Разработка пакета программ для исследования статистических характеристик сигналов сердечно-сосудистой системы средствами языка программирования Python 3

Лапшева Е.Е. (СГУ), Пономаренко В.И. (СФ ИРЭ РАН)

Описание данных

Группы исследуемых

1. Здоровые люди (13 чел.)
2. Пациенты с собственным ритмом сердца до постановки ЭКС (15 чел.)
3. Пациенты с ЭКС со временем работы 40 – 100% (33 чел.)

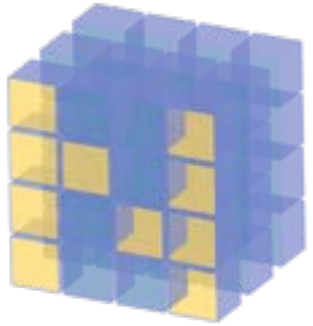
Реовазографические исследования

- ЭКГ
- Абдоминальное дыхание
- Фотоплетизмограмма, левое ухо
- Реограмма, левое плечо
- Реограмма, правое плечо
- Фотоплетизмограмма, палец левой руки
- Фотоплетизмограмма, палец правой руки

- 250 Гц, длительность ~10 мин
- разрешение 16 бит

Используемое ПО

- Для проведения анализа описанных данных использовался язык программирования Python 3.5 в сборке WinPython.
- NumPy – работа с большими многомерными массивами с использованием высокоуровневых математических функций.
- SciPy – научные и инженерные расчёты.
- MatPlotLib – визуализация данных в типографском качестве.



Цифровой спектральный анализ. Прямое преобразование Фурье

Прямое преобразование Фурье имеет вид:

$$X(f) = \int_{-\infty}^{+\infty} x(t) e^{-j2\pi f t} dt,$$

где $x(t)$ – некоторая функция, j – мнимая единица, f – переменная частота (Гц).

$$e^{-j2\pi f t} = \cos(2\pi f t) + j \cdot \sin(2\pi f t)$$

При обработке цифрового сигнала с использованием компьютера, переменная частоты будет дискретна.

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{-j\frac{2\pi}{N}kn}, k = 0, 1, 2, \dots, N-1,$$

где k – индекс отсчета по частоте, а N – число отсчетов одного периода спектра.

Цифровой спектральный анализ. Обратное преобразование Фурье

Обратное преобразование Фурье можно интерпретировать как *синтез* сигнала $x(t)$:

$$x(t) = \int_{-\infty}^{+\infty} X(f) e^{j2\pi f t} df$$

Выражение для обратного дискретного преобразования Фурье задаётся уравнением:

$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} X(k) e^{j\frac{2\pi}{N}kn}, n = 0, 1, 2, \dots, N-1.$$

Программная реализация

- Для нахождения дискретного преобразования Фурье в настоящем исследовании использовался метод **`rfft`** из библиотеки **`NumPy FFT`**, где используется следующий способ вычисления: вычисляются первые $\left[\frac{n}{2}\right] + 1$ точек n-точечного дискретного преобразования Фурье. Возвращаемое значение – первая половина дискретного преобразования Фурье, соответствующая положительным частотам.
- Обратное преобразование Фурье вычислялось при помощи метода **`irfft`**, вычисляющий n-точечное обратное преобразование Фурье, используя первые $\left[\frac{n}{2}\right] + 1$ точек. Очевидно, что **`irfft(rfft(x)) == x`**.
- Для нахождения частотной области исследуемого сигнала использовался метод **`rfftfreq`**, возвращающий частоты дискретного преобразования Фурье.

numpy.fft.rfft

numpy.fft.**rfft**(*a*, *n=None*, *axis=-1*, *norm=None*) [source]

Compute the one-dimensional discrete Fourier Transform for real input.

This function computes the one-dimensional *n*-point discrete Fourier Transform (DFT) of a real-valued array by means of an efficient algorithm called the Fast Fourier Transform (FFT).

