Delta-Sigma-Modulation

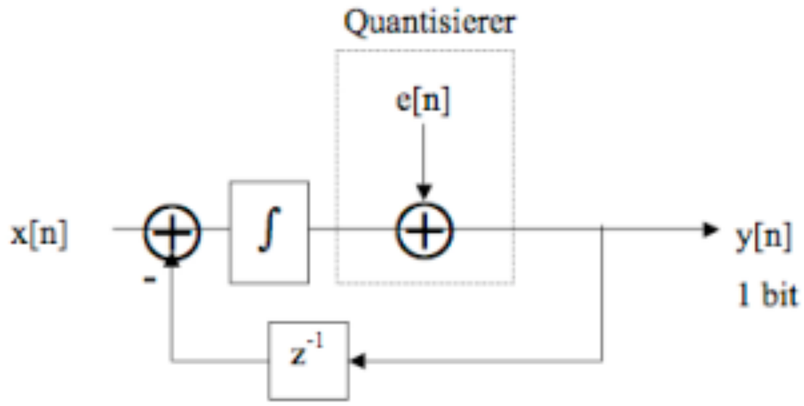


Abb. 1: Schaltbild eines Delta-Sigma-Modulators

x[*n*] und *y*[*n*] seien normiert auf einen Amplitudenbereich von [-1,1]. Am Quantisierer findet eine Quantisierung auf eine Wortbreite von 1 bit statt. x[n] ist eine abgetastete, aber nicht quantisierte Zahlenfolge.

a. Erläutern Sie kurz die Funktionsweise eines Delta Sigma Modulators.

Der δσ-Modulator ist Grundbaustein von Sigma-Delta-A/D- und -D/A-Wandlern. Es wird die Differenz („Delta“) zwischen Eingangs- und Ausgangssignal integriert („Sigma“) und bei sehr hoher Abtastrate auf sehr niedrige Wortbreite quantisiert (s. Abb. 1). Dabei wird der Quantisierungsfehler Spektral geformt.
Durch eine Abtastung des Eingangssignals mit dem L-fachen der gewünschten Abtastrate lassen sich trotz der niedrigen Wortbreite sehr gute SNRs realisieren.

