Fakultät für Technik, Studiengänge EIT/TI
Dipl.-Ing.(FH) Felix Becker

Laborversuch

DFT / FFT

*Einen weiteren wichtigen Baustein in digitalen Signalverarbeitungssystemen stellt die **Diskrete Fourier-Transformation - DFT** als Approximation der kontinuierlichen Fourier-Transformation dar.*
Der Gütegrad dieser Approximation hängt streng von dem zu transformierenden Signal ab. Die Unterschiede zwischen den beiden Transformationen ergeben sich durch die zur Berechnung der DFT notwendigen Maßnahmen der Abtastung und der Zeitbegrenzung [1].
*Im ersten Teil dieses Laborversuchs sollen sowohl die **Eigenschaften** der diskreten Fourier-Transformation als auch das **Ergebnis** mit Hilfe von **MATLAB** näher untersucht werden. Dabei wird insbesondere auf die meist notwendige Fensterung des zu transformierenden Signals eingegangen.*
*Der optionale zweite Teil hat die **Realisierung** der DFT, d.h. die Implementierung **auf dem DSK6713** zum Gegenstand.*

1. Theorie

1.1 Definition der DFT und die Voraussetzungen

Im Gegensatz zur Definitionsgleichung der kontinuierlichen Fourier-Transformation, in der das Fourier-Integral alle Funktionswerte von -∞ bis +∞ umfasst, kann in einem digitalen Signalverarbeitungssystem immer nur ein begrenzter Ausschnitt eines abgetasteten Zeitsignals, im folgenden Datensatz genannt, bearbeitet werden.

Unter der Voraussetzung, dass

1.) es sich bei dem abzutastenden Zeitsignal um ein periodisches und bandbegrenztes Signal handelt

und

2.) sich darüber hinaus der durch Abtastung gewonnene Datensatz (durch Aneinanderreihung) auch noch periodisch ohne Sprungstelle (Unstetigkeitsstelle) fortsetzen lässt,

erhält man aus einem solchen Datensatz der Länge N über

$$\underline{X}(n) = \frac{1}{N}\sum_{k=0}^{N-1}\underline{x}(k)\cdot e^{-j\frac{2\pi}{N}kn} = \frac{1}{N}\sum_{k=0}^{N-1}\underline{x}(k)\cdot W_N^{kn} \quad mit \quad n = 0, 1, ..., N-1 \qquad (1.1)$$

ein diskretes und periodisches komplexes Spektrum, das ebenfalls N Frequenzwerte umfasst. Jede Periode dieser N Frequenzwerte gibt dabei das relevante (endliche) Frequenzspektrum des Datensatzes wieder.

Zu beachten ist, dass in der Definitionsgleichung von einer komplexen Eingangsfolge $\underline{x}(k)$ ausgegangen wird; dazu später mehr (Kapitel 4.2).

Der komplexe Drehoperator W_N lässt sich leicht in der komplexen Ebene veranschaulichen. Der Einheitskreis wird dabei von N Zeigern in gleich große Sektoren geteilt. Jeder dieser Zeiger repräsentiert eine Potenz von W_N^k. Mit $k = 0, 1, ..., N\text{-}1$ durchläuft der Drehoperator den Einheitskreis ähnlich einem Uhrzeiger.

$$W_N^k = e^{-j\frac{2\pi}{N}k} = \cos\left(\frac{2\pi}{N}k\right) - j\sin\left(\frac{2\pi}{N}k\right) \tag{1.2}$$

Und mit $n = 0, 1, 2, 3, ..., N–1$ (siehe Definitionsgleichung 1.1) wird der Einheitskreis dann immer öfter, allerdings auch mit immer größerer Schrittweite ($k \cdot n$), durchlaufen.

1.2 Eigenschaften der DFT

1.2.1 Frequenzauflösung

Der zu transformierende Datensatz der Länge N wird durch Abtastung des Zeitsignals mit der Abtastfrequenz f_a gewonnen. Da sich daraus über die DFT ein Spektrum errechnet, das ebenfalls N Frequenzwerte umfasst, beträgt die Frequenzauflösung $\boldsymbol{\Delta f = f_a/N}$.