Parameters:

a : *array_like*
Input array

n : *int, optional*
Number of points along transformation axis in the input to use. If *n* is smaller than the length of the input, the input is cropped. If it is larger, the input is padded with zeros. If *n* is not given, the length of the input along the axis specified by *axis* is used.

axis : *int, optional*
Axis over which to compute the FFT. If not given, the last axis is used.

norm : *{None, "ortho"}, optional*
New in version 1.10.0.
Normalization mode (see numpy.fft). Default is None.

Returns:

out : *complex ndarray*
The truncated or zero-padded input, transformed along the axis indicated by *axis*, or the last one if *axis* is not specified. If *n* is even, the length of the transformed axis is (n/2)+1. If *n* is odd, the length is (n+1)/2.

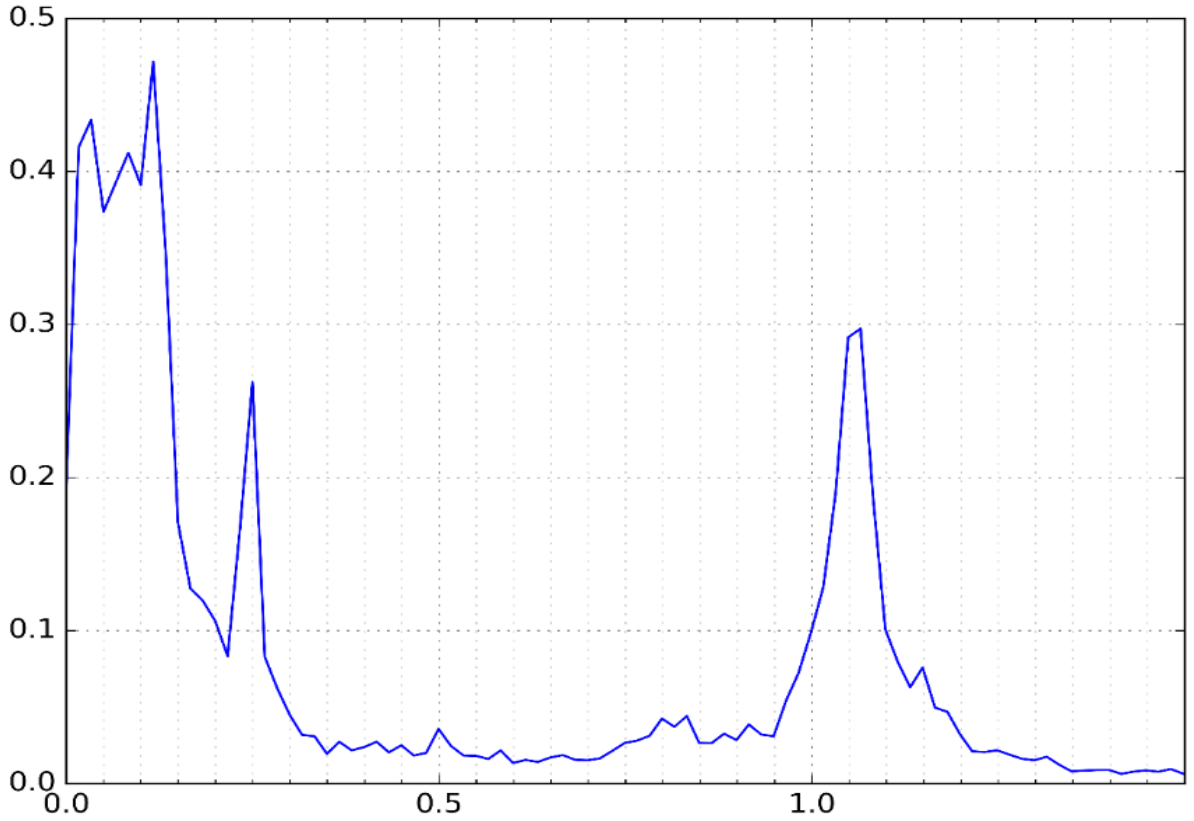
Raises:

IndexError
If *axis* is larger than the last axis of *a*.