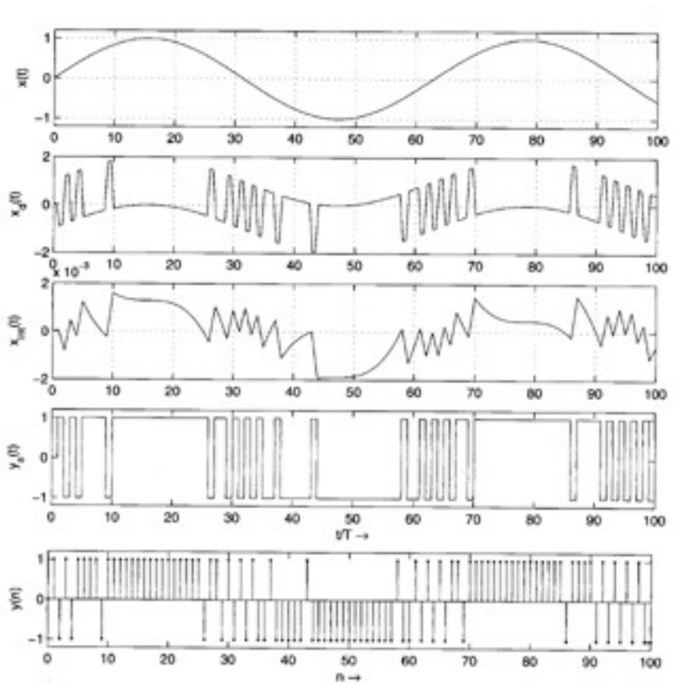


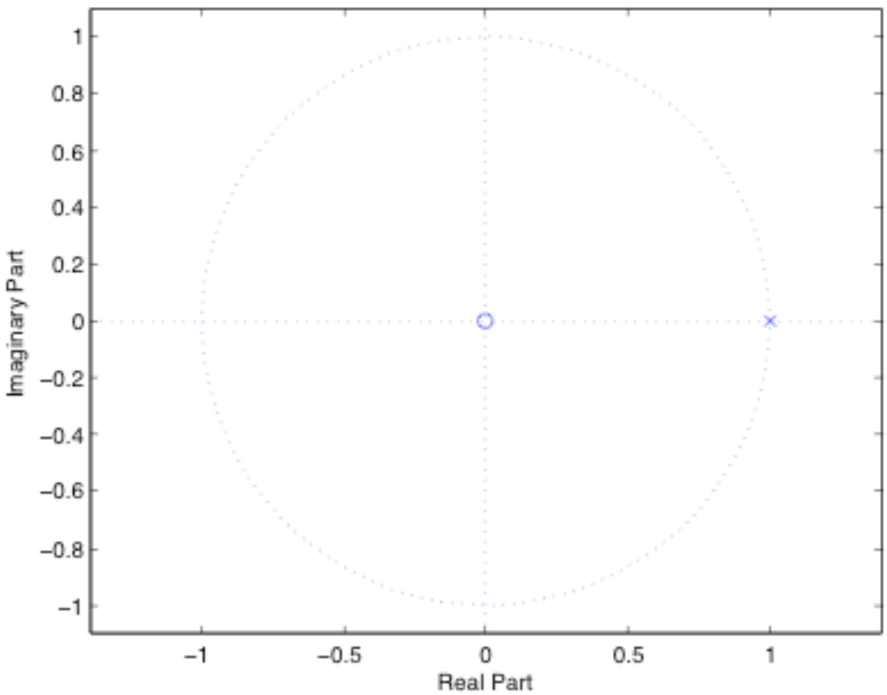
Abb. 2: Delta-Sigma-Modulation (Quelle: Zölzer (2005))

b. Welche Filter werden benötigt, wenn eine A/D-Wandlung mit Hilfe eines Delta Sigma Modulators realisiert werden soll? Zeichnen Sie das dazugehörige Blockschaltbild.

c. Die Übertragungsfunktion des Integrieres sei $H(z) = \frac{1}{1-z^{-1}}$. Um was für eine Art Filter handelt es sich.

Um abschätzen zu können, um was für eine Art Filter es sich handelt werden Pol- und Nullstellen berechnet und im Pol- Nullstellen-Diagramm dargestellt.

$$H(z) = \frac{1}{1-z^{-1}} = \frac{z}{z-1}$$
$$\Rightarrow n_1 = 0 \qquad und \qquad p_1 = 1$$



Wegen der Polstelle bei 1 (was der Frequenz f=0 Hz entspricht) handelt es sich um eine Art Tiefpass. Daher auch die Bezeichnung Integrierer. Die Übertragungsfuntion kann einfach mit Matlab geplottet werden: freqz([1 0], [1 -1], 512)

d. Stellen Sie die Differenzengleichung des Sigma-Delta-Modulators im Zeitbereich dar.

Differenzengleichung mit *h*(*n*) als Impulsantwort des Integrierers:

$$y(n) = \left[x(n) - y(n-1)\right] * h(n) + e(n)$$

In den z-Bereich transformiert:

$$Y(z) = \left[X(z) - z^{-1}Y(z)\right] \cdot H(z) + E(z) = X(z)H(z) - z^{-1}Y(z)H(z) + E(z)$$

$$Y(z) + z^{-1}Y(z)H(z) = X(z)H(z) + E(z)$$

$$Y(z)\left[1 + z^{-1}H(z)\right] = X(z)H(z) + E(z)$$

$$Y(z) = \frac{H(z)}{1 + z^{-1}H(z)} X(z) + \frac{1}{1 + z^{-1}H(z)} E(z)$$

mit $H(z) = \frac{1}{1 - z^{-1}}$:

$$\frac{\frac{1}{1-z^{-1}}}{1 + z^{-1}\frac{1}{1-z^{-1}}} X(z) + \frac{1}{1 + z^{-1}\frac{1}{1-z^{-1}}} E(z) = \frac{\frac{1}{1-z^{-1}}}{\frac{1-z^{-1}}{1-z^{-1}} + \frac{z^{-1}}{1-z^{-1}}} X(z) + \frac{1}{\frac{1-z^{-1}}{1-z^{-1}} + \frac{z^{-1}}{1-z^{-1}}} E(z)$$

$$= \frac{\frac{1}{1-z^{-1}}}{\frac{1}{1-z^{-1}}} X(z) + \frac{1}{\frac{1}{1-z^{-1}}} E(z) = X(z) + (1 - z^{-1})E(z)$$

Dies entspricht Noiseshaping erster Ordnung!