Beispiel:

Sie berechnen für ein Audiosignal, das Sie mit f_a = 48 kHz abgetastet haben, das Spektrum mit einer 1024-Punkte-DFT. Die Frequenzauflösung beträgt dann nur $\Delta f =$ 46,875 Hz.
($512 \cdot \Delta f =$ 24 kHz $= f_a/2 = f_n$)

1.2.2 Amplitude der Frequenzwerte

Unter Einhaltung der in Kapitel 1.1 genannten Voraussetzungen (und nur dann) erreicht die Amplitude der einzelnen Frequenzwerte die erwartete Höhe (siehe nachfolgende Beispieluntersuchungen).

1.2.3 Leakage-Effekt

In der Praxis wird das vorgegebene Zeitsignal in der Regel nicht die oben angegebene Voraussetzung der Periodizität erfüllen. Insbesondere ist der Fall eher unwahrscheinlich, dass genau eine ganze Anzahl von Perioden im Datensatz enthalten ist.
Oder im Frequenzbereich ausgedrückt: die im Datensatz enthaltenen Frequenzen betragen in der Regel kein ganzzahlig Vielfaches der Frequenzauflösung der N-Punkte-DFT.

Als Folge dieser **Voraussetzungsverletzung** stellt sich im berechneten Spektrum von z.B. einer Sinusschwingung, deren Frequenz kein ganzzahlig Vielfaches der Frequenzauflösung der N-Punkte-DFT beträgt ($f_{sin} \neq n \cdot \Delta f$), ein Zerlaufen des eigentlichen (wahren) Maximums über sämtliche Indexwerte ein. Das resultierende Maximum erreicht dann auch nicht die erwartete Amplitude (100%). Diese Eigenschaft der DFT wird **Leck-Effekt** bzw. im Englischen Leakage-Effekt genannt.

Um diesen Effekt zu verringern, kann der Datensatz vor der Transformation mit einer Fensterfunktion multipliziert werden. Dieser Vorgang wird auch **Fensterung** genannt, wobei einige bekannte Fensterfunktionen wie z.B. Hamming-, Hann-, Blackman-, Bartlett-, Kaiser-Window mit günstigen Eigenschaften zur Verfügung stehen. Allerdings gilt auch hier (wie so oft), keine Wirkung ohne Nebenwirkung (siehe Erläuterung Kapitel 3.4 und die Beispieluntersuchung in Kapitel 3.5).

2. Der Begriff FFT

Bei der Fast Fourier-Transformation - FFT handelt es sich um schnellere Berechnungsvarianten der DFT. Die FFT liefert also dieselben Ergebnisse wie die DFT, nur mit weniger Rechenaufwand (und damit möglicherweise auch etwas genauere Ergebnisse – je nach verwendeten Datentypen und zur Berechnung benutztem Prozessor).
Im Laufe der letzten drei Jahrzehnte wurden viele verschiedene Algorithmen zur FFT-Berechnung entwickelt. Eine grundlegende Arbeit dazu legten J.W. Cooley und J.W. Tukey im Jahr 1965 vor, wobei die Idee zur FFT bereits auf C.F. Gauß (1777 – 1855) zurückgeht.

Die Grundidee der FFT besteht in der Zerlegung von N in Faktoren, wobei der Fall von Zweierpotenzen ($N = 2^R$) ein Optimum darstellt. Mit steigendem N wächst der Berechnungsaufwand dann nicht mehr wie bei der DFT nahezu quadratisch an, sondern nur noch mit $N \cdot ld(N)$.

Da es sich um zahlreiche verschiedene Berechnungsvarianten mit teilweise aufwändigen Herleitungen handelt, muss hier auf entsprechende Literatur verwiesen werden:

[1] Brigham E.O. (1974) Prentice-Hall Inc. deutsche Fassung: Oldenbourg Verlag
The Fast Fourier Transform FFT

[2] Cooley J.W., Tukey J.W. (1965) Math. Computation. Vol. 19 pp. 297-301
An Algorithm for the Machine Calculation of Complex Fourier Series

[3] Burrus C.S., Parks T.W. (1985) John Wiley & Sons
DFT/FFT and Convolution Algorithms

[4] Kaiser J.F., Mitra S.K. (1993) John Wiley & Sons
Handbook for Digital Signal Processing

u.v.m.

