.

Глава 3

Сравнение разных окон

В рамках данного пункта постараемся сравнить между собой разные окна, которые мы рассматривали в третьей лабораторной работе.

```
signal = SquareSignal(freq=440)
wave = signal.make_wave(duration=1.0, framerate=44100)

M = 15
std = 2.5

gaussian = scipy.signal.gaussian(M=M, std=std)
bartlett = np.bartlett(M)
blackman = np.blackman(M)
hamming = np.hamming(M)
hanning = np.hanning(M)

windows = [blackman, gaussian, hanning, hamming]
names = ['blackman', 'gaussian', 'hanning', 'hamming']

for window in windows:
    window /= sum(window)

for window, name in zip(windows, names):
    plt.plot(window, label=name)

decorate(xlabel='Index')
```

Листинг 3.1: Создание окон

Нарисуем DFT каждого из них на одном графике.

```
def zero_pad(array, n):
    """Extends an array with zeros.

    array: NumPy array
    n: length of result
```

```
6
7       returns: new NumPy array
8       """
9       res = np.zeros(n)
10      res[:len(array)] = array
11      return res
12
13  def plot_window_dfts(windows, names):
14      for window, name in zip(windows, names):
15          padded =  zero_pad(window, len(wave))
16          dft_window = np.fft.rfft(padded)
17          plt.plot(abs(dft_window), label=name)
18
19  plot_window_dfts(windows, names)
20  decorate(xlabel='Frequency (Hz)')
```

Листинг 3.2: DFT

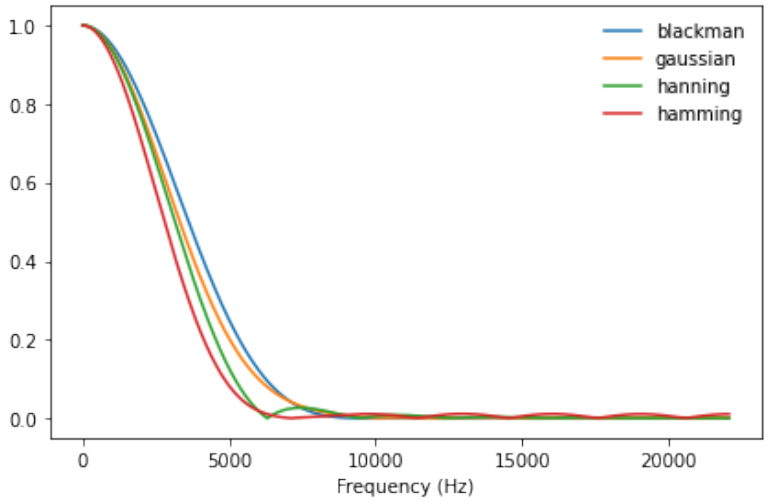

Рис. 3.1: DFT

Различия есть в отскоках, но их сложно заметить, поэтому отобразим то же самое в логарифмическом масштабе через

```
1  plot_window_dfts(windows, names)
2  decorate(xlabel='Frequency (Hz)', yscale='log')
```

Листинг 3.3: DFT в логарифмическом масштабе

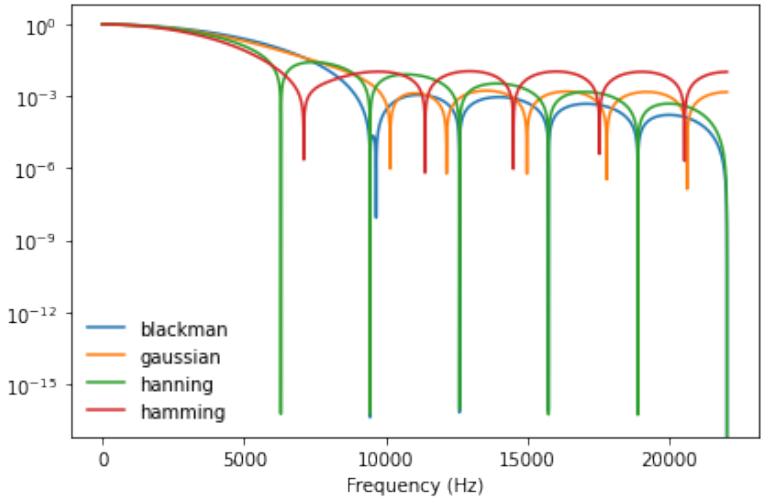

Рис. 3.2: DFT в логарифмическом масштабе

Среди всех самые «низкие» отскоки вышли у Хэннинга. При этом, он и Блэкмен продолжают уменьшаться, в то время как Хэмминг, несмотря на скачки, вроде как, держится на одном уровне (хотя поначалу он уменьшался быстрее всех остальных).