Simulation von Übertragungssystemen mit MATLAB und Scilab

Begleitmaterial zum Buch

Grundlagen der Nachrichtentechnik

Carsten Roppel (c.roppel@hs-sm.de)

Carl Hanser Verlag, 2. Aufl. 2023

Einführung

Wir beschreiben die Simulation verschiedener Übertragungsstrecken mit MATLAB und Scilab. Grundkenntnisse in der Anwendung dieser Werkzeuge werden vorausgesetzt. Für einen Einstieg in die Arbeit mit MATLAB bieten sich die Tutorien und Beispiele auf `https://de.mathworks.com` an, entsprechendes gilt für Scilab und `https://www.scilab.org`. In den vorliegenden Modellen wird die Umsetzung der folgenden Verfahren bzw. Funktionen in MATLAB und Scilab behandelt:

- Basisbandübertragung: Bipolares NRZ-Signal
- Modulation: QPSK in äquivalenter Basisband-Darstellung
- Wurzel-Kosinus-roll-off-Filter als Pulsformfilter und als signalangepasstes Filter
- Augendiagramm
- Leistungsdichtespektrum
- AWGN-Kanal
- Ermittlung der Bit- und Symbolfehlerhäufigkeit
- Signalraumplot
- Mehrwegekanal in äquivalenter Basisband-Darstellung
- T/2-Entzerrer

Der MATLAB-Code wurde mit der Version R2020a getestet. Teilweise werden Funktionen aus der Communications System Toolbox und der Signal Processing Toolbox verwendet. Stehen diese Toolboxen nicht zur Verfügung, können die Funktionen aber mithilfe von Standardfunktionen realisiert werden. Dazu kann der entsprechende Scilab-Code als Grundlage dienen.

Der Scilab-Code wurde mit der Version 5.5.2 getestet. Teilweise werden Funktionen aus der Communication Toolbox 0.3 verwendet. Diese muss über den Modulmanager ATOMS geladen werden.

Die MATLAB- und Scilab-Dateien stehen auf den Seiten
`https://plus.hanser-fachbuch.de/` und
`https://www.hs-schmalkalden.de/nachrichtentechnik` zur Verfügung.

Eine Python-Version ist dort ebenfalls verfügbar. Viele weitere MATLAB-Beispiele aus der Nachrichtentechnik findet man in [1] und [2].

1 Basisband-Übertragung und Empfänger mit signalangepasstem Filter

Wir bauen schrittweise die Simulation einer Basisband-Übertragungsstrecke auf. Es handelt sich um eine bipolare NRZ-Codierung. Als Pulsformfilter und dazu passendes signalangepasstes Filter dient ein Wurzel-Kosinus-roll-off-Filter. Nach dem Einfügen eines

AWGN-Kanals wird die Bitfehlerhäufigkeit bestimmt und mit dem theoretischen Ergebnis verglichen.

Versuch 1.1: Wurzel-Kosinus-roll-off-Filter

Wir entwerfen ein Wurzel-Kosinus-roll-off-Filter mit einer Impulsantwort gemäß Gl. (5.41) und einem Roll-off-Faktor $\alpha = 0{,}5$. Das Filter wird wie in Abschnitt 5.2.3 beschrieben als FIR-Filter realisiert. Die Impulsantwort wird auf den Bereich $\pm 3\,T_s$ beschränkt und hat 8 Abtastwerte pro Symbol.

Mithilfe von Gl. (4.5) wird die Energie der Impulsantwort bestimmt. Falls erforderlich, wird die Impulsantwort so skaliert, dass die Energie gleich 1 ist. Plots (ähnlich Bild 5.12 und Bild 5.27) veranschaulichen die Impulsantwort.

Bei Verwendung der MATLAB-Funktion `rcosdesign()` zur Erzeugung der Impulsantwort beträgt die Energie bereits 1 und es ist keine Skalierung erfoderlich. In Scilab wird die Impulsantwort entsprechend Gl. (5.41) definiert. Bei $t/T_s = 0$ und bei $t/T_s = 1/(4\alpha)$ wird der Nenner und auch der Zähler null. Für diese Werte muss der Funktionswert separat mithilfe der Regel von l'Hospital bestimmt werden.

MATLAB:

```
%% Parameter
alpha = 0.5;  % Roll-off-Faktor
os = 8;       % Samples pro Symbol (Oversampling Ratio)
dly_rc = 3;   % Verzögerung des Wurzel-Kosinus-roll-off-Filters

%% Wurzel-Kosinus-roll-off-Filter
psrc = rcosdesign(alpha, 2*dly_rc, os, 'sqrt');
E = sum(psrc.^2)  % Energie

figure;
stem(psrc);
xlabel('n');
ylabel('p_s_r_c(n)');

t = (-dly_rc*os:dly_rc*os)/os;
figure;
plot(t, psrc);
xlabel('t/T_s');
ylabel('p_s_r_c(t)');
```

Scilab:

```
//**********
// Parameter
//**********
alpha = 0.5;  // Roll-off-Faktor
os = 8;       // Samples pro Symbol (Oversampling Ratio)
dly_rc = 3;   // Verzögerung des Wurzel-Kosinus-roll-off-Filters
```

```
//*****************************
// Wurzel-Kosinus-roll-off-Filter
//*****************************
t = -dly_rc*os:dly_rc*os;

psrc = (4*alpha*(t/os).*cos(%pi*(1 + alpha)*(t/os)) + sin(%pi*(1 - alpha)*..
(t/os)))./((1 - (4*alpha*(t/os)).^2)*%pi.*(t/os));
// Grenzwert für t/T_s = 0
psrc(dly_rc*os + 1) = (4*alpha + %pi*(1 - alpha))/%pi;
// Grenzwert für t/T_s = 1/(4 alpha))
if round(os/(4*alpha)) == os/(4*alpha) then  // Ist os/(4*alpha) ganzzahlig?
    psrc(dly_rc*os + 1 + os/(4*alpha)) = (4*alpha*cos(%pi*(1 + alpha)/..
    (4*alpha)) - %pi*(1 + alpha)*sin(%pi*(1 + alpha)/(4*alpha)) + ..
    %pi*(1 - alpha)*cos(%pi*(1 - alpha)/(4*alpha)))/(-2*%pi);
    psrc(dly_rc*os + 1 - os/(4*alpha)) = psrc(dly_rc*os + 1 + os/(4*alpha));
end

E = sum(psrc.^2)  // Energie
psrc = psrc/sqrt(E);  // Skalierung auf E = 1
sum(psrc.^2)

plot2d3(psrc);
xlabel("n");
ylabel("p_src(n)");

figure;
plot2d(t/os, psrc);
xlabel("t/T_s");
ylabel("p_src(t)");
```

Versuch 1.2: Basisbandsignal

Wir erzeugen 1000 Zufallszahlen $\in \{0, 1\}$, die in binäre bipolare Symbole $\in \{-1, 1\}$ umgewandelt werden. Anschließend wird die Abtastrate auf 8 Abtastwerte pro Symbol erhöht, indem 7 Nullen zwischen den Symbolen eingefügt werden. Dieses Signal wird mit der Impulsantwort des Wurzel-Kosinus-roll-off-Filters gefaltet, und ein Plot der ersten 20 Symbole erstellt. Das Augendiagramm wird erzeugt und das Leistungsdichtespektrum in Form des Periodogramms wird berechnet und geplottet.