Im Folgenden sind einige in diesem Zusammenhang auftretende Begriffe aufgelistet:

Butterfly
Decimation in Time – DIT
Decimation in Frequency – DIF
Bit-reversal
Digit-reversal
In-place calculation
Radix R (Radix 2, Radix 4, Radix 8, Radix 16)
Prime

Die verschiedenen Algorithmen sind unter folgenden Namen bekannt:

Goertzel
DIT Radix 2, DIF Radix 2
DIT / DIF Radix 4, 8, …
Split-Radix, Mixed-Radix, Prime-Factor
Winograd

3. Frequenzuntersuchung mit Hilfe von MATLAB

3.1 Die MATLAB-Funktion fft.m

Mit Hilfe der MATLAB-Funktion `fft` kann die Diskrete Fourier-Transformierte, also das komplexe Spektrum eines Datensatzes `x` berechnet werden.

Aufruf der Funktion:

```
X = fft(x, N);
```

Der optionale Parameter `N` bestimmt die gewünschte Anzahl von Frequenzwerten. Hat der übergebene Datensatz `x` weniger als `N` Elemente, dann wird `x` (vor der Berechnung) am Ende mit Nullen ergänzt. Hat er mehr, dann werden nur die ersten `N` Elemente des Datensatzes verwendet.
Ist `N` ein Vielfaches von 2 oder in kleine Primfaktoren zerlegbar, dann wird zur Berechnung des Spektrums anstelle des allgemeinen DFT-Algorithmus ein schnellerer Split-Radix-FFT-Algorithmus verwendet.

Standardmäßig erfolgt keine Normierung der Frequenzwerte. Je nach Bedarf kann `X` nachträglich durch `N` bzw. `N/2` dividiert werden. Im letzten Fall erreicht die Amplitude des berechneten Frequenzwertes einer Sinusschwingung, welche die in Kapitel 1.1 genannten Voraussetzungen erfüllt, die im Datensatz vorgegebene Amplitude (das ist die sinnvolle Normierung für die Darstellung des einseitigen Spektrums, siehe nachfolgende Beispieluntersuchungen). Zu beachten ist allerdings die Skalierung des ersten Frequenzwertes (sog. DC-Bin), die nicht mit `N/2` sondern mit `N` erfolgen muss!

3.2 Erste Beispieluntersuchung

3.2.1 Signalgenerierung

Die erste Beispieluntersuchung soll anhand eines Datensatzes erfolgen, der ein abgetastetes Sinussignal enthält. Die Frequenz dieses abgetasteten periodischen Zeitsignals soll so gewählt werden, dass sich der Datensatz der Länge N durch Aneinanderreihung auch noch periodisch ohne Sprungstelle (Unstetigkeitsstelle) fortsetzen lässt. Der Datensatz erfüllt damit alle in 1.1 genannten Voraussetzungen und kann in MATLAB wie folgt erzeugt werden:

1. Beispieluntersuchung:

```
fa = 8000;  % Abtastfrequenz
fn = fa/2;  % Nyquistfrequenz
N = 1024;  % gewünschte FFT-Länge (N=2^x, sonst wird der DFT-Algorithmus verwendet!)
df = fa/N;  % Frequenzauflösung

% Erzeugung eines Datensatzes mit N Abtastwerten
% ----------------------------------------------
t = 0 : 1/fa : (N-1)/fa;  % x-Vektor

% Frequenzvorgabe in Hz als ganzzahlig Vielfaches der Frequenzauflösung der DFT/FFT:
f1 = df*100;  % bei fa = 8000 Hz und N = 1024 beträgt df = 7,8125 Hz und
              % f1 damit 781,25 Hz

a1 = 1;  % Amplitudenvorgabe

y = a1*sin(2*pi*f1*t);  % y-Vektor
```

```
% Graphische Darstellung
% ----------------------
% max. Amplitude zur Skalierung der graphischen Darstellung feststellen:
max_y = max(abs(y))*1.1;

fig = figure(1);
plot(y)
axis([0 N -max_y max_y])
title('Datensatz')
ylabel('Amplitude')
xlabel('N Stützstellen')
grid
```