Zurück transformiert in den Zeitbereich:

$$y(n) = x(n) - e(n-1) + e(n)$$

e. Berechnen Sie den Ausgang *y*[*n*] für ein 5 Samples langes Eingangssignal *x*[*n*] mit der konstanten Amplitude 0.7.

$$y(n) = x(n) - e(n-1) + e(n) = [x(n) - e(n-1)]_Q$$

$$e(n) = y(n) - [x(n) - e(n-1)]$$

Berechnung der Ausgangswerte

$$y(0) = \left[x(0) - e(-1)\right]_Q = \left[0{,}7 - 0\right]_Q = \left[0{,}7\right]_Q = 1$$

$$e(0) = 1 - 0{,}7 = 0{,}3$$

$$y(1) = \left[x(1) - e(0)\right]_Q = \left[0{,}7 - 0{,}3\right]_Q = \left[0{,}4\right]_Q = 1$$

$$e(1) = 1 - 0{,}4 = 0{,}6$$

$y(2) = \left[x(2) - e(1)\right]_Q = \left[0{,}7 - 0{,}6\right]_Q = \left[0{,}1\right]_Q = 1$

$e(2) = 1 - 0{,}1 = 0{,}9$

$y(3) = \left[x(3) - e(2)\right]_Q = \left[0{,}7 - 0{,}9\right]_Q = \left[-0{,}2\right]_Q = -1$

$e(3) = -1 + 0{,}2 = -0{,}8$

$y(4) = \left[x(4) - e(3)\right]_Q = \left[0{,}7 + 0{,}8\right]_Q = \left[1{,}5\right]_Q = 1$

$e(4) = 1 - 1{,}5 = -0{,}5$

f. Implementieren Sie die Differenzengleichung aus d) in einer Matlabfunktion

   [y1bit, e] = dsm(x)

   Testen Sie die Funktion mit einem voll ausgesteuertem Sinussignal mit einer Periodendauer von 20 Samples, sowie dem Signal aus e) und stellen Sie Eingangs- und Ausgangssignal dar.

Der Code für die Funktion dsm.m ist in Lösungsteil g) gegeben.

