# Modul 5033 Digitale Signalverarbeitung Dozent: Rolf Vetter

# 

### *Praktische Übung No 1*

# Erzeugung digitaler Signale, Abtastung

### Ziel

* Einführung in MATLAB, wie es in der digitalen Signalverarbeitung gebraucht wird
  + Signalerzeugung und graphische Darstellungen
* Einführung in die Datenerfassung mit MATLAB
* Illustration des Abtasttheorems mittels kommerziellen Datenerfassungskarten

### Aufgaben

*Vorbemerkung zu MATLAB*

Der Studierente soll MATLAB starten und im Befehlsmenu Help anklicken. Automatisch öffnet sich ein sehr umfangreicher „online help“ und unter Getting Started findet der Studierende alle nützlichen Informationen. Eine umfangreiche deutsche MATLAB Einführung kann z.B. auf <http://people.inf.ethz.ch/arbenz/MatlabKurs/> gefunden werden. In dieser Übung werden die Studierenden mit Hilfe einiger Beispiele DSV-orientiert in MATLAB eingeführt.

*Signalerzeugung (fakultativ)*

1. Öffnen Sie durch Anklicken von File => New => Script im Befehlsmenu eine neue Datei, in der alle Ihre Befehle gespeichert werden. Erzeugen Sie ein Sinussignal durch die untenstehende Befehlssequenz. Kopieren Sie diese Befehle in Ihr Skript. Speichern Sie das Skript durch den Save As Befehl als SinusErzeugung.m und führen Sie das Skript durch den Run Befehl (grüner Pfeil im Befehlsmenu) aus.

fa=8000; % Abtastfrequenz (% leitet Kommentar ein)

N=10; % Anzahl Samples

t=[1 :N ]/fa % Diskreter Zeitvektor

fo=1000; % Frequenz der Sinusschwingung

x=sin(2\*pi\*fo\*t) % Erzeugung von einem Sinussignal mit Frequenz fo

Im Command Window werden nun die berechneten zeitdiskreten Vektoren t und x angezeigt. Das Abschliessen einer Befehlslinie mit dem Semikolon verhindert diese Anzeige. Das Signal kann nun auf zwei verschiedene Arten graphisch dargestellt werden:

figure

subplot(2,1,1)

plot(t,x) % Zeitdiskrete Samples sind mit Linien verbunden

xlabel('t [s]'); ylabel('Signal [-]')

subplot(2,1,2)

stem(t,x) % Klare Darstellung der zeitdiskreten Struktur

xlabel('t [s]'); ylabel('Signal [-]')

Führen Sie das ganze Skript durch den Run Befehl aus. Erhöhen Sie die Anzahl Samples N auf 8000 und führen Sie das Skript nochmals aus.

1. Öffnen Sie ein neues Skript RechtecksignalErzeugung.m (d.h. durch den Save As Befehl des SinusErzeugung.m), erzeugen Sie ein Rechtecksignal (Befehl square()) und stellen Sie es graphisch dar.
2. Erzeugen Sie verschiedene Realisierungen eines Zufallssignals durch das Ausführen der Anleitung des Paragraphen 2.1.10 des Skriptes „Signale und Systeme mit MATLAB“. Feststellungen?

*Abtastung*

1. Entwerfen Sie ein Messprotokoll, das Ihnen erlaubt, das Abtasttheorem mit einem Signalgenerator eines Smartphones, einer Datenerfassungskarte ohne Antialiasing Filter (NI USB 6008/6009), eines PC-Standartaudioausganges und der Data Acquisition Toolbox von MATLAB, zu illustrieren.
   1. Zeichnen Sie das Konzeptschema.
   2. Schreiben Sie das MATLAB Skript (MATLAB 64Bits, Session-Based-Interface, Acquiring Data in the Foreground).
   3. Überprüfen Sie die spektralen Spiegelungen (Aliasing).
2. Zeigen Sie, dass die Signale mit demselben Protokoll wie im Punkt 4 getreu erfasst werden können, wenn ein korrekt dimensioniertes AAF verwendet wird.
3. Wie kann man die Quantisierung der kommerziellen Datenerfassungskarte NI USB 6008/6009 am einfachsten hervorheben? Entsprechen die Messungen den theoretischen Werten?