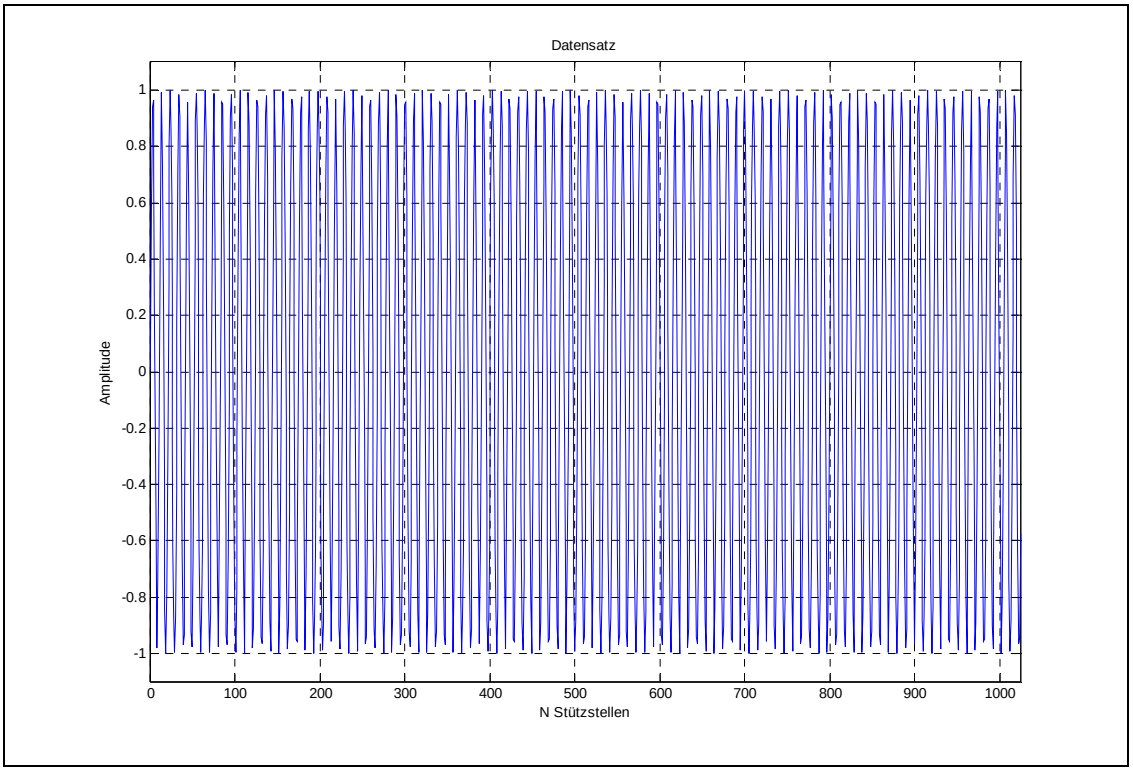

Abbildung 3.1: Datensatz (Ausschnitt der Länge N eines periodischen Zeitsignals)

3.2.2 Berechnung der FFT und Darstellung des Amplitudengangs

Die Berechnung des komplexen Spektrums erfolgt mit der bereits in Kapitel 3.1 vorgestellten MATLAB-FFT-Funktion. Die graphische Darstellung des Amplitudengangs erfolgt zunächst über alle berechneten Werte, also von $-f_n$ bis $+f_n$ jeweils linear und logarithmisch und danach in der bekannten, zumeist einzig interessierenden Form von 0 bis $+f_n$ auch jeweils linear und logarithmisch. (Anmerkung: Bei geradem N erhält man Werte bis $+f_n$ - Δf.)