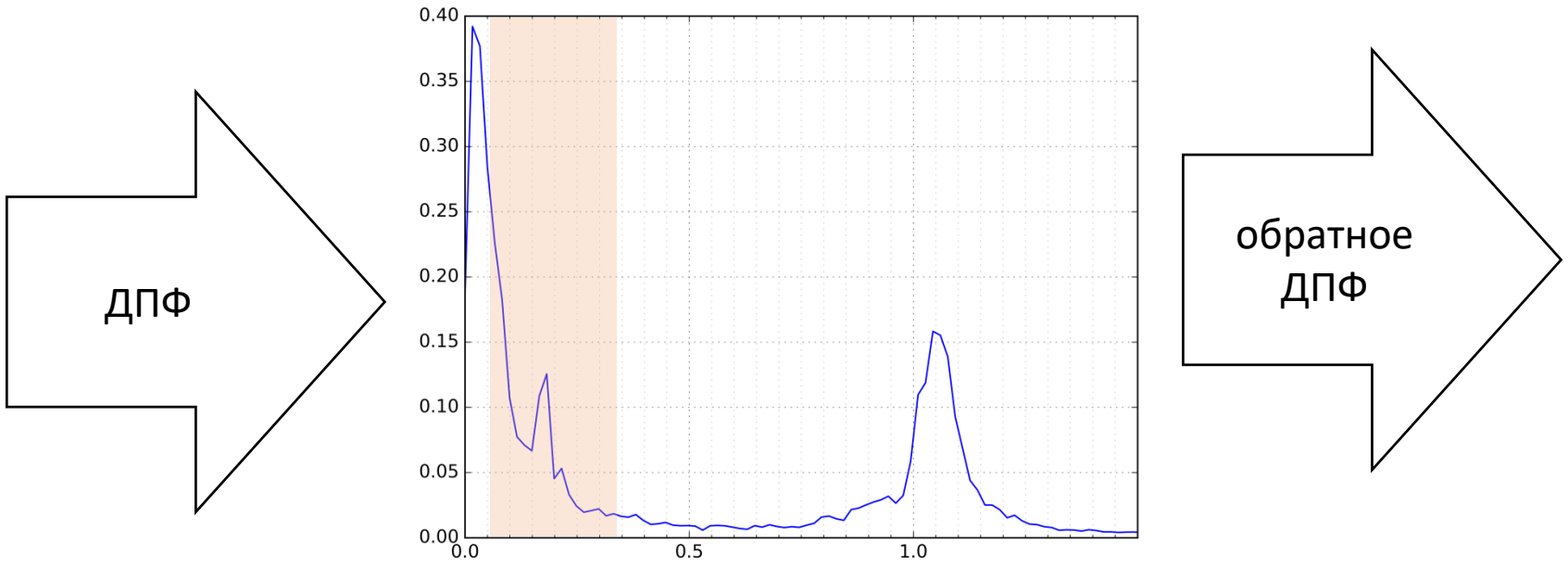
See also:

numpy.fft	For definition of the DFT and conventions used.
irfft	The inverse of rfft.
fft	The one-dimensional FFT of general (complex) input.
fftn	The *n*-dimensional FFT.
rfftn	The *n*-dimensional FFT of real input.