Es ist günstig, den Zufallsgenerator zu initialisieren, so dass immer die gleichen Zufallszahlen erzeugt werden und somit die Ergebnisse reproduzierbar bleiben. Da das Wurzel-Kosinus-roll-off-Filter kein Nyquist-Filter ist, schneiden sich im Augendiagramm die Linien an der Stelle der größten Augenöffnung *nicht* in einem Punkt.

Während in MATLAB für das Leistungsdichtespektrum die Funktion `periodogram()` verwendet werden kann, wird in Scilab das Spektrum gemäß Abschnitt 4.1.4, Gl. (4.27) berechnet. Da wir von einer Symboldauer $T_s = 1$ (Sekunde) und 8 Abtastwerten pro Symbol ausgehen, beträgt die Abtastrate $f_s = 8\,\text{Hz}$ und das Spektrum erstreckt sich von

$-f_s/2 \leq f \leq f_s/2$ oder $-4\,\text{Hz} \leq f \leq 4\,\text{Hz}$.

Ferner ist die Symbolrate $r_s = 1$ (Symbole pro Sekunde) und für $\alpha = 0{,}5$ beträgt die Übertragungsbandbreite nach Gl. (5.7) $B_K = 0{,}75\,\text{Hz}$. Das Leistungsdichtespektrum ist deutlich auf diese Bandbreite begrenzt. Die Signalanteile oberhalb von B_K gehen auf die numerische Berechnung des Spektrums mittels der diskreten Fourier-Transformation zurück.

MATLAB:

```
...
nSym = 1000;  % Anzahl Symbole
...
%% Basisbandsignal
rng(1234);  % Initialisiere Zufallsgenerator
bits = randi([0 1], nSym, 1);  % Spaltenvektor der Länge nSym mit
                               % Zufallszahlen 0 oder 1
symbols = 2*bits - 1;  % Binäre bipolare Symbolfolge
symbols = upsample(symbols, os);  % Einfügen von os - 1 Nullen zwischen
                                  % den Symbolen
x = conv(psrc, symbols);  % Faltung (convolution)
x = x(dly_rc*os + 1:end - dly_rc*os);  % Entfernen der ersten und letzten
                                       % Werte
figure;
plot(x(1:20*os));
xlabel('n');
ylabel('x(n)');

eyediagram(x, 2*os, 2);

figure;
periodogram(x, [], [], os, 'centered');
```

Scilab:

```
...
nSym = 1000;  // Anzahl Symbole
...
//****************
// Basisbandsignal
//****************
grand("setsd", 1234);  // Initialisiere Zufallsgenerator
bits = grand(nSym, 1, "uin", 0, 1);  // Spaltenvektor der Länge nSym mit
                                     // Zufallszahlen 0 oder 1
symbols = 2*bits - 1;  // Binäre bipolare Symbolfolge

symbols = upsample(symbols, os);  // Einfügen von os - 1 Nullen zwischen
                                  // den Symbolen
x = conv(psrc, symbols);  // Faltung (convolution)
x = x(dly_rc*os + 1:$ - dly_rc*os);  // Entfernen der ersten und letzten
                                     // Werte
```

```
figure;
plot(x(1:20*os));
xlabel("n");
ylabel("x(n)");

// Augendiagramm
figure;
eyemax = floor(length(x)/(2*os));
teye = ((-os:os)/os)';
for i = 1:eyemax - 1
    plot(teye, x(2*(i - 1)*os + 1:2*i*os + 1));
end
plot(teye(1:$-1), x(2*(eyemax - 1)*os + 1:2*eyemax*os));
xlabel("t/T_s");

// Leistungsdichtespektrum (Periodogramm)
len = length(x);
len = 2*floor(len/2);
x = x(1:len);
f = (os*(-len/2:len/2 - 1)/len)';
lds = 10*log10((1/(len*os))*fftshift(abs(fft(x)).^2));
figure;
plot(f, lds)
xlabel("f in Hz");
ylabel("LDS in dB");
```

Versuch 1.3: Signalangepasstes Filter

Da beim Wurzel-Kosinus-roll-off-Filter das signalangepasste Filter identisch zum Pulsformfilter ist, genügt es, das Basisbandsignal mit der Impulsantwort des Wurzel-Kosinus-roll-off-Filters zu falten. Das Signal wird vor und nach dem Empfangsfilter für die ersten 20 Symbole geplottet. Nach dem Empfangsfilter wird das Augendiagramm erstellt und das Leistungsdichtespektrum in Form des Periodogramms berechnet und geplottet.

Am Ausgang des Empfangsfilters entstehen Nyquist-Impulse, und im Augendiagramm schneiden sich die Linien an der Stelle der größten Augenöffnung alle in einem Punkt. Bei genauerem Hinsehen erkennt man allerdings, dass die Linien nicht exakt durch den gleichen Punkt verlaufen. Dies hängt damit zusammen, dass wir die Wurzel-Kosinus-roll-off-Filter nur näherungsweise implementiert haben, da die Impulsantwort der Filter zeitlich begrenzt ist. Die Näherung wird besser, wenn wir die Impulsantwort verlängeren (z. B. mit `dly_rc = 5`). Das Leistungsdichtespektrum hat die Form entsprechend Beispiel 5.1.

MATLAB:

```
...
%% Signalangepasstes Filter
ye = conv(psrc, x);
ye = ye(dly_rc*os + 1:end - dly_rc*os);
```

```
figure;
subplot(2, 1, 1);
plot(x(1:20*os));
xlabel('n');
ylabel('x(n)');
subplot(2, 1, 2);
plot(ye(1:20*os));
xlabel('n');
ylabel('y_e(n)');

eyediagram(ye, 2*os, 2);

figure;
periodogram(ye, [], [], os, 'centered');
```

Scilab:

```
...
//************************
// Signalangepasstes Filter
//************************
ye = conv(psrc, x);
ye = ye(dly_rc*os + 1:$ - dly_rc*os);

figure;
subplot(211);
plot(x(1:20*os));
xlabel("n");
ylabel("x(n)");
subplot(212);
plot(ye(1:20*os));
xlabel("n");
ylabel("y_e(n)");

// Augendiagramm
figure;
eyemax = floor(length(ye)/(2*os));
teye = ((-os:os)/os)';
for i = 1:eyemax - 1
    plot(teye, ye(2*(i - 1)*os + 1:2*i*os + 1));
end
plot(teye(1:$-1), ye(2*(eyemax - 1)*os + 1:2*eyemax*os));
xlabel("t/T_s");

// Leistungsdichtespektrum (Periodogramm)
len = length(ye);
len = 2*floor(len/2);
ye = ye(1:len);
f = (os*(-len/2:len/2 - 1)/len)';
lds = 10*log10((1/(len*os))*fftshift(abs(fft(ye)).^2));
```

```
figure;
plot(f, lds)
xlabel("f in Hz");
ylabel("LDS in dB");
```

Versuch 1.4: AWGN-Kanal

Wir fügen vor dem signalangepassten Filter einen AWGN-Kanal mit $E_b/N_0 = 6\,\mathrm{dB}$ ein. Das entsprechende Signal-Rausch-Verhältnis im Kanal wird gemäß Gl. (5.47) bestimmt. Wir müssen noch beachten, dass sich in dieser Gleichung die Rauschleistung auf die Bandbreite des Nutzsignals bezieht. Die Rauschleistung ist das Produkt Rauschleistungsdichte mal Bandbreite. Da wir in unserer Simulation die Bandbreite aufgrund der Überabtastung mit os = 8 Abtastwerten pro Symbol um den Faktor 8 erhöht haben, muss die Rauschleistung um den Faktor 8 größer und der Störabstand um $10 \lg 8\,\mathrm{dB}$ kleiner sein.

