Теорема Котельникова (Nyquist–Shannon). Преобразование Фурье. Прямое. Обратное.

Ruslan V. Akhpashev - 2021, e-mail: ruslan.akhpashev@gmail.com

Основной источник информации: https://github.com/capitanov

Частота дискретизации. Теорема Котельникова.

Периодический сигнал - это сигнал, который повторяется во времени с определенным периодом, то есть для которого выполняется условие:

$$s(t) = s(t + kT),$$

где *k* – любое целое число, *T* – период повторения.

Пример *периодического* сигнала – гармоническое колебание, описываемое следующим выражением:

$$s(t) = A \cdot cos(\frac{2\pi \cdot t}{T} + \phi),$$

где *A* – амплитуда колебания, φ – начальная фаза.

Известно, что любой сложный периодический сигнал может быть представлен в виде суммы гармонических колебаний с частотами, кратными основной частоте **ω = 2π/T**.

Цифровые сигналы

Все сигналы можно разделить на четыре группы:

- аналоговые,
- дискретные,
- квантованные,
- цифровые.

Аналоговый сигнал – описывается непрерывной функцией времени. Аналоговый сигнал обеспечивает передачу данных путем непрерывного изменения во времени амплитуды, частоты или фазы. Практически все физические процессы описываются непрерывными функциями времени, поэтому представляют собой аналоговые сигналы. Для аналогового сигнала область значений и определения описывается *непрерывным множеством*.

Для **дискретного** сигнала свойственно прерывистое (дискретное) изменение сигнала во времени. То есть изменения в сигнале происходят скачкообразно через некоторые промежутки времени, называемые интервалом дискретизации – Δt или Td. Дискретизация *аналогового сигнала* состоит в том, что сигнал представляется в виде

последовательности значений, взятых в дискретные моменты времени, которые называются *отсчётами* (сэмплами).

Для правильного восстановления аналогового сигнала из цифрового без искажений и потерь используется теорема отсчетов, известная как **теорема Котельникова (Найквиста-Шеннона)**.

> Любой непрерывный сигнал с ограниченным спектром может быть восстановлен однозначно и без потерь по своим дискретным отсчетам, взятым с частотой строго больше удвоенной верхней частоты спектра непрерывного сигнала.

Формула теоремы Котельникова:

$F_s = \frac{1}{T_s} > 2F_a,$

где

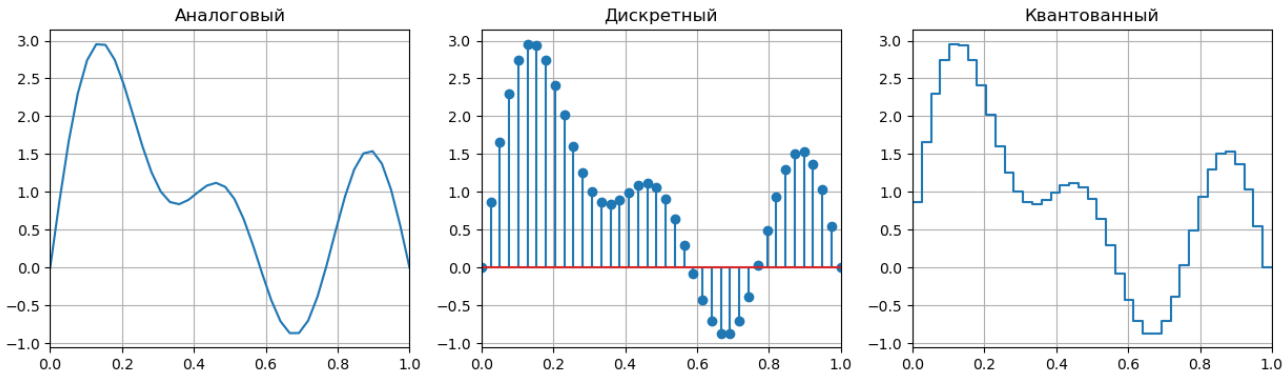
- F_s - частота дискретизации сигнала,
- F_a - верхняя частота спектра аналогового сигнала.

Такое определение относится к функциям времени, которые состоят из частот от нуля до F_a.