Спектр Фурье ФПГ пальца левой руки

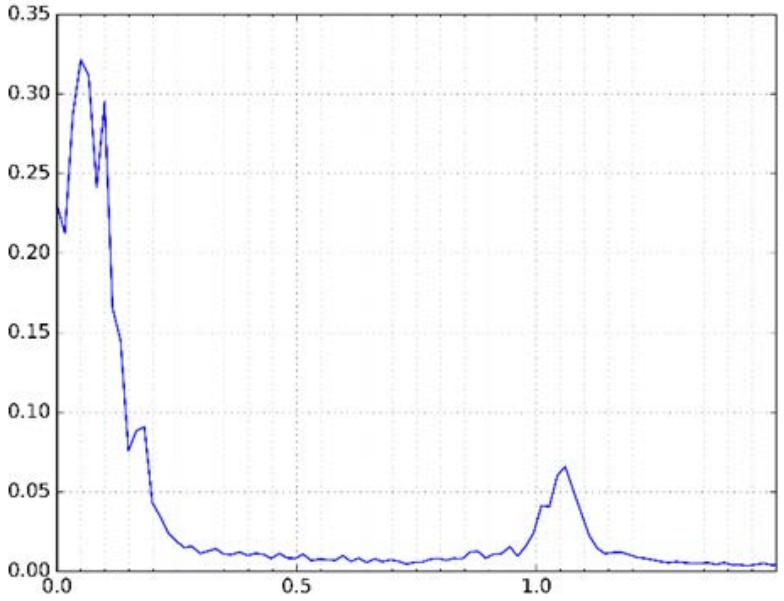


Частотная фильтрация на основе ДПФ

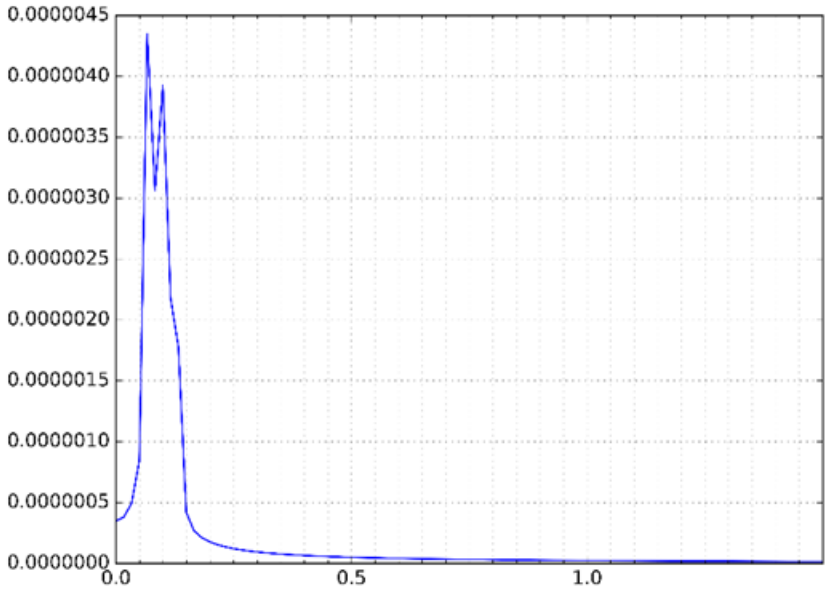


Результат частотной фильтрации

Спектр нефильтрованного сигнала



Спектр фильтрованного сигнала



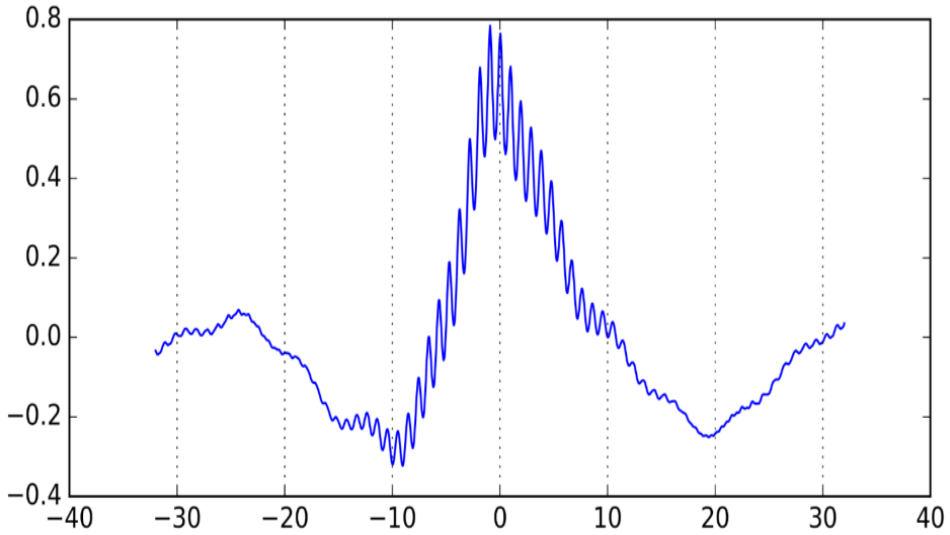