Wir bestimmen die Leistung $\overline{x^2(n)}$ des Basisbandsignals. Mithilfe des zuvor ermittelten Signal-Rausch-Verhältnises ergibt sich die erforderliche Rauschleistung N. Als Rauschsignal werden mittelwertfreie normal verteilte Zufallszahlen der Leistung N erzeugt. Diese werden zum Basisbandsignal addiert. Das Augendiagramm und das Leistungsdichtespektrum wird erzeugt. Aufgrund des Rauschens ist im Augendiagramm keine Öffnung mehr zu erkennen, und im Spektrum ragt das Nutzsignal nur noch wenig über das Rauschen hinaus.

Wir fügen wieder das signalangepasste Filter ein und erzeugen das Augendiagramm. Nach dem Empfangsfilter ist wieder eine Augenöffnung zu erkennen. Die Abtastrate wird auf 1 Abtastwert pro Symbol herabgesetzt. Dazu wird nur jeder achte Wert des Signals nach dem Empfangsfilter beibehalten. Dies müssen die Werten bei der größten Augenöffnung sein. Es folgt der Entscheider, der den Wert 0 oder 1 liefern soll, je nachdem ob der Abtastwert größer oder kleiner 0 ist.

Die Anzahl der Bitfehler sowie die Bitfehlerhäufigkeit wird durch Vergleich mit der zuvor erzeugten Bitfolge bestimmt. Zum Vergleich wird die theoretische Bitfehlerwahrscheinlichkeit für $E_b/N_0 = 6\,\mathrm{dB}$ mithilfe von Gl. (5.39) berechnet.

Der Entscheider wird mithilfe der Signum-Funktion `ysym = sign(ysym)` realisiert, es ist ysym $\in \{-1, 0, 1\}$. Durch die nachfolgende Anweisung `ybit = round((ysym + 1)/2)` wird 1 auf 1, -1 auf 0 und 0 auf 1 abgebildet. In MATLAB werden die Anzahl der Bitfehler und die Bitfehlerhäufigkeit mit der Funktion `biterr()` ermittelt. In Scilab werden die unterschiedlichen Bits mit der Exklusiv-Oder-Funktion `bitxor()` bestimmt. Es erfolgt jeweils noch eine Ausgabe der ermittelten Werte mithilfe einer print-Anweisung.

MATLAB:

```
...
EbN0 = 6;     % Eb/N0 in dB
...
%% AWGN-Kanal
snr = 10*log10(2) + EbN0 - 10*log10(os);  % Störabstand in dB
sigPwr = mean(x.^2);  % Signalleistung
noisePwr = sigPwr/(10^(snr/10));  % Rauschleistung
```

```
rng(1234); % Initialisiere Zufallsgenerator
noise = randn(length(x), 1)*sqrt(noisePwr);
y = x + noise;  % Signal + Rauschen

eyediagram(y, 2*os, 2);

figure;
periodogram(y, [], [], os, 'centered');

%% Empfänger mit signalangepasstem Filter
ye = conv(psrc, y);
ye = ye(dly_rc*os + 1:end - dly_rc*os);

eyediagram(ye, 2*os, 2);

ysym = downsample(ye, os);  % Herabsetzen der Abtastrate auf 1 Sample pro
                            % Symbol
ysym = sign(ysym);  % Entscheider
ybit = round((ysym + 1)/2);

% Bestimmung der Bitfehlerhäufigkeit
[nerror, ber] = biterr(ybit, bits);
berTheo = 0.5*erfc(sqrt(10^(EbN0/10)));
fprintf('Eb/N0: %f dB  Pb: %f  Bitfehler: %d  Bitfehlerhäufigkeit: %f\n', ...
EbN0, berTheo, nerror, ber);
```

Scilab:

```
...
EbN0 = 6;     // Eb/N0 in dB
...
//***********
// AWGN-Kanal
//***********
snr = 10*log10(2) + EbN0 - 10*log10(os);  // Störabstand in dB
sigPwr = mean(x.^2);  // Signalleistung
noisePwr = sigPwr/(10^(snr/10));  // Rauschleistung
rand("seed", 1234);
noise = rand(length(x), 1, "normal")*sqrt(noisePwr);
y = x + noise;

// Augendiagramm
figure;
eyemax = floor(length(y)/(2*os));
teye = ((-os:os)/os)';
for i = 1:eyemax - 1
    plot(teye, y(2*(i - 1)*os + 1:2*i*os + 1));
end
plot(teye(1:$-1), y(2*(eyemax - 1)*os + 1:2*eyemax*os));
xlabel("t/T_s");
```

```
// Leistungsdichtespektrum (Periodogramm)
len = length(y);
len = 2*floor(len/2);
y = y(1:len);
f = (os*(-len/2:len/2 - 1)/len)';
lds = 10*log10((1/(len*os))*fftshift(abs(fft(y)).^2));
figure;
plot(f, lds)
xlabel("f in Hz");
ylabel("LDS in dB");

//*************************************
// Empfänger mit signalangepasstem Filter
//*************************************
ye = conv(psrc, y);
ye = ye(dly_rc*os + 1:$ - dly_rc*os);

// Augendiagramm
figure;
eyemax = floor(length(ye)/(2*os));
teye = ((-os:os)/os)';
for i = 1:eyemax - 1
    plot(teye, ye(2*(i - 1)*os + 1:2*i*os + 1));
end
plot(teye(1:$-1), ye(2*(eyemax - 1)*os + 1:2*eyemax*os));
xlabel("t/T_s");

ysym = downsample(ye, os);  // Herabsetzen der Abtastrate auf 1 Sample
                            // pro Symbol
ysym = sign(ysym);  // Entscheider
ybit = round((ysym + 1)/2);

// Bestimmung der Bitfehlerhäufigkeit
nerror = sum(bitxor(ybit, bits));
ber = nerror/length(ybit);
berTheo = 0.5*erfc(sqrt(10^(EbN0/10)));
mprintf('Eb/N0: %f dB  Pb: %f  Bitfehler: %d  Bitfehlerhäufigkeit: %f\n', ..
EbN0, berTheo, nerror, ber);
```

Versuch 1.5: Bitfehlerhäufigkeit

Wir wollen abschließend die Bitfehlerhäufigkeit für $E_b/N_0 = 4,\ 5,\ \ldots,\ 8\,\mathrm{dB}$ bestimmen. Dazu legen wir einen entsprechenden Vektor für E_b/N_0 an. Die Anzahl der Symbole beträgt 20000. Innerhalb einer for-Schleife werden die Anweisungen für den AWGN-Kanal, den Empfänger mit signalangepasstem Filter (ohne das Augendiagramm) und die Bestimmung der Bitfehlerhäufigkeit ausgeführt. Die Schleife wird für jeden der E_b/N_0-Werte einmal durchlaufen. Die Werte für die Bitfehlerhäufigkeit und die theoretische Bitfehler-

wahrscheinlichkeit werden jeweils in einem Vektor abgespeichert. Nachdem die for-Schleife durchlaufen wurde, wird die ermittelte Bitfehlerhäufigkeit und die theoretische Bitfehlerwahrscheinlichkeit in einem halblogarithmischen Plot über E_b/N_0 in dB geplottet.