Пример сигналов

```
import numpy as np
import matplotlib.pyplot as plt

# Формирование сигнала
n = 40 # number количество временных отсчетов
time = np.linspace(0, 1, n, endpoint=True) # 1 секунда, в которой "n" отсчетов
signal = np.sin(np.pi*time) + np.sin(2*np.pi*time) + np.sin(3*np.pi*time) + np.sin(5*np.pi

# Построение графиков
fig = plt.figure(figsize=(15, 4), dpi=100)
plt.subplot(1, 3, 1)
plt.title("Аналоговый")
plt.plot(time, signal)
plt.xlim([0, 1])
plt.yticks(np.linspace(np.floor(np.min(signal)), np.ceil(np.max(signal)), 9))
plt.grid(True)

plt.subplot(1, 3, 2)
plt.title("Дискретный")
plt.stem(time, signal)
plt.xlim([0, 1])
plt.yticks(np.linspace(np.floor(np.min(signal)), np.ceil(np.max(signal)), 9))
plt.grid(True)
```

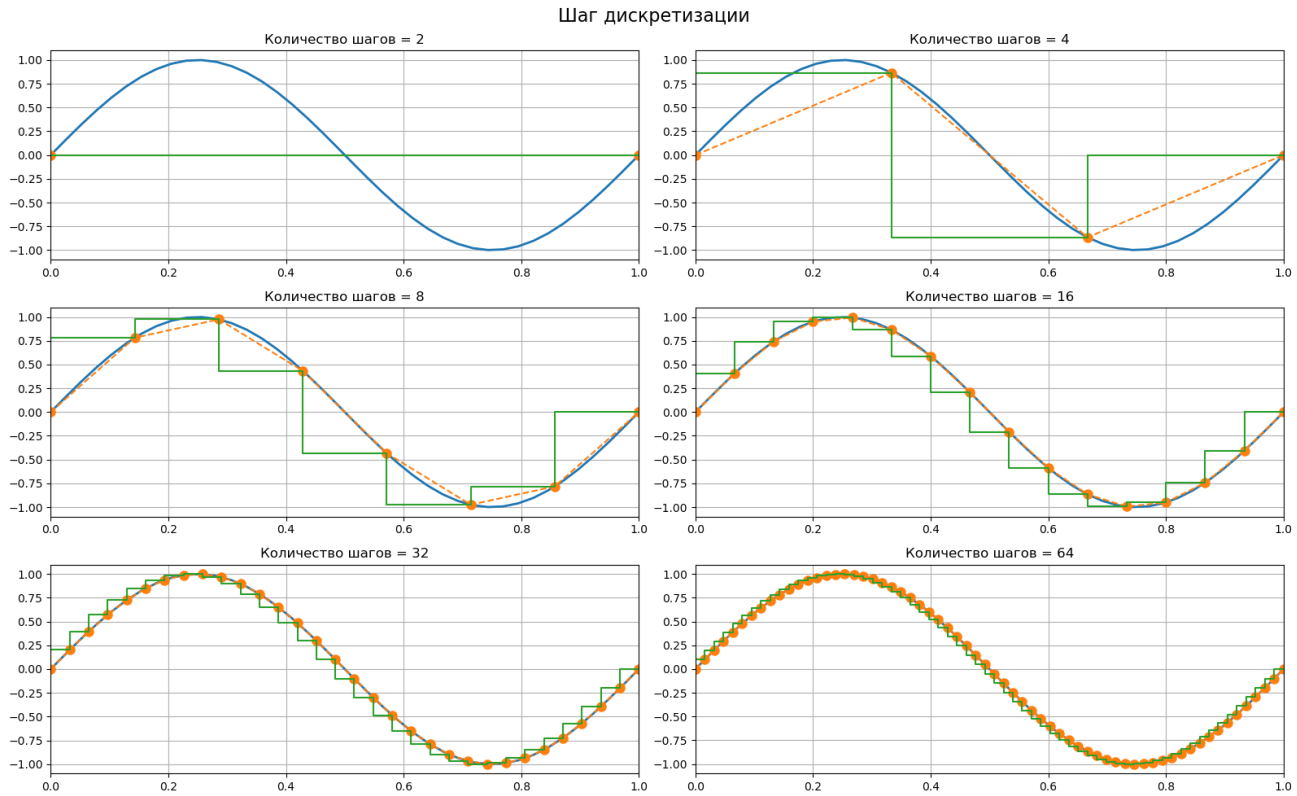
```
plt.subplot(1, 3, 3)
plt.title("Квантованный")
plt.step(time, signal)
plt.xlim([0, 1])
plt.yticks(np.linspace(np.floor(np.min(signal)), np.ceil(np.max(signal)), 9))
plt.grid(True)
```



Шаг дискретизации. Наглядно.

```
n = 40
time = np.linspace(0, 1, n, endpoint=True)
frequency = [1, 2, 4, 8]
freq_index = 0
signal = np.sin(2*np.pi*time*frequency[freq_index])


# discrete step
discrete_steps = [2, 4, 8, 16, 32, 64]

#plot figure
fig = plt.figure(figsize=(16, 10), dpi=100)
plt.suptitle("Шаг дискретизации", fontsize=16)
i = 0
for step in discrete_steps:
    discrete_time = np.linspace(0, 1, step, endpoint=True)
    plt.subplot(3, 2, i+1)
    plt.title('Количество шагов = {}'.format(step))
    plt.plot(time, signal, '-', linewidth=2.0)
    plt.plot(discrete_time, np.sin(2*np.pi*discrete_time*frequency[0]), '--o', linewidth=1
    plt.step(discrete_time, np.sin(2*np.pi*discrete_time*frequency[0]), linewidth=1.5)
    plt.grid()
    plt.xlim([0, 1])
    i += 1
```

```
plt.tight_layout()
```



Преобразование Фурье. Прямое. Обратное.