Корреляция

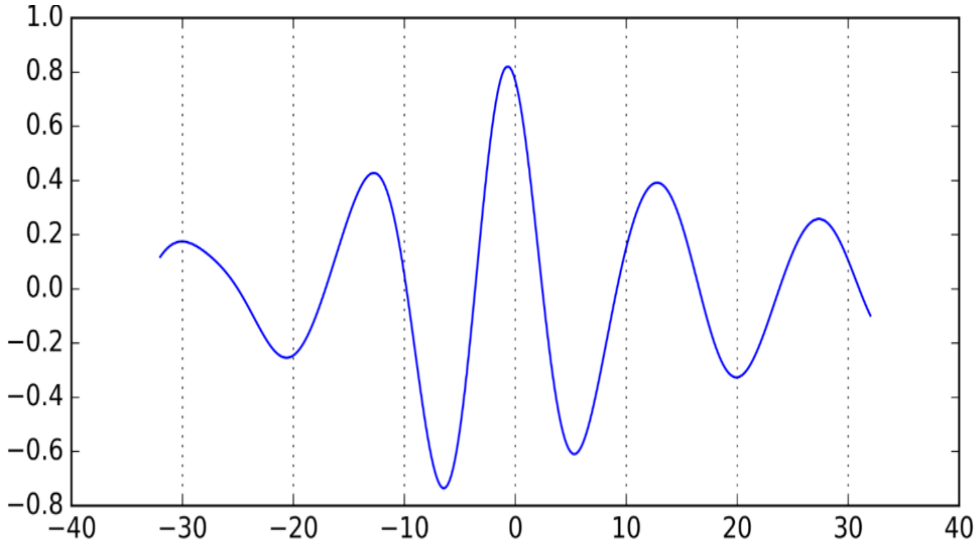
Метод **`correlate`** из библиотеки **`NumPy`**, где используется следующий способ вычисления:

$$c_{xy}[k] = \sum_n x[n+k] \cdot \overline{y[n]},$$

где $\overline{y[k]}$ – сопряженное комплексное число.