MATLAB

help Hilfe « on-line »

lookfor Schlüsselwort Suche in allen Funktionen nach dem Schlüsselwort

t=[0 :N-1 ]/fa Erzeugung eines disktreten Zeitvektors mit Abtastfrequenz fa und Länge N

x=sin(2\*pi\*fo\*t) Erzeugung eines Sinussignals mit Frequenz fo

x=exp(a\*t) Erzeugung eines exponentiellen Signals

x=exp(j\*2\*pi\*fo\*t) Erzeugung eines komplexen exponentiellen Signals

xr=real(x), xi=imag(x) Reeller und imaginärer Teil des Signals x

plot(t,x), stem(t,x) Graphische Darstellung im Zeitbereich

plot(xr,xi) Graphische Darstellung in der komplexen Ebene

[X,f]=fft(x) Schnelle diskrete Fourier Transformation (Fast Fourier Transform)

psd(x) Leistungsdichtespektrum

xcorr, xcov. mean, var Statistische Operatoren

median Median Operator

*Beispiel der Berechnung eines Spektrums mittels der FFT*

fo=1;fa=100;Dauer =100/fo; % Simulationsparameter

t=[0:1/fa:Dauer]'; x=sin(2\*pi\*fo\*t); % Signalerzeugung Sinus

deltaf=0.01; % Gewünschte Frequenzauflösung in Hz

n\_t=fa/deltaf; % Anzahl diskrete Stützpunkte im Frequenzbereich

Xf=fft(x,n\_t);f=[0:n\_t-1]\*deltaf-fa/2;

figure; % Anzeige

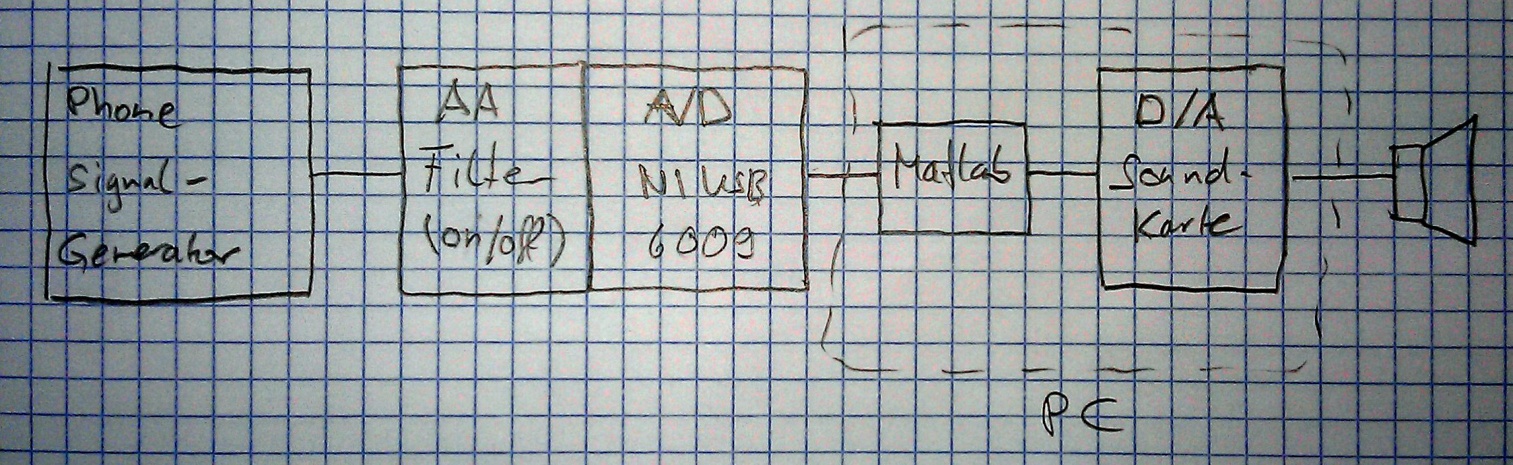
plot(f,abs(fftshift(Xf))/length(Xf));

ylabel('|X(f)| [\_]');xlabel('f [Hz]');

# Arbeitsjournal

### Aufgabe 4

Das Konzeptschema der Illustration:



Matlab Code:

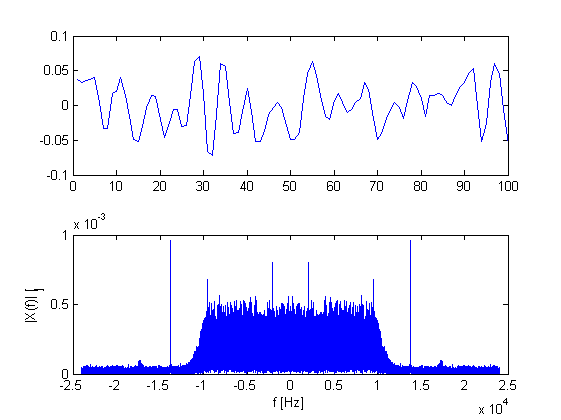
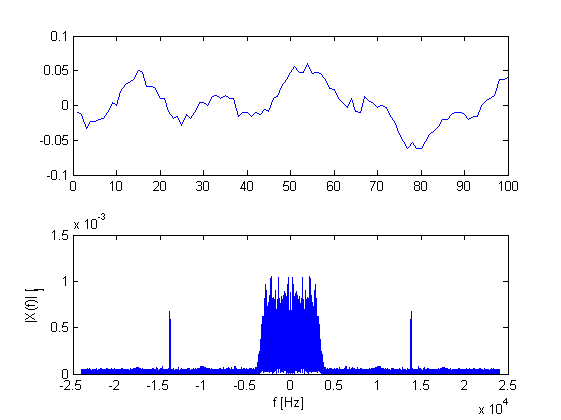
|  |
| --- |
| s = daq.createSession('ni');  s.addAnalogInputChannel('Dev2','ai0','Voltage');  s.Rate = 8000;  s.DurationInSeconds = 1;    figure    while 1  data = s.startForeground();  plot(data(1:21));  s.stop();  soundsc(data);  end |

Es wird immer eine Sekunde lang das Signal mit 8kS/s abgetastet, eingelesen und anschliessend auf die Soundkarte ausgegeben. Wenn wir das Eingangssignal von 1kHz bis 10kHz erhöhen und das AAF auf 10kHz haben, wird das Signal ziemlich unbeeinflusst in das Fenster 0kHz – 8kHz heruntergespiegelt.

### Aufgabe 5

Wird das oben erwähnte Filter jedoch auf 3.3kHz eingestellt, wird schon die Frequenz 4kHz deutlich leiser. Um diesen Effekt zu bemerken, muss jedoch die Zeile soundsc(data) durch sound(data) ersetzt werden (sonst werden die leiseren Daten einfach aufskaliert. In diesem Fall hört man hingegen auch deutlich dass das Signal-Rausch-Verhältnis reduziert ist – starkes Rauschen auf dem Signal).

Das spektrale Filterverhalten des AAF kann durch Einspeisen eines weissen Rauschens und einer Spektrumanalyse in MATLAB gezeigt werden.

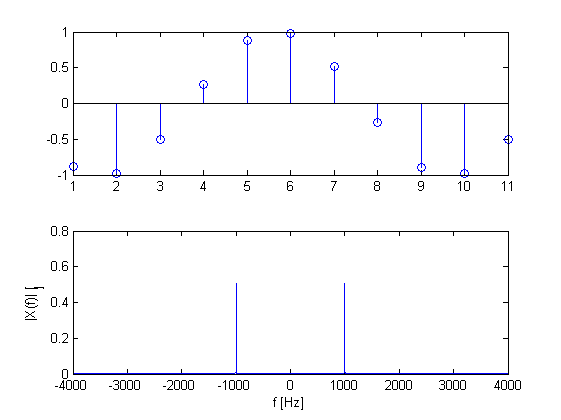
3.3kHz Filter Spektrum 10kHz Filter Spektrum  


Matlab Code:

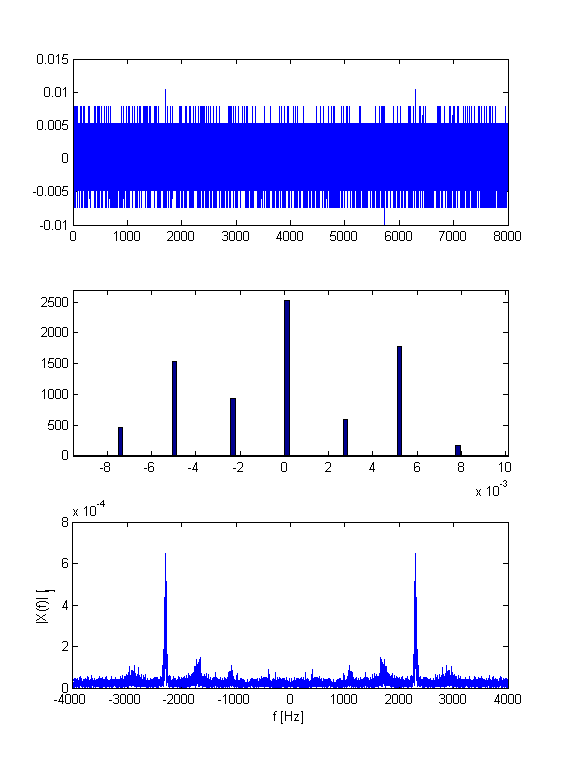
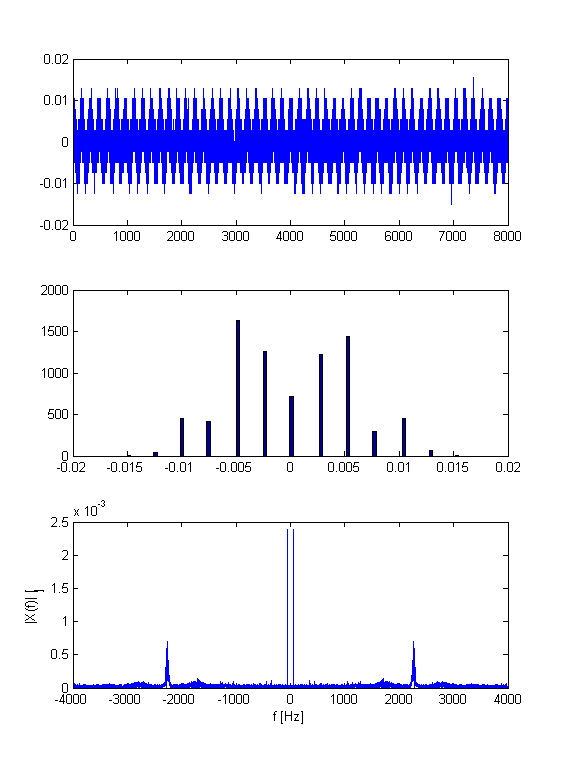
|  |
| --- |
| s = daq.createSession('ni');  s.addAnalogInputChannel('Dev3','ai0','Voltage');  s.Rate = 48000;  s.DurationInSeconds = 1;    figure    while 1  data = s.startForeground();  data = data - mean(data);    subplot(2,1,1), plot(data(1:100));      fa=48000;Dauer =1; % Simulationsparameter    deltaf=1; % Gewünschte Frequenzauflösung in Hz  n\_t=fa/deltaf; % Anzahl diskrete Stützpunkte im Frequenzbereich  Xf=fft(data,n\_t);f=[0:n\_t-1]\*deltaf-fa/2;    subplot(2,1,2), plot(f,abs(fftshift(Xf))/length(Xf));  ylabel('|X(f)| [\_]');xlabel('f [Hz]');    s.stop();  sound(data);  end |

### Aufgabe 6

Die Diskretisierung in der Zeit wird gut sichtbar, wenn man ein schnelles Signal einspeist, welches nur mit wenigen Abtastpunkten pro Periode eingelesen wird: Hier wurde ein 2kHz Signal mit 8kHz Samplefrequenz abgetastet.



Die Aufgabe aber war, die Quantisierung des Signals darzustellen. Das haben wir gelöst, indem wir das Rauschen (welches nur die untersten 2-3 Bits betrifft) eines offenen Eingangs dargestellt haben. Um zu zeigen, dass zwischen den aufgenommenen Spannungswerten *keine* weiteren Schritte sind, haben wir alle Datenwerte in einem Histogramm dargestellt. Darin sieht man, dass nur gewisse Werte vorkommen.

Rauschen mit 50Hz überlagert Nur Eingangsrauschen  


Matlab Code:

|  |
| --- |
| clear;  close all;    s = daq.createSession('ni');  s.addAnalogInputChannel('Dev3','ai0','Voltage');  s.Rate = 8000;  s.DurationInSeconds = 1;    figure    while 1  data = s.startForeground();  data = data - mean(data);    subplot(3,1,1), plot(data);  subplot(3,1,2), hist(data,100)    fa=8000;Dauer =1; % Simulationsparameter    deltaf=1; % Gewünschte Frequenzauflösung in Hz  n\_t=fa/deltaf; % Anzahl diskrete Stützpunkte im Frequenzbereich  Xf=fft(data,n\_t);f=[0:n\_t-1]\*deltaf-fa/2;    subplot(3,1,3), plot(f,abs(fftshift(Xf))/length(Xf));  ylabel('|X(f)| [\_]');xlabel('f [Hz]');        s.stop();  sound(data);  end |