MATLAB:

```
...
nSym = 2e4;   % Anzahl Symbole
EbN0List = [4:8];  % Eb/N0 in dB
berTheoList = zeros(1, length(EbN0List));
berList = zeros(1, length(EbN0List));
...
%% Schleife für verschiedene Eb/N0
for i = 1:length(EbN0List)
    % AWGN-Kanal
    EbN0 = EbN0List(i);
    ...
    berList(i) = ber;
    berTheo = 0.5*erfc(sqrt(10^(EbN0/10)));
    berTheoList(i) = berTheo;
    fprintf('Eb/N0: %5.2f dB  Pb: %f  Bitfehler: %4d  Bitfehlerhäufigkeit: ...
    %f\n', EbN0, berTheo, nerror, ber);
end

%% Plot BER vs. Eb/N0
figure;
semilogy(EbN0List, berList, '-.or');
grid;
xlabel('E_b/N_0 in dB'),
ylabel('Bitfehlerhäufigkeit');
hold on;
plot(EbN0List, berTheoList, '-x');
hold off;
legend('Simulation', 'Theoretisch', 'Location', 'SouthWest');
```

Scilab:

```
...
nSym = 2e4;  // Anzahl Symbole
EbN0List = [4:8];  // Eb/N0 in dB
berTheoList = zeros(1, length(EbN0List));
berList = zeros(1, length(EbN0List));
...
//*******************************
// Schleife für verschiedene Eb/N0
//*******************************
for i = 1:length(EbN0List)
    // AWGN-Kanal
    EbN0 = EbN0List(i);
    ...
```

```
    berList(i) = ber;
    berTheo = 0.5*erfc(sqrt(10^(EbN0/10)));
    berTheoList(i) = berTheo;
    mprintf('Eb/N0: %5.2f dB  Pb: %f  Bitfehler: %4d  Bitfehlerhäufigkeit: ..
    %f\n', EbN0, berTheo, nerror, ber);
end

//*******************
// Plot BER vs. Eb/N0
//*******************
figure;
plot2d("nl", EbN0List, berList, 5);
plot2d("nl", EbN0List, berList, -2);
xlabel("Eb/N0 in dB");
ylabel("Bitfehlerhäufigkeit");
plot2d("nl", EbN0List, berTheoList, 2);
```

2 QPSK-Übertragungsstrecke und Empfänger mit signalangepasstem Filter

Wir verwenden QPSK (Quadrature Phase-Shift Keying) als Modulationsverfahren. Zur Pulsformung dient wieder ein Wurzel-Kosinus-roll-off-Filter, und im Empfänger kommt das passende signalangepasste Filter zum Einsatz. Die Signale werden mit Hilfe des Augendiagramms, des Konstellationsdiagramms und des Leistungsdichtespektrums untersucht. Die Symbolfehlerhäufigkeit wird ermittelt und mit dem theoretischen Ergebniss verglichen.

Das System wird im Basisband simuliert, d. h. es werden die zum Bandpasssignal äquivalenten Tiefpasssignale $x_i(t)$ und $x_q(t)$ erzeugt und verarbeitet. Dadurch werden die bei einer Simulation im Bandpassbereich auftretenden hohen Frequenzen und Abtastraten vermieden. Andererseits können alle Einflüsse auf das Bandpasssignal (Kanalübertragungsfunktion, Störungen) durch äquivalente Tiefpassmodelle in die Simulation einbezogen werden.

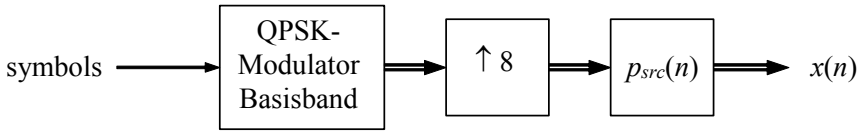
Versuch 2.1: QPSK-Modulator und Demodulator

Wir erzeugen als Symbolfolge 1000 Zufallszahlen $\in \{0, \ldots, m-1\}$ mit $m = 4$. Für $\lambda = \pi/4$ werden den Symbolen gemäß Gl. (6.55) bei der QPSK-Modulation folgende Phasenwinkel zugeordnet:

a_k	φ_k
0	$\pi/4$
1	$3\pi/4$
2	$5\pi/4$
3	$7\pi/4$

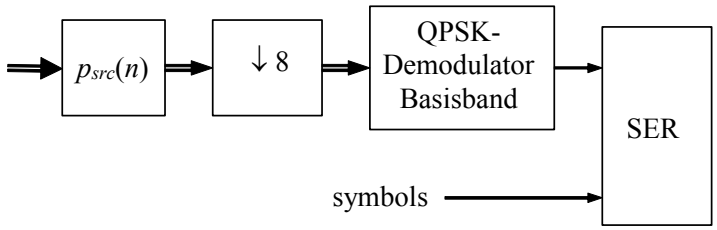
Ein Basisbandmodell eines QPSK-Modulators erzeugt die komplexen Werte $I_k + j\,Q_k = \cos\varphi_k + j\,\sin\varphi_k = \exp(j\,\varphi_k)$. Diese entsprechen den vier Punkten im Signalraum. In

MATLAB steht die Funktion `pskmod()` zur Verfügung; in Scilab wird der Modulator mithilfe obiger Gleichung realisiert. Anschließend wird wie in Versuch 1.2 die Abtastrate auf 8 Abtastwerte pro Symbol erhöht und das Signal mit der Impulsantwort des Wurzel-Kosinus-roll-off-Filters gefaltet. Es ergibt sich folgendes Simulationsmodell zur Erzeugung des QPSK-Signals:



Nun kann das Augendiagramm und das Leistungsdichtespektrum erzeugt werden. Da das Basisbandsignal jetzt ein komplexes Signal ist, werden zwei Augendiagramme jeweils für den Realteil (die I-Komponente) und den Imaginärteil (die Q-Komponente) erstellt.

Im Empfänger wird das Signal zuerst mit der Impulsantwort des signalangepassten Filters gefaltet und das Augendiagramm erzeugt. Anschließend wird die Abtastrate auf 1 Abtastwert pro Symbol herabgesetzt (die Werte bei der größten Augenöffnung werden beibehalten) und ein Signalraumplot (Scatterplot) erzeugt. Es folgt noch der Basisband-Demodulator, und wir erhalten folgendes Simulationsmodell zur Demodulation des QPSK-Signals:



Im Signalraumplot sind die vier Punkte der QPSK-Signalraumkonstellation zu erkennen wie in Bild 6.33 mitte, mit einer zusätzlichen Drehung um $\lambda = \pi/4$. Die Punkte liegen nicht exakt übereinander, da wir die Wurzel-Kosinus-roll-off-Filter nur näherungsweise implementiert haben (vgl. die Bemerkung zu Versuch 1.3).