Спектр сигнала

Из предыдущей части вы узнали, что *сигнал* - это физический процесс во времени, параметры которого изменяются в соответствии с передаваемым сообщением. Мы научились представлять дискретные (цифровые) сигналы во времени. В этом разделе будет показан переход между временной и частотной областями для дискретных сигналов.

Прямое преобразование Фурье

Чтобы преобразовать сигнал из временной области в частотную и обратно необходимо выполнить операцию под названием дискретное **преобразование Фурье**.

Запишем формулу прямого преобразования Фурье для дискретной последовательности x(nT). Прямым дискретным преобразованием Фурье (ДПФ) называется преобразование последовательности `x(n), n = 0, ... , N–1` в последовательность `X(k), k = 0, ..., N–1` по следующей формуле:

$X(k) = \sum_{n=0}^{N-1} x(nT) \cdot e^{(-2\pi j \cdot nk/N)} = \sum_{n=0}^{N-1} x(nT) \cdot W^{-nk}$

где $k = 0, \ldots, N-1$.

- N – количество компонент разложения, число измеренных за период значений сигнала;
- n – номер отсчета дискретизированного сигнала, n = 0,1, ... , N–1;
- k – номер гармоники компонента преобразования, а T — период времени, в течение которого брались входные данные;
- $W = e^{-2\pi j/N}$ – поворотный множитель.

В этой формуле $X(kT) = X(e^{j\omega T})$ является спектральной плотностью (спектром) дискретной последовательности. Выражение для спектра дискретной последовательности можно найти, заменив в её Z-форме переменную $z = e^{j\omega T}$

Для аналоговых сигналов выражение суммы превращается в интеграл.

Используя **формулу Эйлера** $e^{j\omega T} = cos(\omega T) + j \cdot sin(\omega T)$, можно определить *вещественную* и *мнимую* составляющие, а также **модуль** и **аргумент** спектральной плотности, которые связаны с вещественной и мнимой частями спектра через формулы теории функции комплексного переменного.

Модуль:

$|X(kT)| = \sqrt{Re(X)^2 + Im(X)^2}$

Фаза:

$arg(X(kT)) = \arctan \frac{Im(X)}{Re(X)}$

Таким образом, ДПФ для N входных отсчетов сигнала ставит в соответствие N спектральных отсчётов. Из формулы ДПФ для вычисления одного спектрального отсчета требуется N операций комплексного умножения и сложения. Поскольку таких операций N, то общая вычислительная сложность ДПФ равна N^2

Обратное преобразование Фурье

Обратное дискретное преобразование Фурье (ОДПФ) есть перевод последовательности `X(k), k = 0, ..., N-1` в последовательность `x(n), n = 0, ... , N-1` по формуле:

$$x(nT) = \frac{1}{N}\sum_{n=0}^{N-1} X(k) \cdot e^{(2\pi j \cdot nk/N)} = \frac{1}{N}\sum_{n=0}^{N-1} X(k) \cdot W^{nk}$$

где `x(n)` – измеренная последовательность в дискретных временных точках, значения которой являются исходными данными для прямого преобразования и выходными для обратного `X(k)` – N–последовательность комплексных амплитуд синусоидальных сигналов, образующих исходный сигнал `x(n)`; значения последовательности являются выходными данными для *прямого* преобразования и входными для *обратного*

