VRIJE UNIVERSITEIT BRUSSEL

NETWERKEN

IQ DEMODULATIE

Baris Sahan

1 FM and AM modulation/demodulation

1.1 Frequency modulation and demodulation

Voor het modulatie en demodulatie gebruiken we het volgende signalen en sample rate:

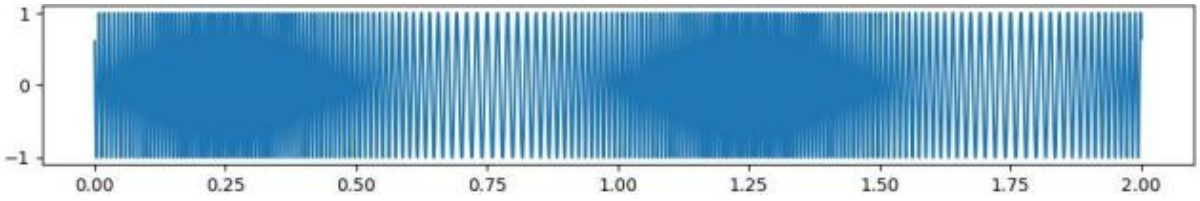
- Informatie signaal = 1Hz
- Carrier frequentie = 100Hz
- Sample rate is 10000

1.1.1 Modulation

Om ons signaal te moduleren kunnen we het gemoduleerd signaal als volgt schrijven:

```
ymodulated = plt.cos(2.0 * plt.pi * fc * ts + 2.0 * plt.pi * modfactor * intgysignal)
```

Modfactor bepaalt de zichtbaarheid van ons grafiek van het gemoduleerd signaal. Volgende grafiek is een overzicht van het gemoduleerd signaal met een modfactor 5:



1.1.2 Demodulation

Om het signaal te demoduleren moeten we eerst het signaal uitschrijven in I en Q-componenten. De I-component is het product van gemoduleerd signaal en cosinus van carrier frequentie, Q-component is het product van gemoduleerd signaal en sinus van carrier frequentie. I en Q-componenten schrijven we als volgt:

```
I = ymodulated*np.sin(2*np.pi*fc*ts)
```

```
Q = ymodulated*np.cos(2*np.pi*fc*ts)
```

Nadat I en Q-componenten van het signaal berekend zijn, kan je het signaal demoduleren. Door I en Q-componenten te voorstellen in het complex vlak met het complex getal:

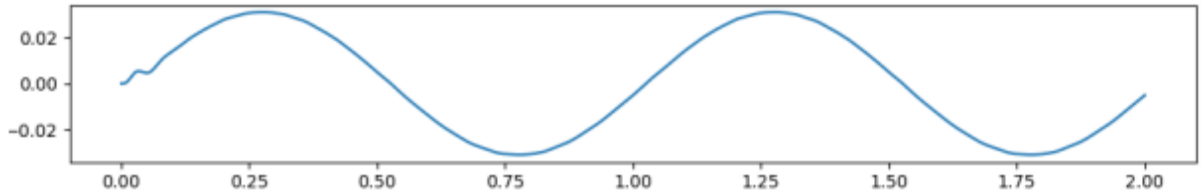
$$I+jQ$$

Kunnen we de amplitude bereken van dit complex getal:

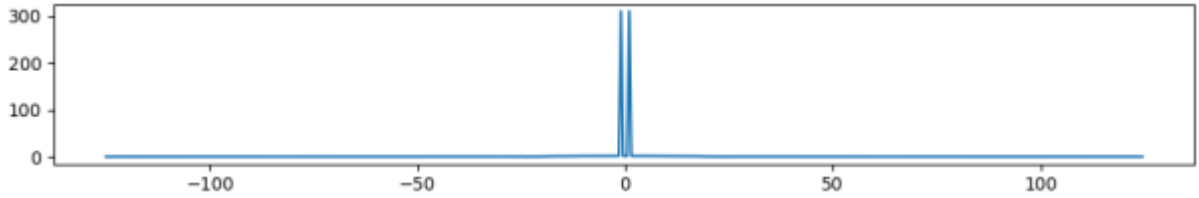
$$\sqrt{I^2 + Q^2}$$

Om verder het signaal helemaal gemoduleerd te krijgen moeten we dit doen voor elke sample. Nadat we het gemoduleerd signaal hebben gaan we het filteren gebruik makend van de scipy library in python. Scipy.signal.butter() is een stukje code die een butterworth filter genereerd. Een Butterworth filter is een soort lineair filter, ontworpen om zo constant een versterking mogelijk in zijn doorlaatband