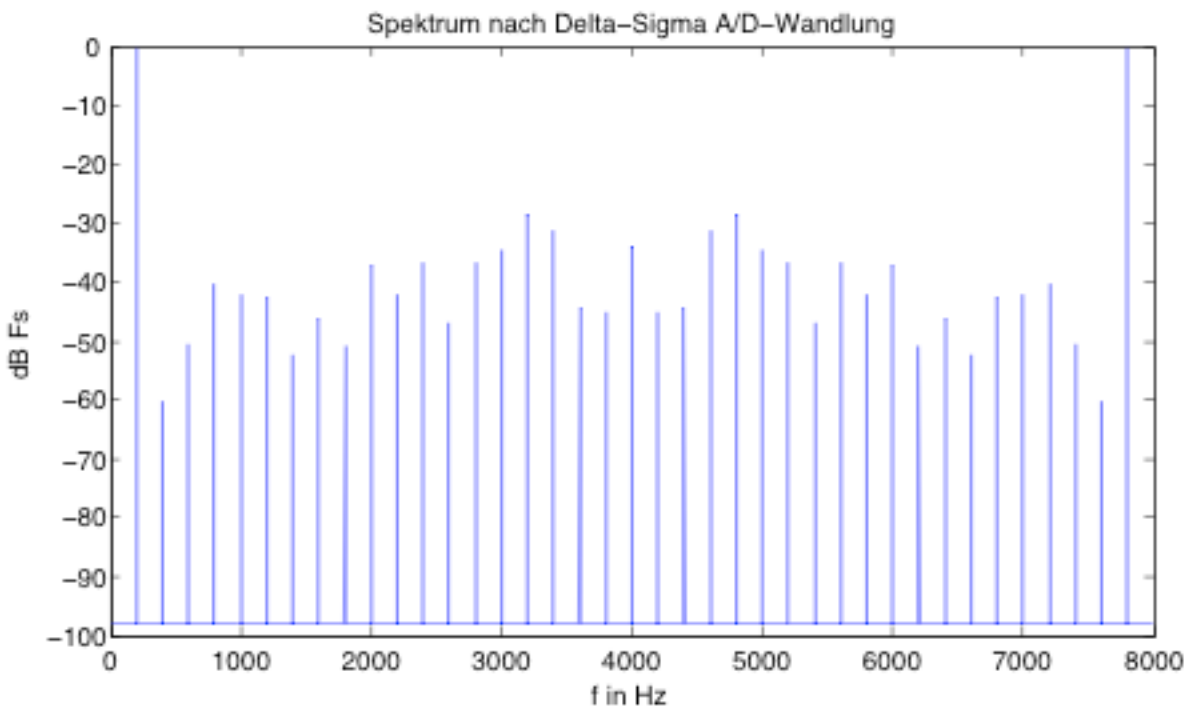
```
% Erzeuge konstantes Eingangssignal
x1 = ones(1,20)*.7;


% Erzeuge Sinus
% Zeitvektor f?r eine Sinus-Periode
t = 0:1/20:1-1/20;

% Erzeuge mit 16 bit quantisiertes Eingangssignal
x2 = sin(2*pi*t);


[y1, e1] = dsm2(x1);
[y2, e2] = dsm2(x2);


%% Plots
figure(1)
subplot(2,2,1), stem(x1), title('Eingangsignal Aufg b)')
xlabel('samples'), ylabel('Amplitude'), axis([0 20 -1 1]), grid on
subplot(2,2,2), stem(y1), title('Ausgangssignal Aufg b)')
xlabel('samples'), ylabel('Amplitude'), grid on
subplot(2,2,3), stem(x2), title('Eingangsignal Aufg c)')
xlabel('samples'), ylabel('Amplitude'), axis([0 20 -1 1]), grid on
subplot(2,2,4), stem(y2), title('Ausgangssignal Aufg c)')
xlabel('samples'), ylabel('Amplitude'), grid on
```

g. Erweitern Sie die Matlabfunktion um ein Dezimationsfilter (Zölzer: Digitale Audiosignalverarbeitung, 2005 Gl. 3.30) mit anschließender Unterabtastung. Testen Sie die Funktion mit einem geeigneten Signal und plotten Sie dessen Spektrum.

