ДЗ 1

Витя Ефремов

27 января 2023 г.

Содержание

1 Теория

Если говорить о цифровых сигналах, то их отличие от аналоговых в дискретности. Дискретность по времени обычно называют дискретизацией. Дискретность по амплитуде - квантованием. А цифровым считают сигнал дискретный и по времени, и по амплитуде.

Дискретизация сигнала - это семплирование, представление непрерывного сигнала/функции набором точек/значений. Количество семплов в секунду - частота дискретизации. Исторически стандартными частотами дискретизации являются 44.1 и 48 кГц. Человек слышит звуки до примерно 20 кГц. Из теоремы Котельникова следует, что частота дискретизации должна быть вдвое больше. Откуда и получается частота чуть больше 40 кГц. В вики есть обсуждение других, менее значительных причин.

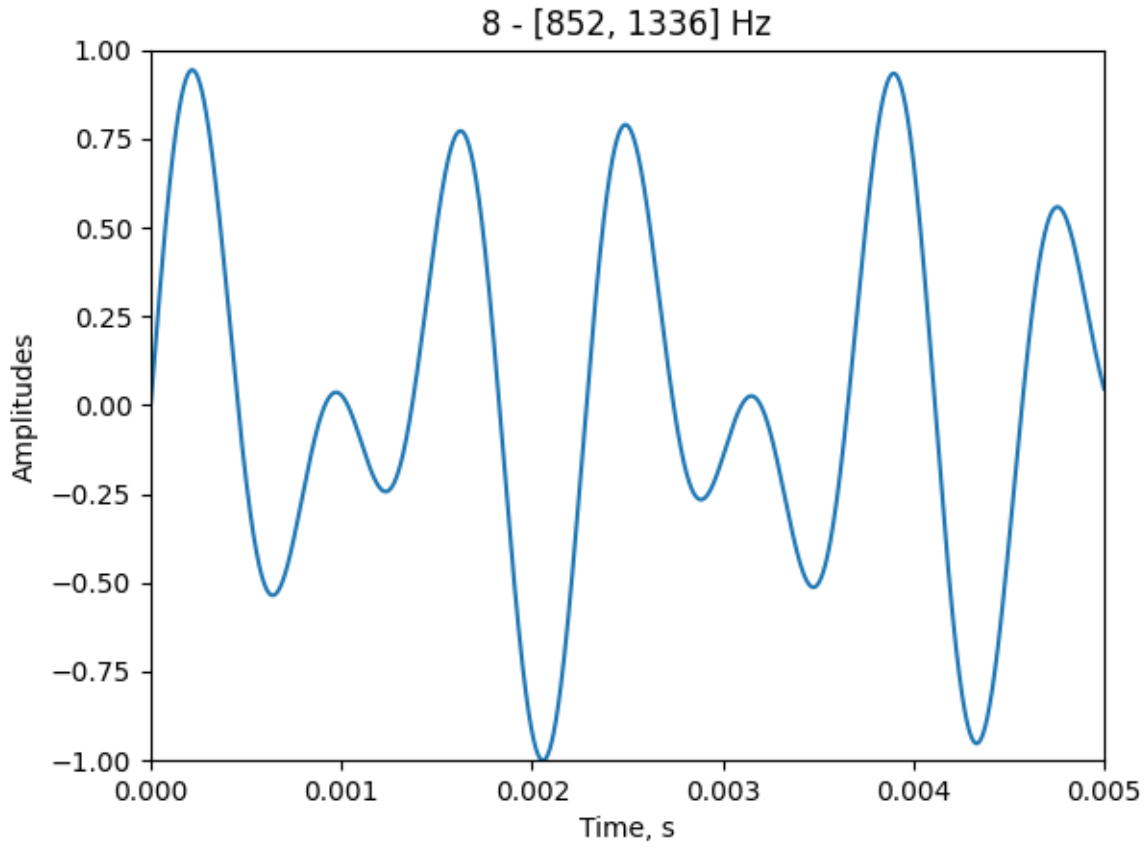
Тональный режим используется для передачи слежбной информации по телефонной сети. Например, вызываемый номер. Тональный режим позволяет кодировать 16 символов - `1234567890*#ABCD`. Каждый символ кодируется коротким звуковым сигналом состоящим из суммы двух синусоид.