In MATLAB kann der Demodulator mit der Funktion `pskdemod()` realisiert werden. In Scilab wird zunächst die Drehung um λ durch Multiplikation mit $\exp(-j\lambda)$ rückgängig gemacht. Anschließend wird die Phase mithilfe der arctan-Funktion bestimmt, diese liegt im Bereich $-\pi$ bis π. Durch Multiplikation mit $m/(2\,\pi)$ und Runden auf ganze Zahlen erhält man Werte $\in \{-2, -1, 0, 1, 2\}$. Zu den negativen Werten wird $m = 4$ addiert und man erhält die demodulierten Symbole $\in \{0, 1, 2, 3\}$. Dieses Vorgehen entspricht Entscheidungsgrenzen, die gleich den Winkelhalbierenden zwischen den Punkten im Signalraum sind. Es kann auch für $m > 4$ angewendet werden. Für $m = 8$ erhält man beispielsweise 8-PSK mit einer Signalraumkonstellation und Entscheidungsgrenzen entsprechend Bild 6.55 a.

Abschließend vergleichen wir die sendeseitig erzeugte Symbolfolge mit der demodulierten Symbolfolge. Hier kann es zu einer Verschiebung der Folgen relativ zueinander kommen, die für die Ermittlung der Symbolfehler kompensiert werden muss.

MATLAB:

```
%% Parameter
alpha = 0.5;     % Roll-off-Faktor
os = 8;          % Samples pro Symbol (Oversampling Ratio)
```

```
dly_rc = 3;     % Verzögerung des Wurzel-Kosinus-roll-off-Filters
nSym = 1000;    % Anzahl Symbole
m = 4;          % m-wertige Symbole
k = log2(m);    % k Bit pro Symbol
lambda = pi/4;  % QPSK Phasenoffset
...
%% QPSK-Modulator und Pulsformfilter
rng(1234);  % Initialisiere Zufallsgenerator
symbols = randi([0 (m - 1)], nSym, 1);  % Spaltenvektor der Länge nSym
                                % mit Zufallszahlen im Bereich 0 bis m-1

xmod = pskmod(symbols, m, lambda);  % Modulation

xmod = upsample(xmod, os);  % Einfügen von os - 1 Nullen zwischen den
                            % Symbolen
x = conv(psrc, xmod);  % Faltung (convolution)
x = x(dly_rc*os + 1:end - dly_rc*os);  % Entfernen der ersten und letzten
                                       % Werte
eyediagram(x, 2*os, 2);

figure;
periodogram(x, [], [], os, 'centered');

%% Signalangepasstes Filter und QPSK-Demodulator
ye = conv(psrc, x);
ye = ye(dly_rc*os + 1:end - dly_rc*os);

eyediagram(ye, 2*os, 2);

ye = downsample(ye, os);

scatterplot(ye);

ysym = pskdemod(ye, m, lambda);  % Demodulation

[symbols(1:10) ysym(1:10)]  % Vergleich
```

Scilab:

```
//**********
// Parameter
//**********
alpha = 0.5;     // Roll-off-Faktor
os = 8;          // Samples pro Symbol (Oversampling Ratio)
dly_rc = 3;      // Verzögerung des Wurzel-Kosinus-roll-off-Filters
nSym = 1000;     // Anzahl Symbole
m = 4;           // m-wertige Symbole
k = log2(m);     // k Bit pro Symbol
lambda = %pi/4;  // QPSK Phasenoffset
...
```

```
//*********************************
// QPSK-Modulator und Pulsformfilter
//*********************************
grand("setsd", 1234);  // Initialisiere Zufallsgenerator
symbols = grand(nSym, 1, "uin", 0, m - 1);  // Spaltenvektor der Länge
                        // nSym mit Zufallszahlen im Bereich 0 bis m-1

// Modulation
xmod = exp(%i*((2*%pi/m).*symbols + lambda));

xmod = upsample(xmod, os);  // Einfügen von os - 1 Nullen zwischen den
                            // Symbolen
x = conv(psrc, xmod);  // Faltung (convolution)
x = x(dly_rc*os + 1:$ - dly_rc*os);  // Entfernen der ersten und letzten
                                     // Werte
// Augendiagramm
xi = real(x);
xq = imag(x);
eyemax = floor(length(xi)/(2*os));
teye = ((-os:os)/os)';
figure;
subplot(211);
for i = 1:eyemax - 1
    plot(teye, xi(2*(i - 1)*os + 1:2*i*os + 1));
end
plot(teye(1:$-1), xi(2*(eyemax - 1)*os + 1:2*eyemax*os));
xlabel("t/T_s");
ylabel("x_i");
subplot(212);
for i = 1:eyemax - 1
    plot(teye, xq(2*(i - 1)*os + 1:2*i*os + 1));
end
plot(teye(1:$-1), xq(2*(eyemax - 1)*os + 1:2*eyemax*os));
xlabel("t/T_s");
ylabel("x_q");

// Leistungsdichtespektrum (Periodogramm)
len = length(x);
len = 2*floor(len/2);
x = x(1:len);
f = (os*(-len/2:len/2 - 1)/len)';
lds = 10*log10((1/(len*os))*fftshift(abs(fft(x)).^2));
figure;
plot(f, lds)
xlabel("f in Hz");
ylabel("LDS in dB");

//*********************************************
// Signalangepasstes Filter und QPSK-Demodulator
//*********************************************
```

```
ye = conv(psrc, x);
ye = ye(dly_rc*os + 1:$ - dly_rc*os);

// Augendiagramm
yi = real(ye);
yq = imag(ye);
eyemax = floor(length(yi)/(2*os));
teye = ((-os:os)/os)';
figure;
subplot(211);
for i = 1:eyemax - 1
    plot(teye, yi(2*(i - 1)*os + 1:2*i*os + 1));
end
plot(teye(1:$-1), yi(2*(eyemax - 1)*os + 1:2*eyemax*os));
xlabel("t/T_s");
ylabel("y_i");
subplot(212);
for i = 1:eyemax - 1
    plot(teye, yq(2*(i - 1)*os + 1:2*i*os + 1));
end
plot(teye(1:$-1), yq(2*(eyemax - 1)*os + 1:2*eyemax*os));
xlabel("t/T_s");
ylabel("y_q");

ye = downsample(ye, os);

// Scatterplot
figure;
plot(real(ye), imag(ye), 'ob')
xlabel("y_i");
ylabel("y_q");

// Demodulation
ye = ye.*exp(-%i*lambda);
ysym = round(m/(2*%pi)*atan(imag(ye), real(ye)));
ysym(ysym < 0) = ysym(ysym < 0) + m;

[symbols(1:10) ysym(1:10)]  // Vergleich
```

Versuch 2.2: AWGN-Kanal für komplexes Basisbandsignal

Analog zu Versuch 1.4 ergänzen wir unser Simulationsmodell um einen AWGN-Kanal mit $E_b/N_0 = 6\,\text{dB}$. Für den Zusammenhang zwischen S/N und E_b/N_0 gilt nun Gl. (6.99). Zur Bestimmung der Signalleistung bilden wir den Mittelwert über das Betragsquadrat der komplexen Werte. Da jede der Quadraturkomponenten die gleiche Leistung wie das Bandpasssignal selbst hat, müssen wir noch mit 1/2 multiplizieren. Wie in Versuch 1.4 erzeugen wir normal verteilte Zufallszahlen mit der erforderlichen Rauschleistung. Dies

muss hier getrennt für die I- und die Q-Komponente mit unabhängig voneinander erzeugten Zufallszahlen erfolgen.