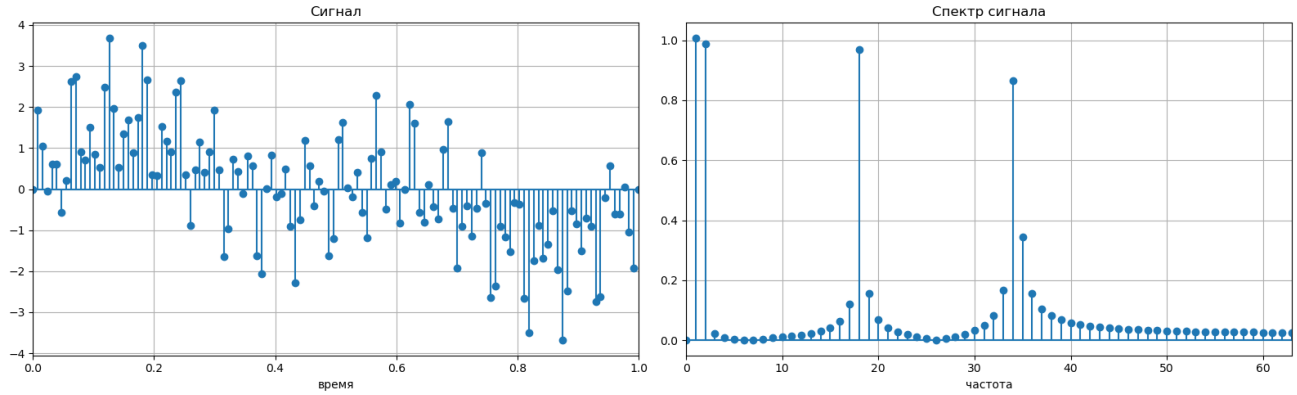
Поскольку амплитуды спектральных отсчетов - комплексные величины, то по ним можно вычислить одновременно и амплитуду, и фазу сигнала.

Как следует из теоремы **Найквиста-Котельникова**, ДПФ точно соответствует непрерывному преобразованию Фурье, если преобразуемая функция есть функция с ограниченным спектром, при этом частота дискретизации **Fд** должна быть не меньше удвоенной максимальной частоты спектра **Fв**.

Сгенерирует произвольный сигнал, состоящий из четырех гармоник:

```
import matplotlib

n = 128
f = [1, 2, 34, 18]
A = [1, 3, 4, 2]
t = np.linspace(0, 1, n, endpoint=True)
# Генерация произвольного сигнала
signal = A[0]*np.sin(2*np.pi*t*f[0]) + A[0]*np.sin(2*np.pi*t*f[1]) + A[0]*np.sin(2*np.pi*t

# Расчет преобразования Фурье
signalFFT = np.fft.fft(signal)
signalFFTabs = 2*np.abs(signalFFT) / n


# Построение графика
fig = plt.figure(figsize=(16, 5), dpi=100)
# График: сигнал во времени
plt.subplot(1, 2, 1)
plt.title('Сигнал')
plt.stem(t, signal, basefmt='C0')
plt.xlim([0, t[len(t)-1]])
plt.xlabel('время')
plt.grid()

# График: спектр сигнала
plt.subplot(1, 2, 2)
plt.title('Спектр сигнала')
```

```
plt.stem(signalFFTabs, basefmt='C0')
plt.xlim([0, n//2-1])
plt.xlabel('частота')
plt.grid()
plt.tight_layout()
```



Из результата видно, что на графике с частотной составляющей после преобразования Фурье видны пики амплитуд, которые относятся к определенным частотам. Если приглядется внимательно, то значения частот будут совпадать с частотами гармоник, которые мы использовали.

Задание на лабораторную работу №3

Необходимо:

1) Из лабораторной №1 взять за основу сигнал, сгенерированный вами.

2) С помощью преобразования Фурье получить спектр сигнала из пункта (1). Построить график спектральной составляющей.

3) Произвести обратное преобразование фурье для, получившейся спектральной составляющей. Сравнить с оригиналом в **пункте 1**.

4) Записать с помощью микрофона свой голос (ниже показан пример).

5) Проанализировать влияние частоты **семплирования** на качество воспроизводимого звука. Сделать выводы.

6) Получить частотный спектр, записанного звука с помощью преобразования Фурье.

7) Составить отчет.

8) Для доп. баллов реализовать функцию преобразования Фурье (Дискретное преобразование Фурье сложностью самостоятельно.

Запись и воспроизведение звука

```
import sounddevice as sd

fs=100000
duration = 3  # seconds
myrecording = sd.rec(duration * fs, samplerate=fs, channels=1,dtype='float64')
myrecording = myrecording.reshape(myrecording.size,)
print ("Recording Audio")
sd.wait()
print ("Audio recording complete , Play Audio")
```

```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
<ipython-input-1-a87a0b6b24da> in <module>()
----> 1 import sounddevice as sd
      2
      3 fs=100000
      4 duration = 3  # seconds
      5 myrecording = sd.rec(duration * fs, samplerate=fs,
channels=1,dtype='float64')

ModuleNotFoundError: No module named 'sounddevice'

---------------------------------------------------------------------------
NOTE: If your import is failing due to a missing package, you can
manually install dependencies using either !pip or !apt.

To view examples of installing some common dependencies, click the
"Open Examples" button below.
---------------------------------------------------------------------------
```

SEARCH STACK OVERFLOW