Hochschule Konstanz Technik, Wirtschaft und Gestaltung (HTWG)
Fakultät Informatik
Rechner- und Kommunikationsnetze
Prof. Dr. Dirk Staehle

Vorlesung Kommunikationstechnik

Digitale Modulation (Python)

Prof. Dr. Dirk Staehle

Bearbeitung in Zweier-Teams

Team-Mitglied 1:

Team-Mitglied 2:

In dieser Laborübung sollen die einzelnen Schritte der digitalen Modulation in Python nachvollzogen werden. Wir verwenden dazu teilweise das *komm* Modul (getestet mit Version 0.8.2). zusätzlich wird das Skript simDigMod.py bereitgestellt, das Sie anwenden und anpassen können.

1 Modulation und Demodulation

In dieser Teilaufgabe geht es darum, ein Verständnis für Modulation und Demodulation zu erhalten. Die Modulation besteht aus den Schritten digitale Modulation, Pulse Shaping und analoge Modulation. Die Demodulation aus den Schritten analoge Demodulation und Symbolrückgewinnung.

1.1 Die grundlegenden Parameter für diese Teilaufgabe sind

- die Samplingrate f_s: gibt an, mit welcher Rate das Signal dargestellt wird. Das inverse der Samplingrate ist der Abstand zwischen zwei „Signalpunkten“
- die Trägerfrequenz f_c
- die Anzahl Bits k
- das Modulationsschema mit m der Anzahl Bits pro Symbol
- die Datenrate r_b in Bits pro Sekunde
- Pulse-Shape (immer RectangularPulse)

Abgeleitete Parameter sind:

- die Anzahl Symbole $s = k/m$
- die Symbolrate $r_s = r_b/m$
- die Symboldauer $T_s = 1/r_s$
- die Anzahl Samples pro Symbol $n_s = T_s \cdot f_s$

Verwenden Sie zum Testen die folgenden Parameter:

- Modulationsschema: QPSK d.h. m=2
- Datenrate: 2 bps
- Trägerfrequenz: 4 Hz
- Anzahl Bits: 10
- Samplingrate: 64 Hz

1.2 Funktionen zur Modulation am Sender

- symbol_sequence=bit2symbol(bit_sequence, mod_scheme): Die Funktion transferiert eine Bit-Sequenz in eine Sequenz von komplexen Symbol entsprechend des gegebenen Modulationsschemas. Verwenden Sie dazu die entsprechenden Modulationen (PSKModulation bzw. QAMModulation) aus dem „komm“-Modul.
- bb_signal=pulse_shaping(symbol_sequence, pulse, nsamp): Die Funktion transferiert eine Symbolsequenz in ein Basisbandsignal, das mit nsamp Abtastpunkten pro Symbol dargestellt wird. Verwenden Sie die Klassen *RectangularPulse* und *TransmitFilter* aus dem „komm“-Modul

- fb_signal=amplitude_modulation(bb_signal, fc, fs): Die Funktion führt die Amplitudenmodulation durch. Implementieren Sie die Funktion nach der Vorlesung, indem Sie den Realteil des Basisbandsignals mit einem Cosinusträger und den Imaginärteil des Basisbandsignals mit einem Sinusträger multiplizieren. Das Verhältnis von Trägerfrequenz fc und Samplingrate fs definiert die Frequenz der Schwingung.

1.3 Funktionen zur Demodulation am Empfänger

- bb_signal=amplitude_demodulation(fb_signal, fc, fs): Die Funktion führt die Amplitudendemodulation durch. Die Funktion besteht aus einer erneuten Anwendung der Amplitudenmodulation auf die Trägerfrequenz gefolgt von einem Tiefpass mit der Trägerfrequenz las Grenzfrequenz. Achten Sie darauf, dass Sie ein komplexes Signal zurückerhalten. Den Realteil erhalten Sie durch die Amplitudenmodulation (Multiplikation) mit dem Cosinusträger und den Imaginärteil durch die Amplitudenmodulation (Multiplikation) mit dem Sinusträger. Implementieren Sie den Tiefpass, indem Sie im Frequenzbereich alle Signale oberhalb der Grenzfrequenz (hier: der Trägerfrequenz) abschneiden und das Signal dann zurück transferieren. Verwenden Sie die Funktionen *fft*, *ifft* und *fftfreq* aus dem Modul *numpy.fft.* Hilfreich ist auch die Funktion *nonzero* aus dem *numpy* Modul. Verdoppeln Sie die Stärke des Signals, um die Halbierung der Signalenergie durch die Demoudation zu kompensieren.
- symbol_sequence=integrate_and_dump(bb_signal, nsamp): Die Funktion transferiert ein komplex-wertiges Basisbandsignal in eine komplex-wertige Symbolsequenz, in dem die Abtastwerte für ein Symbol aufsummiert werden.
- bit_sequence=bit2symbol(symbol_sequence, mod_scheme): Die Funktion transferiert eine komplex-wertige Symbol-Sequenz entsprechend des gegebenen Modulationsschemas. Verwenden Sie dazu die entsprechenden Modulationen (PSKModulation bzw. QAMModulation) aus dem „komm“-Modul.

1.4 Grafische Darstellung und Auswertung

Stellen Sie die Bits, Symbole und Signale nach den einzelnen Schritten jeweils grafisch dar. Betrachten Sie dazu verschiedene Modulationsschemata.

1. Erstellen Sie ein Konstellationsdiagramm, das die Original-Symbole, die verrauschten Symbole und die wiedergewonnenen Symbole enthält.
2. Stellen Sie das Original-Basisbandsignal (bb_signal) und das verrauschte Basisbandsignal (noisy_bb_signal) im Zeitbereich dar. Erstellen Sie dazu eine Grafik mit zwei Subplots für die In-Phase- und die Quadrature-Komponente des Signals.
3. Vergleichen Sie das ursprüngliche und das verrauschte Signal im Freuqenzbereich (fb_signal, noisy_fb_signal) im Zeitbereich.

Bestimmen Sie außerdem die Bitfehlerwahrscheinlichkeit.

1.5 Simulationsstudie

Sie können die Simulation auf verschiedenen Detailebenen durchführen: auf Symbolebene, auf Basisbandebene und im Frequenzbereich (inkl. analoger Modulation). Dementsprechend wird das Signal auch auf der detailliertesten simulierten Ebene verrauscht, was Sie mit dem Schalter

`noise_level` einstellen können. Die Stärke des Rauschens geben Sie über das Signal-Rauschverhältnis SNRdB an.

Führen Sie zunächst eine Studie mit wenigen Symbole (<=20) im Frequenzbereich mit hohem SNR (50dB) durch. Verringern Sie dann den SNR und beachten Sie, wie sich das verrauschte Signal verändert. Dokumentieren Sie die Studie durch die erstellen Grafiken.

Führen dann eine Studie auf Symbolebene mit vielen Bits durch und betrachten Sie das Konstellationsdiagramm mit den verrauschen Punkten für unterschiedliche Modulationsschemata und SNR-Werte. Bestimmen Sie auch die Bitfehlerwahrscheinlichkeit.

Erstellen Sie eine Grafik mit dem SNR-Wert auf der X-Achse und der Bitfehlerwahrscheinlichkeit auf der Y-Achse. Die Grafik soll Kurven für BPSK, QPSK, 16QAM und 64QAM enthalten. Die Y-Achse soll logarithmisch skaliert sein. Bitfehlerwahrscheinlichkeit soll für alle Kurven von nahe 0.5 bis 0.0001 laufen. Wählen Sie den SNR-Bereich entsprechend.