Корреляция двух сигналов ФПГ пальцев на левой и правой руках





Когерентность

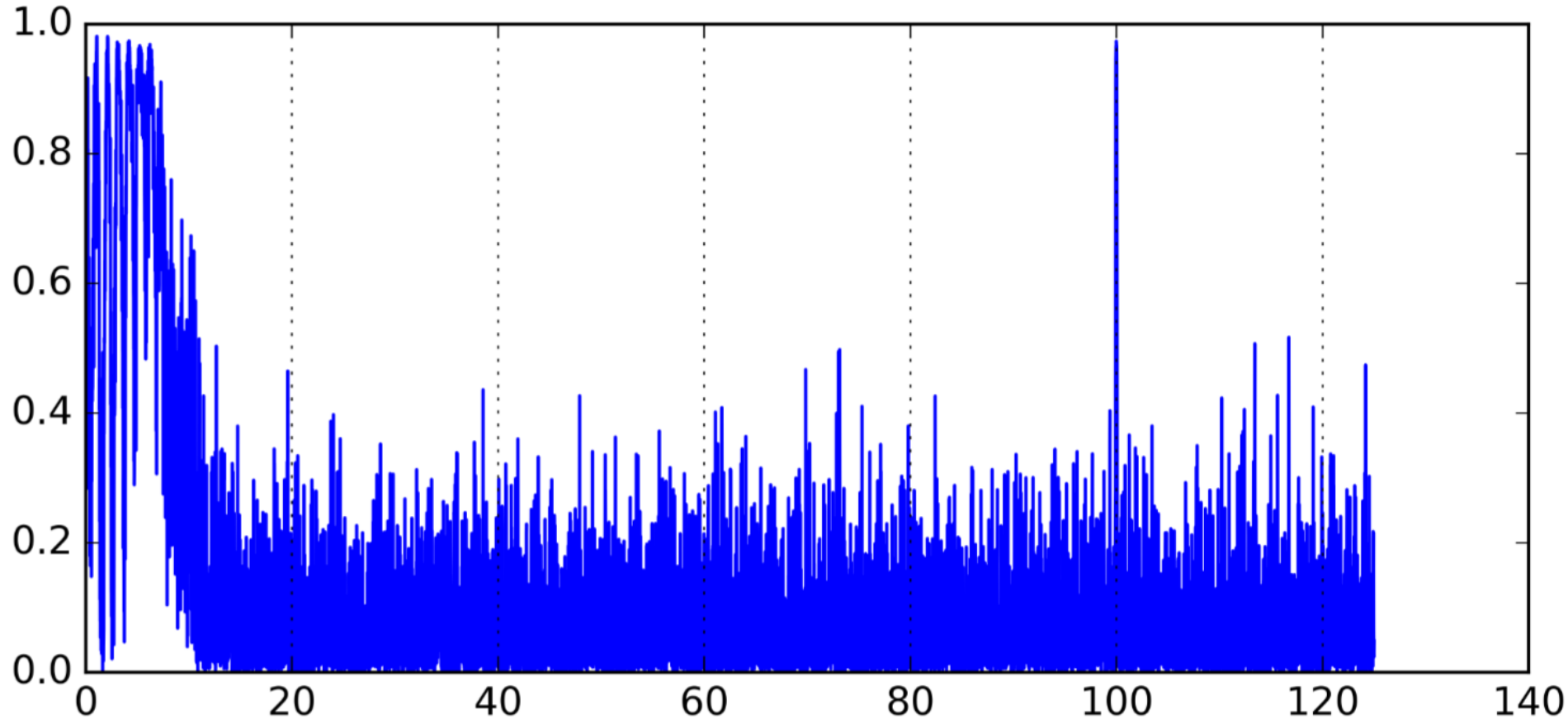
Аналог функции корреляции в частотной области. Когерентность отражает линейную зависимость гармонических составляющих исследуемых процессов. Чем ближе функция когерентности к единице на какой-либо частоте, тем выше совпадение гармонических составляющих на этой частоте.

Метод **`coherence`** из библиотеки **`SciPy Sygnal`**. Здесь спектральная плотность вычисляется с использованием периодограммного метода Уэлча. При этом исходный сигнал делится на ряд перекрывающихся сегментов, на каждом из которых выполняется быстрое преобразование Фурье. Математическое ожидание вычисляется путем усреднения полученных таким образом выборочных спектров.

