

Лабораторная работа 7

Чернышев Ярослав

31 мая 2021 г.

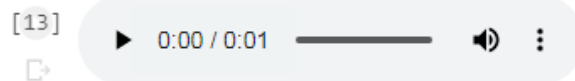
Оглавление

1	Задание 7.1	2
2	Задание 7.2	3

Глава 1

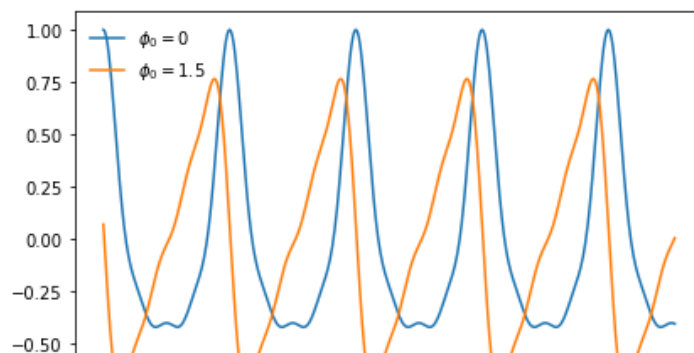
Задание 7.1

В данном задании требуется ознакомиться и запустить примеры из chap07.ipynb.
Фрагмент работы:



To see the effect of a complex amplitude, we can rotate the amplitudes by 1.5 radian:

```
phi = 1.5  
amps2 = amps * np.exp(1j * phi)  
ys2 = synthesize2(amps2, freqs, ts)  
  
n = 500  
plt.plot(ts[:n], ys.real[:n], label=r'$\phi_0 = 0$')  
plt.plot(ts[:n], ys2.real[:n], label=r'$\phi_0 = 1.5$')  
decorate(xlabel='Time')
```



Глава 2

Задание 7.2

В этом задании требуется исследовать и реализовать быстрое преобразование Фурье, взяв за основу дискретное преобразование Фурье и лемму Дэниэлсона-Ланцоша:

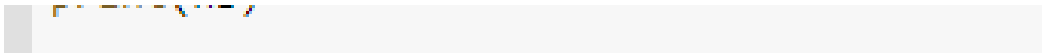
$$DFT(y)[n] = DFT(e)[n] + e^{-\frac{2\pi i n}{N}} \cdot DFT(o)[n]$$

Реализация заключается в делении массива значений некоторого сигнала на подмассивы с четными и нечетными элементами, вычислении ДПФ для них и применении к каждому значению приведенной выше леммы. Завершается алгоритм тогда, когда деление у дойдет до массива в 1 элемент.

Зададим исходный сигнал и вычисляем его FFT:

```
1 ys = [-0.4, 0.5, 0.1, -0.8]
2 hs = np.fft.fft(ys)
3 print(hs)
```

Результат его выполнения:



```
[-0.6+0.j -0.5-1.3j  0. +0.j -0.5+1.3j]
```

Для последующего примера, ниже приведён нерекурсивный вариант алгоритма:

```
1 def dft(ys):
2     N = len(ys)
```

```

3     ts = np.arange(N) / N
4     freqs = np.arange(N)
5     args = np.outer(ts, freqs)
6     M = np.exp(1j * PI2 * args)
7     amps = M.conj().transpose().dot(ys)
8     return amps
9
10    hs2 = dft(ys)
11    np.sum(np.abs(hs - hs2))

```

Итогом стала разница между двумя алгоритмами в 9.78820793827296e-16.

Далее, реализация алгоритма, разбивающего массив на половины, и применяющего к каждой из них FFT:

```

1    def fft_norec(ys):
2        N = len(ys)
3        He = np.fft.fft(ys[F:2])
4        Ho = np.fft.fft(ys[1F:2])
5        ns = np.arange(N)
6        W = np.exp(-1j * PI2 * ns / N)
7        return np.tile(He, 2) + W * np.tile(Ho, 2)
8
9    hs3 = fft_norec(ys)
10    np.sum(np.abs(hs - hs3))

```

В результате имеем ошибку, равную 3.142951601307097e-16, то есть в 3 раза меньше предыдущей.

Наконец, можно заменить библиотечную FFT на рекурсивные вызовы:

```

1    def fft(ys):
2        N = len(ys)
3        if N == 1:
4            return ys
5
6        He = fft(ys[::2])
7        Ho = fft(ys[1::2])
8
9        ns = np.arange(N)
10       W = np.exp(-1j * PI2 * ns / N)
11
12       return np.tile(He, 2) + W * np.tile(Ho, 2)
13
14    hs4 = fft(ys)
15    np.sum(np.abs(hs - hs4))

```

Ошибка - 5.363397650557411e-16, сравнима с предыдущей. Итоговый массив - $[-0.6+0.j \ -0.5-1.3j \ 0. \ +0.j \ -0.5+1.3j]$, что подтверждает правильность подтверждённых нами изысканий.