Nach dem signalangepassten Filter und der Demodulation bestimmen wir die Symbolfehler und die Symbolfehlerhäufigkeit, und es folgt der Vergleich mit dem theoretischen Wert $P_{s,\,\mathrm{QPSK}} \approx 2\,P_{b,\,\mathrm{QPSK}}$ mit der Bitfehlerwahrscheinlichkeit $P_{b,\,\mathrm{QPSK}}$ nach Gl. (6.89).

MATLAB:

```
...
EbN0 = 6;       % Eb/N0 in dB
...
%% AWGN-Kanal
snr = EbN0 + 10*log10(k) - 10*log10(os);  % Störabstand in dB
sigPwr = 0.5*mean(abs(x).^2);  % Signalleistung
noisePwr = sigPwr/(10^(snr/10));  % Rauschleistung
rng(1234);
noise_i = randn(length(x), 1)*sqrt(noisePwr);
rng(4321);
noise_q = randn(length(x), 1)*sqrt(noisePwr);
y = x + noise_i + j*noise_q;  % Signal + Rauschen
...
% Bestimmung der Symbolfehlerhäufigkeit
[nerror, ser] = symerr(symbols, ysym);
if m <= 4
    serTheo = erfc(sqrt(10^(EbN0/10)));
else
    serTheo = erfc(sqrt(log2(m)*10^(EbN0/10))*sin(pi/m));
end
fprintf('Eb/N0: %f dB  Ps: %f  Symbolfehler: %d  Symbolfehlerhäufigkeit: ...
%f\n', EbN0, serTheo, nerror, ser);
```

Scilab:

```
...
EbN0 = 6;        // Eb/N0 in dB
...
//***********
// AWGN-Kanal
//***********
snr = EbN0 + 10*log10(k) - 10*log10(os);  // Störabstand in dB
sigPwr = 0.5*mean(abs(x).^2);  // Signalleistung
noisePwr = sigPwr/(10^(snr/10));  // Rauschleistung
rand("seed", 1234);
noise_i = rand(length(x), 1, "normal")*sqrt(noisePwr);
rand("seed", 4321);
noise_q = rand(length(x), 1, "normal")*sqrt(noisePwr);
y = x + noise_i + %i*noise_q;  // Signal + Rauschen
...
// Bestimmung der Symbolfehlerhäufigkeit
nerror = sum(ceil(bitxor(symbols, ysym)/m));
```

```
ser = nerror/length(ysym);
serTheo = erfc(sqrt(10^(EbN0/10)));
mprintf('Eb/N0: %f dB  Ps: %f  Symbolfehler: %d  Symbolfehlerhäufigkeit: ..
%f\n', EbN0, serTheo, nerror, ser);
```

Versuch 2.3: Symbolfehlerhäufigkeit

Abschließend wird wieder die Symbolfehlerhäufigkeit für $E_b/N_0 = 4 \ldots 8\,\mathrm{dB}$ bestimmt und die ermitttelten Werte mit den theoretischen Werten in einem halblogarithmischen Plot verglichen (vgl. Versuch 1.5).

MATLAB:

```
...
EbN0List = [4:8];  % Eb/N0 in dB
serTheoList = zeros(1, length(EbN0List));
serList = zeros(1, length(EbN0List));
...
%% Schleife für verschiedene Eb/N0
for i = 1:length(EbN0List)
    % AWGN-Kanal
    EbN0 = EbN0List(i);
    ...
    serList(i) = ser;
    if m <= 4
        serTheo = erfc(sqrt(10^(EbN0/10)));
    else
        serTheo = erfc(sqrt(log2(m)*10^(EbN0/10))*sin(pi/m));
    end
    serTheoList(i) = serTheo;
    fprintf('Eb/N0: %5.2f dB  Ps: %f  Symbolfehler: %4d  ...
    Symbolfehlerhäufigkeit: %f\n', EbN0, serTheo, nerror, ser);
end

%% Plot SER vs. Eb/N0
...
```

Scilab:

```
...
EbN0List = [4:8];  // Eb/N0 in dB
serTheoList = zeros(1, length(EbN0List));
serList = zeros(1, length(EbN0List));
...
//*******************************
// Schleife für verschiedene Eb/N0
//*******************************
for i = 1:length(EbN0List)
```

```
    // AWGN-Kanal
    EbN0 = EbN0List(i);
    ...
    serList(i) = ser;
    serTheo = erfc(sqrt(10^(EbN0/10)));
    serTheoList(i) = serTheo;
    mprintf('Eb/N0: %5.2f dB  Ps: %f  Symbolfehler: %4d  ..
    Symboltfehlerhäufigkeit: %f\n', EbN0, serTheo, nerror, ser);
end

//*******************
// Plot SER vs. Eb/N0
//*******************
...
```

3 QPSK-Übertragungsstrecke mit Mehrwegekanal und Empfänger mit T/2-Entzerrer

Wir ergänzen die QPSK-Übertragungsstrecke um einen Mehrwegekanal, der als äquivalentes Basisbandmodell simuliert wird. Die durch den Kanal hervorgerufenen Verzerrungen führen zu Intersymbolinterferenz (ISI). Zur Kompensation wird ein T/2-Entzerrer verwendet.

Versuch 3.1: Mehrwegekanal

Der Mehrwegekanal sowie dessen Impulsantwort und Übertragungsfunktion wird in Beispiel 2.9 beschrieben. Zu der Impulsantwort $h(t) = \delta(t) + a\,\delta(t - t_0)$ gehört die Impulsantwort des äquivalenten Tiefpasssystems

$$h_{\mathrm{TP}}(t) = \delta(t) + a\,\delta(t - t_0)\ \exp(-j2\pi f_c t_0)$$

wie man sich durch Einsetzen in Gl. (6.3) überzeugen kann. Die Übertragungsfunktion lautet:

$$H_{\mathrm{TP}}(f) = 1 + a\ \exp(-j2\pi f_c t_0)\ \exp(-j2\pi f t_0)$$

Der Betrag der Übertragungsfunktion wird geplottet und das Basisbandsignal wird mit der zeitdiskreten Version der Impulsantwort gefaltet. Das Leistungsdichtespektrum des Signals nach dem Mehrwegekanal wird erzeugt, ebenso wie das Augendiagramm nach dem signalangepassten Filter.

Ist $b = f_c\,t_0$ ganzzahlig, so liegt die Trägerfrequenz bei einem Maximum von $|H(f)|$ (siehe Bild 2.20). Für $b = f_c\,t_0 = 1/2$ liegt die Trägerfrequenz dagegen bei einem Minimum. Die Übertragungsfunktion des äquivalenten Tiefpasssystems hat entspechend ein Maximum bzw. ein Minimum bei $f = 0$. In beiden Fällen ist $h_{\mathrm{TP}}(t)$ reell und $|H_{\mathrm{TP}}(f)|$ ist eine gerade Funktion. Für andere Werte von b ist die Impulsantwort komplex und $|H_{\mathrm{TP}}(f)|$ ist nicht mehr symmetrisch bezüglich $f = 0$.