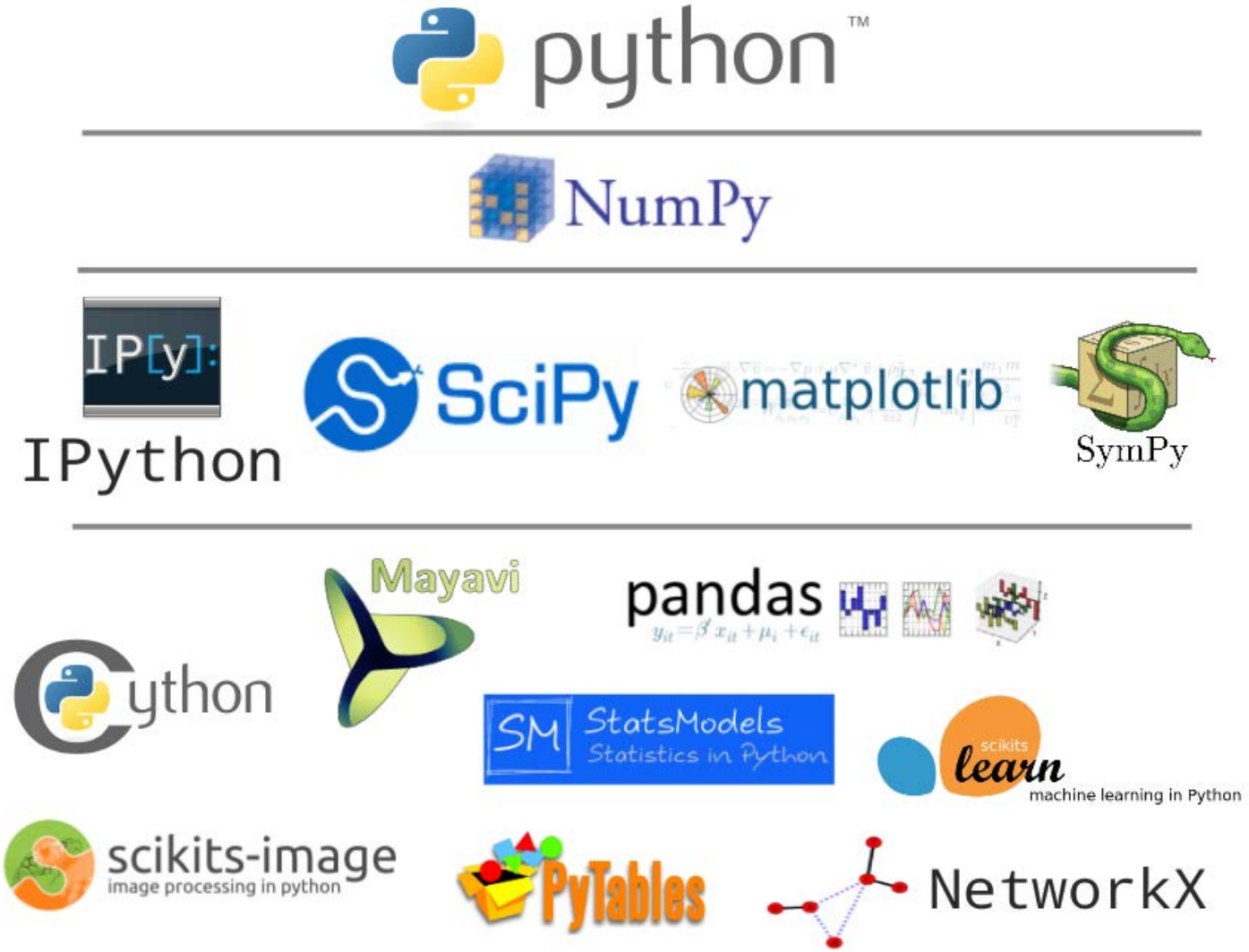
При выборе сегмента на сигнал накладывается окно Ханна:

$$w(n) = 0.5 - 0.5 \cdot \cos\left(\frac{2\pi n}{M-1}\right), \qquad 0 \le n \le M.$$