Het gemoduleerd signaal wordt als volgt geplot:



Het frequentiedomein wordt als volgt geplot:



1.2 Amplitude modulation and demodulation

Voor het modulatie en demodulatie gebruiken we het volgende signalen:

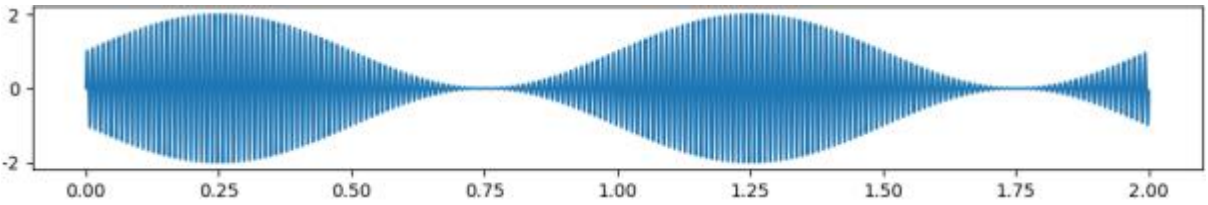
- Informatie signaal = 1Hz
- Carrier frequentie = 100Hz

1.2.1 Modulation

Om ons signaal te moduleren kunnen we het gemoduleerd signaal als volgt schrijven:

ymodulated = yc + yc * ysignal

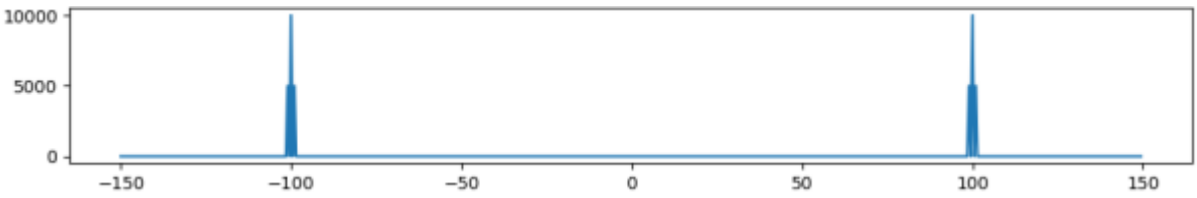
We plotten het gemoduleerd signaal in tijdsdomein:



We krijgen dus een signaal met als frequentie de carrierfrequentie, waarbij de amplitude is gemoduleerd door het signaal.

Daarnaast gaan we het frequentiedomein plotten. Om dit te doen gaan we de fast fourier transform gebruiken. Python laat ons toe om fft te gebruiken. Daarnaast wordt ook fftshift gebruikt. Deze verschuift de nulfrequentie component naar het midden van het spectrum.

Frequentiedomein is als volgt:



De pieken bevinden zich in de carrierfrequentie

1.2.2 Demodulatie

Om het signaal te demoduleren moeten we eerst het signaal uitschrijven in I en Q-componenten. De I-component is het product van gemoduleerd signaal en cosinus van carrier frequentie, Q-component is het product van gemoduleerd signaal en sinus van carrier frequentie. I en Q-componenten schrijven we als volgt:

```
I = ymodulated*np.sin(2*np.pi*fc*ts)

Q = ymodulated*np.cos(2*np.pi*fc*ts)
```

Nadat I en Q-componenten van het signaal berekend zijn, kan je het signaal demoduleren. Door I en Q-componenten te voorstellen in het complex vlak met het complex getal:

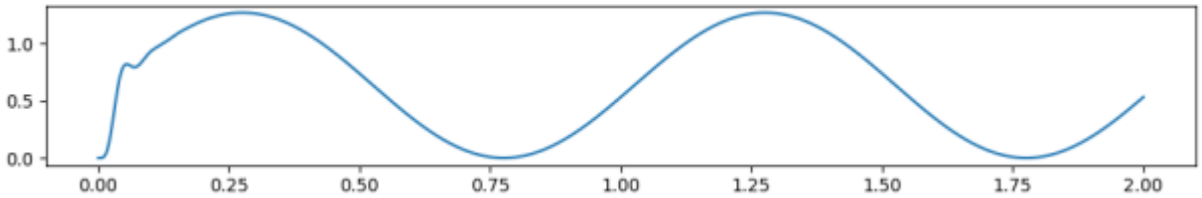
$$I+jQ$$

Kunnen we de amplitude bereken van dit complex getal:

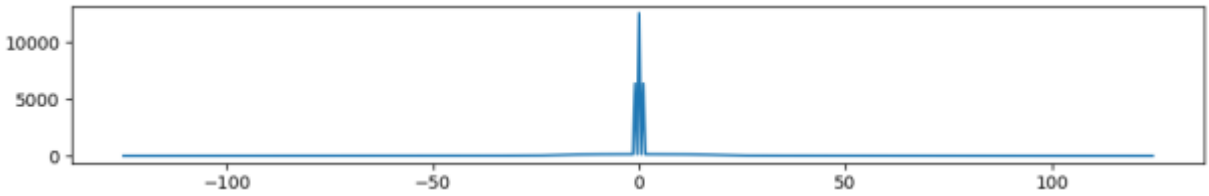
$$\sqrt{I^2 + Q^2}$$

Om verder het signaal helemaal gemoduleerd te krijgen moeten we dit doen voor elke sample. Nadien kunnen we het filteren, zoals beschreven in paragraaf 1.1.2

Het gemoduleerd signaal wordt als volgt geplot:



Het frequentiedomein wordt als volgt geplot:



We zien pieken bij 1Hz en -1Hz. De piek bij 0Hz komt door de piek die zichtbaar is in het begin van het tijdsdomein (die op zijn beurt een gevolg is van de signaalvertraging).

2 Decode FM radio broadcast with SDR-RTL module

2.1 Inleiding

Vooral we beginnen met het decoderen van FM-radio moeten we rekening houden met een aantal praktische zaken. Ten eerste gebruik makend van een power spectral density diagram van de uitgezonden radiosignalen, merken we dat signalen zich bevinden 200kHz van de referentiefrequentie. We gaan zich focussen op een van deze signalen om het signaal te decoderen. Dit wordt gedaan door signaal te vermenigvuldigen met : $e\,(-j2\pi * offset * t\,)$

Ten tweede door middel van power spectral density diagram gaan we volgende informatie gebruiken:

- Monosignaal = 15kHz
- Stereosignaal tussen 20kHz en 60kHz
- RDS-signaal = 57kHz
- Cutofffrequentie = 19kHz
- Sample frequentie van geluidkaart = 44.1 kHz

2.2 Demodulatie

We gaan eerst een offset van 200kHz zetten zoals beschreven in paragraaf 2.1. We gaan gebruik maken van een sample rate van 1102500. Dit is een geheel veelvoud van 220500. Deze waarde gaan we gebruiken voor de FM-brandbreedte. Dankzij decimate() functie kunnen we altijd overgaan tot een sample rate van 220500. Daarna gaan we het signaal verder demoduleren gebruik makend van het zelfde methode als in paragraaf 1.1.2 en we passen ook de butterworth lowpass filter toe. Ten laatste gaan we de functie decimate() gebruiken om naar de sample rate van de geluidkaart over te gaan. In dit geval 44100 sample rate.

2.3 Broadcasting

Voor het broadcasten ga ik gebruik maken van multithreading. We gaan de verwerkte signaal asynchroon in een buffer schrijven. Nadien gaat de geluidskaart gebruik makend van een ander thread asynchroon de buffer uitlezen. Dankzij deze manier kunnen we continu FM-signalen in onze geluidkaart laten afspelen. Al deze methoden zijn beschikbaar in asyncio library in python. Zoals sdr_streaming() die de verwerkte signaal in de buffer schrijft en callback() functie voor het uitlezen van de buffer.

