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

Vorlesung Kommunikationstechnik

Laborübung zu

Digitale Modulation

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 Matlab nachvollzogen werden. Verwenden Sie dazu die Systemobjekte und Funktionen der Communication System Toolbox. Informationen finden Sie unter System Design/Digital Baseband Modulation.

1 Darstellung der Signale

Führen Sie die folgenden Schritte durch und vergleichen Sie grafisch die Bits, Basisbandsignale und modulierten Signale jeweils an Sender und Empfänger. Bei den digitalen Signalen sollten Sie keine Unterschiede feststellen.

1. Erzeugen Sie 20 Bits.
2. Generieren Sie ein 16QAM-Basisbandsignal mit einer Samplingrate f_s von 64 Samples pro Symbol.
3. Führen Sie die Amplitudenmodulation für eine Trägerfrequenz $f_c = f_s/4$ durch.
4. Stellen Sie das Basisbandsignal am Empfänger wieder her
5. Führen Sie die Detektion der Bits durch.
6. Bestimmen Sie die Anzahl fehlerhafter Bits.

Verwenden Sie dazu die folgenden Systemobjekte und Funktionen:

1. Funktion randsrc zur Erzeugung zufälliger Bit.
2. Systemobjekt RectangularQAMModulator zur Erzeugung der komplexen Symbole.
   - Das Systemobjekt wird (leider) in zukünftigen Matlab-Versionen nicht mehr unterstützt. Alternativ können Sie auch die Funktion qammod(x,M) mit den Einstellungen 'InputType'='bit' und 'UnitAveragePower'=1 (true) wählen.
3. Funktion rectpulse zur Erzeugung der Samples des analogen Basisbandsignals.
4. Funktion modulate mit Option „qam“ zur Durchführung der analogen Amplitudenmodulation.
5. Funktion demod mit Option „qam“ zur Durchführung der analogen Amplitudendemodulation.
6. Funktion intdump zur Rückgewinnung der komplexen Symbole aus dem Basisbandsigal.
7. Systemobjekt RectangularQAMDemodulator zur Detektion der Bits aus den komplexen Symbolen.
   - Das Systemobjekt wird (leider) in zukünftigen Matlab-Versionen nicht mehr unterstützt. Alternativ können Sie auch die Funktion qamdemod(x,M) mit den Einstellungen 'OupputType'='bit' und 'UnitAveragePower'=1 (true) wählen.
8. Funktion biterr zur Bestimmung der Anzahl fehlerhafter Bits.

2 Simulationsstudie mit verrauschten Signalen

Führen Sie nun eine Studie durch, um den Einfluss der Kanalqualität auf die Fehleranfälligkeit zu untersuchen. Die Kanalqualität wird durch das Signal-Rausch-Verhältnis (SNR, signal-to-noise-ratio) ausgedrückt und in Dezibel angegeben.

In ihrer Simulation verrauschen Sie ihr Signal über die Funktion awgn (additive white Gaussian noise). Um die Simulation einfach zu halten, führen Sie die Simulation im Basisband durch, d.h. Sie verrauschen direkt ihr Basisbandsignal und führen keine Amplitudenmodulation durch.

Erzeugen Sie das verrauschte durch awgn(x,SNRdB-pow2db(fs)), wobei x ihr Basisbandsignal ist. Sie können anstelle der awgn Funktion auch das AWGNChannel Objekt (Communication toolbox/Propagation and Channel Models) verwenden und hier für die Eigenschaft NoiseSource den Wert 'Signal to noise ratio (Es/No)' einstellen. Den SNR bzw. EsN0 Wert stellen Sie dann über die Eigenschaft 'EsNo' ein und bei 'SamplesPerSymbol' geben Sie die Samplingrate an.

Stellen Sie die originalen und verrauschten komplexen Symbole in einem Konstellationsdiagramm dar. Verwenden Sie dazu die Funktion scatter.

Vergleichen Sie für n=1000 Bits die Ergebnisse, die Sie für SNR-Werte von -10dB, 0dB und 10dB erhalten.