	1209	1336	1477	1633
697	1	2	3	A
770	4	5	6	B
852	7	8	9	C
941	*	0	#	D

Таблица 1: Частоты синусоид

Если сменить частоту дискретизации с дефолтных 10 кГц на 2 кГц, то разница заметна на слух. Это происходит из-за того что частота дискретизации становится меньше удвоенной частоты высокого синуса. Если же менять частоту дискретизации, оставляя её достаточно высокой, то изменений нет. Т.е. что 10 кГц, что 8, что 15 звучат одинаково.

Так же график сигнала соответствующего восьмерке. Только по нему тяжело сделать какие-то выводы, но если сравнить с графиками других цифр, то видно что они действительно отличаются.



2 Приложение. Код

Код также доступен в репозитории на гитхабе. `https://github.com/Wicirelllis/miem-docs/blob/master/ИТСС/11-12 Цифровая обработка сигналов/hw_1/hw_1.py`

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


class hw_1:
    def __init__(self) -> None:
        self.freqs = {
            "1": [697, 1209],
```

```
            "2": [697, 1336],
            "3": [697, 1477],
            "4": [770, 1209],
            "5": [770, 1336],
            "6": [770, 1477],
            "7": [852, 1209],
            "8": [852, 1336],
            "9": [852, 1477],
            "0": [941, 1336],
            "a": [697, 1633],
            "b": [770, 1633],
            "c": [852, 1633],
            "d": [941, 1633],
            "*": [941, 1209],
            "#": [941, 1477],
        }
        self.p = pyaudio.PyAudio()
        self.fs = 10_000
        self.duration = 0.5
        self.stream = None
        self._start_stream()

    def _start_stream(self):
        if self.stream:
            self.stream.stop_stream()
            self.stream.close()
        self.stream = self.p.open(format=pyaudio.paFloat32, channels=1, rate=self.fs, output=True)

    def do_work(self) -> None:
        print("Welcome, type 'help' for help and 'exit' to exit.")
        while True:
            s = input().lower()
            if s == "exit":
                self.stream.stop_stream()
                self.stream.close()
                self.p.terminate()
                sys.exit(0)
            elif s == "help":
                print(
"\nType single key from '1234567890*#ABCDabcd' to play corresponding beep and show plot."
"\nType sequence of keys (e.g. *1801#) to play beeps in sequence. No plot is shown in this mode."
"\nAlso the following options are available:"
"\n    get"
"\n        Get the value of parameter. Available parameters are:"
"\n            fs - sampling frequency, Hz. Must be integer. Default is 10000Hz"
"\n            dur - duration of the beep, seconds. Default is 0.5s"
"\n        Example: get fs"
"\n    set"
"\n        Set the value of parameter. Same options as get."
"\n        Example: set fs 2000"
"\n    help"
"\n        Print this help page."
"\n    exit"
"\n        Close the program."
                )
                continue
            elif s.startswith("get"):
                if "fs" in s:
                    print(self.fs)
                elif "dur" in s:
                    print(self.duration)
                continue
            elif s.startswith("set"):
                if "fs" in s:
                    self.fs = int(s.split(" ")[-1])
                    self._start_stream()
```

```
                elif "dur" in s:
                    self.duration = float(s.split(" ")[-1])
                continue
            elif not set(s) <= set("1234567890*#abcd"):
                print("Unkonwn symbols in the input. Try again.")
                continue
            self.play_seq(s)

    def play_seq(self, s: str) -> None:
        if len(s) == 1:
            self.play_tone(s)
            self.draw_plot(s)
        else:
            for k in s:
                self.play_tone(k)

    def play_tone(self, k: str) -> None:
        freqs = self.freqs[k]
        samples = np.mean(
            np.sin(np.outer(freqs, 2 * np.pi * np.arange(self.fs * self.duration) / self.fs)),
            axis=0).astype(np.float32).tobytes()
        self.stream.write(samples)

    def draw_plot(self, k: str) -> None:
        freqs = self.freqs[k]
        x = np.linspace(0, 0.005, 100_000)
        y = np.mean(np.sin(np.outer(freqs, 2 * np.pi * x)), axis=0)

        fig, ax = plt.subplots()
        ax.set_xlim(0, 0.005)
        ax.set_ylim(-1, 1)
        ax.set_xlabel("Time, s")
        ax.set_ylabel("Amplitudes")
        ax.set_title(f"{k} - {freqs} Hz")
        ax.plot(x, y)
        fig.show()


hw = hw_1()
hw.do_work()
```