Das Dezimationsfilter ist gegeben durch

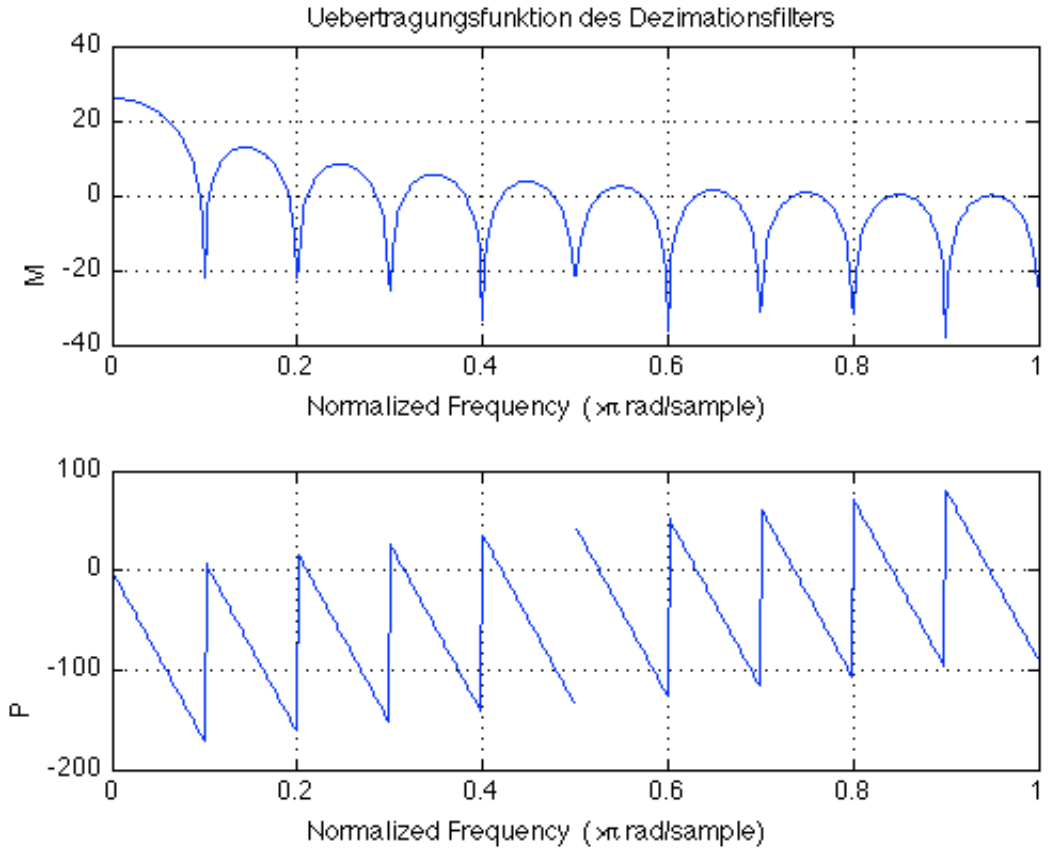
$$H_{TP}(z) = \frac{1}{L} \cdot \frac{1 - z^{-L}}{1 - z^{-1}}$$

```
%% g)
L  = 20;
fs = 8000;
f  = 200;
n  = 1:L*fs-1;

x3 = sin(n*2*pi*f/(L*fs));

[y3_1bit, e3, y3] = dsm(x3, L);

Y3 = abs(fft(y3)/max(fft(y3)));
f  = 0:fs/length(y3):fs-fs/length(y3);


%% plot
figure(2)
    plot(f, 20*log10(Y3))
    xlabel('f in Hz'); ylabel('dB Fs'); title('Spektrum nach Delta-Sigma
A/D-Wandlung')
```



Die bei ca. -30 dB befindlichen Obertöne der 200 Hz Schwingung entstehen durch die geringe Sperrdämpfung des verwendeten Dezimationsfilter (siehe

Frequenzgang). Um diese zu verbessern werden üblicherweise mehrere Filter kaskadiert.



Funktion dsm:

```
function [y1bit, e, y] = dsm(x, L)

% Differenzengleichung:
% y(n) = x(n) + e(n) - e(n-1)
% y(n) = [x(n) - e(n-1)]_Quantisiert
% e(n) = y(n)-x(n)

% Initialisiere Fehlervektor und Ausgabevektor
y1bit = zeros(size(x));
e = zeros(size(x));
e_buf = 0;

% Berechne 1 Bit Ausgangssignal
for n = 1:length(x);
    % bestimme den aktuellen Zustand vor dem Quantisierer:
    zustand = x(n)-e_buf;

    % 1-Bit-Quantisierung des Signals,
    % hierbei entsteht auch der Quantisierungsfehler e(n)
    if zustand >= 0
```

```
        y1bit(n) = 1;
    else
        y1bit(n) = -1;
    end

    %Bestimme den Quantisierungsfehler e(n)
    e_buf = y1bit(n) - zustand;
    e(n)  = e_buf;
end

% Dezimation und Unterabtastung
if nargin > 1

    % H(z) = 1/L * (1-z^-L / 1-z^-1)

    % Filterkoeffizienten erstellen
    numCoeff       = zeros(1, L+1);
    numCoeff(1)    = 1;
    numCoeff(L+1) = -1;

    denumCoeff     = [1 -1];

    % Filtern
    y = 1/L * filter(numCoeff, denumCoeff, y1bit);

    % Unterabtasten
    y = y(1:L:end);


    % Uebertrgungsfunktion des Dezimationsfilters
    freqz(numCoeff, denumCoeff, 512)
    title('Uebertragungsfunktion des Dezimationsfilters')

end
```