Für $b = 1$ und eine Laufzeitdifferenz t_0 entsprechend zwei Abtastperioden liegt der erste Einbruch der Übertragungsfunktion bei 2 Hz und damit außerhalb des Signals mit der Bandbreite $B_K = 0{,}75\,\text{Hz}$. Die Intersymbolinterferenz und damit die Auswirkung auf die vertikale Augenöffnung sind nur gering. Für eine Laufzeitdifferenz entsprechend acht Abtastperioden liegt der erste Einbruch der Übertragungsfunktion bei 0,5 Hz, also innerhalb des Signals. Entsprechend gravierend ist die Intersymbolinterferenz.

MATLAB:

```
...
%% Mehrwegekanal
dt = 2;   % Laufzeitdifferenz (in Samples)
a = 0.8;  % Dämpfung des indirekten Pfades
b = 1;    % Produkt f_c * t_0

% Impulsantwort
hKanal = zeros(1, dt + 1);
hKanal(1) = 1;
hKanal(dt + 1) = a*exp(-j*2*pi*b);

% Übertragungsfunktion
f = (os*(-256:255)/512);
HKanal = 1 + a*exp(-j*2*pi*b)*exp(-j*2*pi*f*dt/os);
figure;
plot(f, abs(HKanal));
xlabel('f in Hz');
ylabel('|H_K(f)|');

% Faltung mit der Kanalimpulsantwort
r = conv(hKanal, x);

figure;
periodogram(r, [], [], os, 'centered');

%% Signalangepasstes Filter
ye = conv(psrc, r);
ye = ye(dly_rc*os + 1:end - dly_rc*os);

eyediagram(ye, 2*os, 2);
```

Scilab:

```
...
//**************
// Mehrwegekanal
//**************
dt = 2;   // Laufzeitdifferenz (in Samples)
a = 0.8;  // Dämpfung des indirekten Pfades
b = 1;    // Produkt f_c * t_0
```

```
// Impulsantwort
hKanal = zeros(1, dt + 1);
hKanal(1) = 1;
hKanal(dt + 1) = a*exp(-%i*2*%pi*b);

//Übertragungsfunktion
f = (os*(-256:255)/512);
HKanal = 1 + a*exp(-%i*2*%pi*b)*exp(-%i*2*%pi*f*dt/os);
figure;
plot(f, abs(HKanal));
xlabel('f in Hz');
ylabel('|H_K(f)|');

// Faltung mit der Kanalimpulsantwort
r = conv(hKanal, x);

// Leistungsdichtespektrum (Periodogramm)
len = length(r);
len = 2*floor(len/2);
r = r(1:len);
f = (os*(-len/2:len/2 - 1)/len)';
lds = 10*log10((1/(len*os))*fftshift(abs(fft(r)).^2));
figure;
plot(f, lds)
xlabel("f in Hz");
ylabel("LDS in dB");

//************************
// Signalangepasstes Filter
//************************
ye = conv(psrc, r);
ye = ye(dly_rc*os + 1:$ - dly_rc*os);

// Augendiagramm
yi = real(ye);
yq = imag(ye);
eyemax = floor(length(yi)/(2*os));
teye = ((-os:os)/os)';
figure;
subplot(211);
for i = 1:eyemax - 1
    plot(teye, yi(2*(i - 1)*os + 1:2*i*os + 1));
end
plot(teye(1:$-1), yi(2*(eyemax - 1)*os + 1:2*eyemax*os));
xlabel("t/T_s");
ylabel("y_i");
subplot(212);
for i = 1:eyemax - 1
    plot(teye, yq(2*(i - 1)*os + 1:2*i*os + 1));
end
```

```
plot(teye(1:$-1), yq(2*(eyemax - 1)*os + 1:2*eyemax*os));
xlabel("t/T_s");
ylabel("y_q");
```

Versuch 3.2: T/2-Entzerrer

Der Einfluss des Mehrwegekanals wird durch einen Entzerrer kompensiert, der mit 2 Abtastwerten pro Symbol arbeitet (T/2-Entzerrer). Er übernimmt gleichzeitig die Funktion des signalangepassten Filters (siehe Abschnitte 5.4.2 und 6.5). Für die Bestimmung der Koeffizienten wird die MMSE-Lösung verwendet.

Der Entzerrer hat 33 Koeffizienten. Zunächst wird die vollständige Faltungsmatrix aufgestellt, anschließend wird jede zweite Zeile gestrichen. Die Berechnung der Koeffizienten erfolgt gemäß den Gln. (5.62), (5.63) und (5.64). Für die Rauschleistung σ_r^2 wird zunächst ein sehr kleiner Wert (10^{-6}) angenommen.

Der zu entzerrende Impuls $g(n)$ und die Impulsantwort des Entzerrers wird geplottet. Die Abtastrate des Eingangssignals wird auf 2 Abtastwerte pro Symbol herabgesetzt. Nun kann das Signal mit der Entzerrer-Impulsantwort gefaltet werden, und das Augendiagramm wird erzeugt.

MATLAB:

```
...
%% Mehrwegekanal
...
%% T/2-Entzerrer
Ne = 33;                    % Anzahl der Koeffizienten
dly_eq = round((Ne-1)/2);  % Verzögerung des Entzerrers

% Zu entzerrender Impuls g(n)
gn = conv(psrc, hKanal);
gn = downsample(gn, os/2);

% Rauschleistung
var_r = 1e-6;
% Faltungsmatrix mit m + 1 + N Zeilen und m + 1 Spalten
gn = gn(:);
F = toeplitz([gn; zeros(Ne-1, 1)], [gn(1) zeros(1, Ne-1)]);
% F hat z Zeilen und s Spalten, Weglassen jeder zweiten Zeile
[z, s] = size(F);
F(2:2:z, :) = [];
% Ideales Ausgangssignal
[z, s] = size(F);
yid = zeros(z, 1);
yid(round(z/2)) = 1;
% Berechnung der Koeffizienten
I = diag(ones(1, s));
en = (F'*F + var_r*I)\F'*yid;
```

```
figure;
subplot(2, 1, 1);
stem(real(gn));
ylabel('g_i(n)');
subplot(2, 1, 2);
stem(imag(gn));
ylabel('g_q(n)');
xlabel('n');

figure;
subplot(2, 1, 1);
stem(real(en));
ylabel('e_i(n)');
subplot(2, 1, 2);
stem(imag(en));
ylabel('e_q(n)');
xlabel('n');

% Eingangssignal, 2 Samples/Symbol, Entzerrung
r = downsample(r, os/2);
y = conv(en, r);
y = y(dly_eq + 1:end - dly_eq);
eyediagram(y, 4, 2);
```