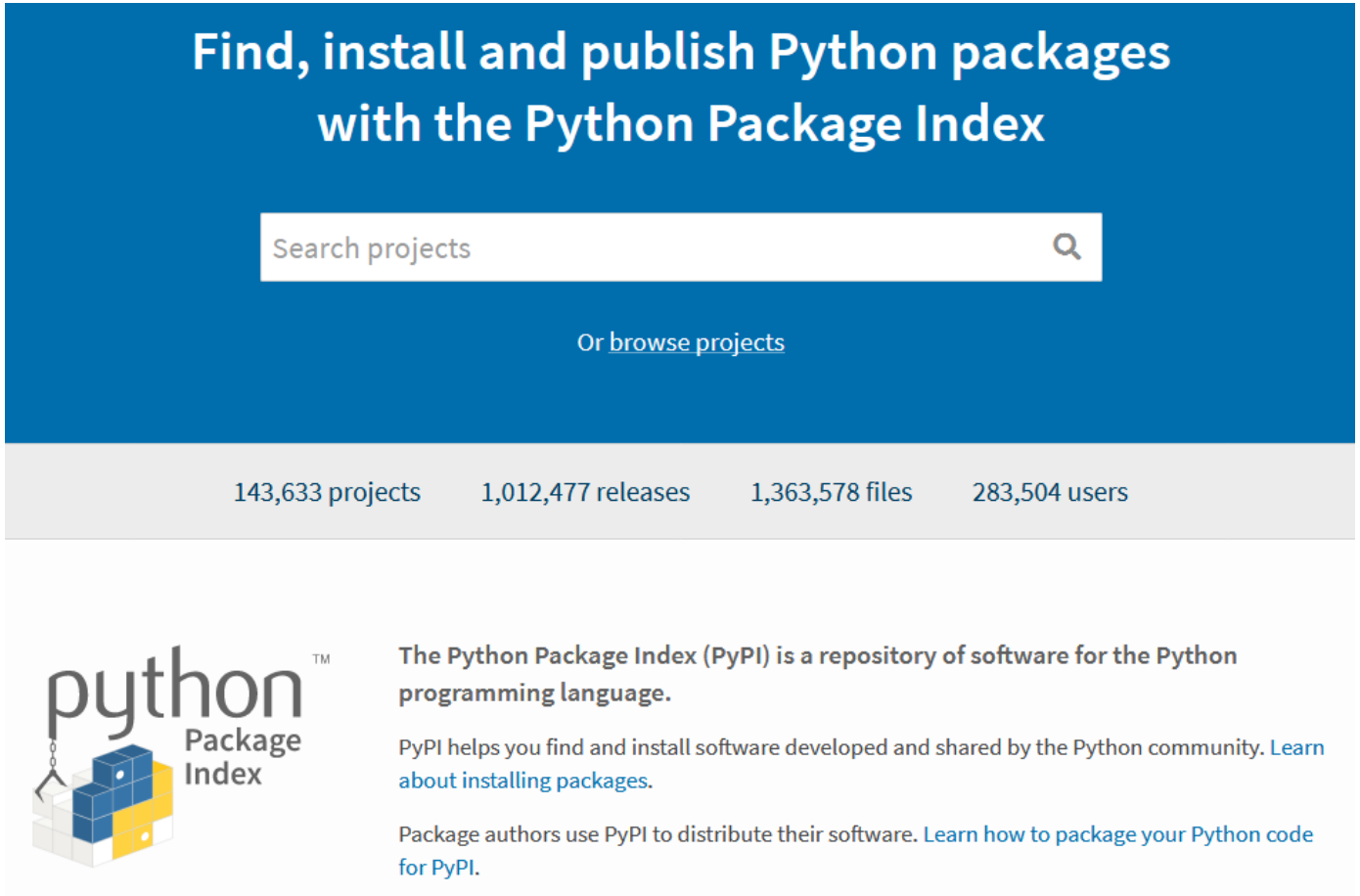
Когерентность сигналов ФПГ пальцев на правой и левой руках



Экосистема научного Python



Репозиторий Python https://pypi.org/



Спасибо за внимание!