Scilab:

```
...
//**************
// Mehrwegekanal
//**************
...
//**************
// T/2-Entzerrer
//**************
Ne = 33;                    // Anzahl der Koeffizienten
dly_eq = round((Ne-1)/2);  // Verzögerung des Entzerrers

// Zu entzerrender Impuls g(n)
gn = conv(psrc, hKanal);
gn = downsample(gn, os/2);

// Rauschleistung
var_r = 1e-6;
// Faltungsmatrix mit m + 1 + N Zeilen und m + 1 Spalten
gn = gn(:);
F = mtlb_toeplitz([gn; zeros(Ne-1, 1)], [gn(1) zeros(1, Ne-1)]);
// F hat z Zeilen und s Spalten, Weglassen jeder zweiten Zeile
[z, s] = size(F);
F(2:2:z, :) = [];
```

```
// Ideales Ausgangssignal
[z, s] = size(F);
yid = zeros(z, 1);
yid(round(z/2)) = 1;
// Berechnung der Koeffizienten
I = diag(ones(1, s));
en = (F'*F + var_r*I)\F'*yid;

figure;
subplot(211);
plot2d3(real(gn));
ylabel("g_i(n)");
subplot(212);
plot2d3(imag(gn));
ylabel("g_q(n)");
xlabel("n");

figure;
subplot(211);
plot2d3(real(en));
ylabel("e_i(n)");
subplot(212);
plot2d3(imag(en));
ylabel("e_q(n)");
xlabel("n");

//*********************************************
// Eingangssignal, 2 Samples/Symbol, Entzerrung
//*********************************************
r = downsample(r, os/2);
y = conv(en, r);
y = y(dly_eq + 1:$ - dly_eq);

// Augendiagramm
yi = real(y);
yq = imag(y);
eyemax = floor(length(yi)/4);
teye = ((-2:2)/2)';
figure;
subplot(211);
for i = 1:eyemax - 1
    plot(teye, yi(2*(i - 1)*2 + 1:2*i*2 + 1));
end
plot(teye(1:$-1), yi(2*(eyemax - 1)*2 + 1:2*eyemax*2));
xlabel("t/T_s");
ylabel("y_i");
subplot(212);
for i = 1:eyemax - 1
    plot(teye, yq(2*(i - 1)*2 + 1:2*i*2 + 1));
end
```

```
plot(teye(1:$-1), yq(2*(eyemax - 1)*2 + 1:2*eyemax*2));
xlabel("t/T_s");
ylabel("y_q");
```

Versuch 3.3: Symbolfehlerwahrscheinlichkeit

Wir ermitteln wieder die Symbolfehlerhäufigkeit für $E_b/N_0 = 4 \ldots 8$ dB (siehe Versuch 2.3). Am Ausgang des Mehrwegekanals erfolgt die Reduzierung der Abtastrate auf 2 Samples pro Symbol. Für die Berechnung der Entzerrerkoeffizienten verwenden wir einen mittleren Wert der Rauschleistung (0.04).

Das Rauschen wird am Ausgang des Mehrwegekanals addiert, also bei einer Abtastrate von 2 Samples pro Symbol. Dies muss bei der Berechnung des Störabstandes berücksichtigt werden.

Für $b = 1$ und eine Laufzeitdifferenz von 2 Abtastperioden kann der Entzerrer den Einfluss des Mehrwegekanals vollständig kompensieren und wir erhalten für die Symbolfehlerhäufigkeit die gleichen Werte wie im Falle des AWGN-Kanals und dem idealen Empfänger mit signalangepasstem Filter. Dies gilt aber nicht für beliebige Konfigurationen des Mehrwegekanals, insbesondere wenn der Einbruch der Übertragungsfunktion noch innerhalb der Signalbandbreite B_K liegt.

MATLAB:

```
...
EbN0List = [4:8];  % Eb/N0 in dB
serTheoList = zeros(1, length(EbN0List));
serList = zeros(1, length(EbN0List));
...
%% Mehrwegekanal
...
% Downsampling des Eingangssignals auf 2 Samples/Symbol
r = downsample(r, os/2);

%% T/2-Entzerrer
...
% Rauschleistung
var_r = 0.04;
...

%% Schleife für verschiedene Eb/N0
for i = 1:length(EbN0List)
    % AWGN-Kanal
    EbN0 = EbN0List(i);
    snr = EbN0 + 10*log10(k) - 10*log10(2);  % Störabstand in dB
    sigPwr = 0.5*mean(abs(r).^2);  % Signalleistung
    noisePwr = sigPwr/(10^(snr/10));  % Rauschleistung
    rng(1234);
    noise_i = randn(length(r), 1)*sqrt(noisePwr);
```

```
    rng(4321);
    noise_q = randn(length(r), 1)*sqrt(noisePwr);
    rn = r + noise_i + j*noise_q;  % Signal + Rauschen

    % Entzerrer und QPSK-Demodulator
    y = conv(en, rn);
    y = y(dly_eq + 1:end - dly_eq);

    y = downsample(y, 2);

    ysym = pskdemod(y, m, lambda);

    % Bestimmung der Symbolfehlerhäufigkeit
    ysym = ysym(1:length(symbols));
    [nerror, ser] = symerr(symbols, ysym);
    serList(i) = ser;
    if m <= 4
        serTheo = erfc(sqrt(10^(EbN0/10)));
    else
        serTheo = erfc(sqrt(log2(m)*10^(EbN0/10))*sin(pi/m));
    end
    serTheoList(i) = serTheo;
    fprintf('Eb/N0: %5.2f dB  Ps: %f  Symbolfehler: %4d  ...
    Symbolfehlerhäufigkeit: %f\n', EbN0, serTheo, nerror, ser);
end

%% Plot SER vs. Eb/N0
...
```

Scilab:

```
...
EbN0List = [4:8];  // Eb/N0 in dB
serTheoList = zeros(1, length(EbN0List));
serList = zeros(1, length(EbN0List));
...
//**************
// Mehrwegekanal
//**************
...
// Downsampling des Eingangssignals auf 2 Samples/Symbol
r = downsample(r, os/2);

//**************
// T/2-Entzerrer
//**************
...
// Rauschleistung
var_r = 0.04;
...
```

```
//*******************************
// Schleife für verschiedene Eb/N0
//*******************************
for i = 1:length(EbN0List)
    // AWGN-Kanal
    EbN0 = EbN0List(i);
    snr = EbN0 + 10*log10(k) - 10*log10(2);  // Störabstand in dB
    sigPwr = 0.5*mean(abs(r).^2);  // Signalleistung
    noisePwr = sigPwr/(10^(snr/10));  // Rauschleistung
    rand("seed", 1234);
    noise_i = rand(length(r), 1, "normal")*sqrt(noisePwr);
    rand("seed", 4321);
    noise_q = rand(length(r), 1, "normal")*sqrt(noisePwr);
    rn = r + noise_i + %i*noise_q;  // Signal + Rauschen

    // Entzerrer und QPSK-Demodulator
    y = conv(en, rn);
    y = y(dly_eq + 1:$ - dly_eq);

    y = downsample(y, 2);

    // Demodulation
    y = y.*exp(-%i*lambda);
    ysym = round(m/(2*%pi)*atan(imag(y), real(y)));
    ysym(ysym < 0) = ysym(ysym < 0) + m;

    // Bestimmung der Symbolfehlerhäufigkeit
    ysym = ysym(1:length(symbols));
    nerror = sum(ceil(bitxor(symbols, ysym)/m));
    ser = nerror/length(ysym);
    serList(i) = ser;
    serTheo = erfc(sqrt(10^(EbN0/10)));
    serTheoList(i) = serTheo;
    mprintf('Eb/N0: %5.2f dB  Ps: %f  Symbolfehler: %4d  ..
    Symboltfehlerhäufigkeit: %f\n', EbN0, serTheo, nerror, ser);
end

//*******************
// Plot SER vs. Eb/N0
//*******************
...
```

Literatur

[1] Hoffmann, j., Quint, F.: *Signalverarbeitung mit MATLAB und Simulink*. Oldenbourg Verlag, 2. Aufl., 2012.

[2] Stewart, R. W., Barlee, K. W., Atkinson, D. S. W., Crockett, L. H.: *Software Defined Radio using MATLAB & Simulink and the RTL-SDR*. Strathclyde Academic Media, 2015.