Fortsetzung der 1. Beispieluntersuchung:

```
% Berechnung der FFT
% ------------------
H = fft(y, N);
% Berechnung des Amplitudengangs aus dem komplexen Frequenzvektor H:
amplH = abs(H);
% Amplitudenskalierung (Normierung auf N) und verschieben der Elemente des
% Amplitudenvektors, so dass die Darstellung des Amplitudengangs von -fn...0...fn
% erfolgen kann:
amplitudengang = fftshift(amplH/N);

% Graphische Darstellung
% ----------------------
% Frequenzvektoren (werden bei der graphischen Darstellung benötigt):
x_fn = 0 : df : fn-df;
x_fa = 0 : df : fa-df;
% max. Amplitude zur Skalierung der graphischen Darstellung feststellen:
%a = max([a1, a2, a3, a4, a5]);  % wird später benötigt
a = a1;

fig = figure(fig+1);
stem(x_fa-fn, amplitudengang, 'b.-')
axis([-fn fn 0 a/2*1.1])
title('Amplitudengang')
ylabel('Amplitude')
xlabel(['Auflösung: ',num2str(df),' Hz          Frequenz in Hz'])
grid
```

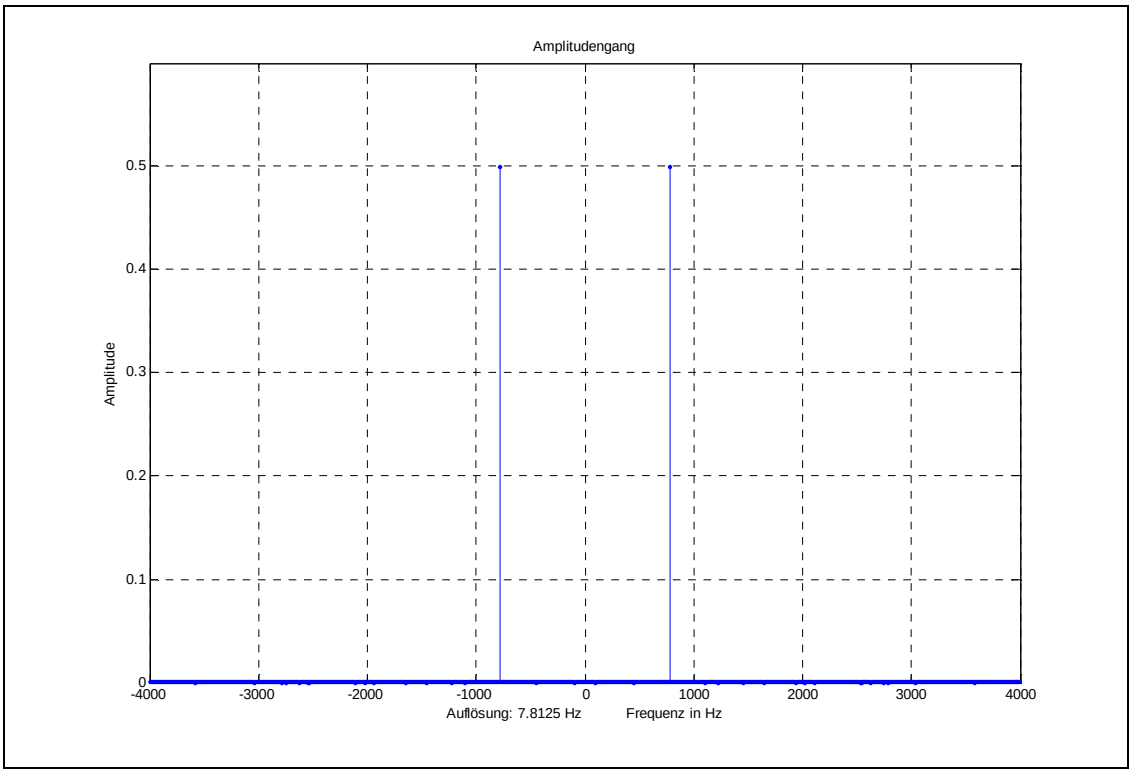

Abbildung 3.2: Amplitudengang (linear)

Fortsetzung der 1. Beispieluntersuchung:

```
fig = figure(fig+1);
plot(x_fa-fn, 20*log10(amplitudengang))
%axis([-fn fn -100 20*log10(a/2)+3])
axis([-fn fn -100 3])
title('Amplitudengang')
ylabel('Amplitude in dB')
xlabel(['Auflösung: ',num2str(df),' Hz          Frequenz in Hz'])
grid
```

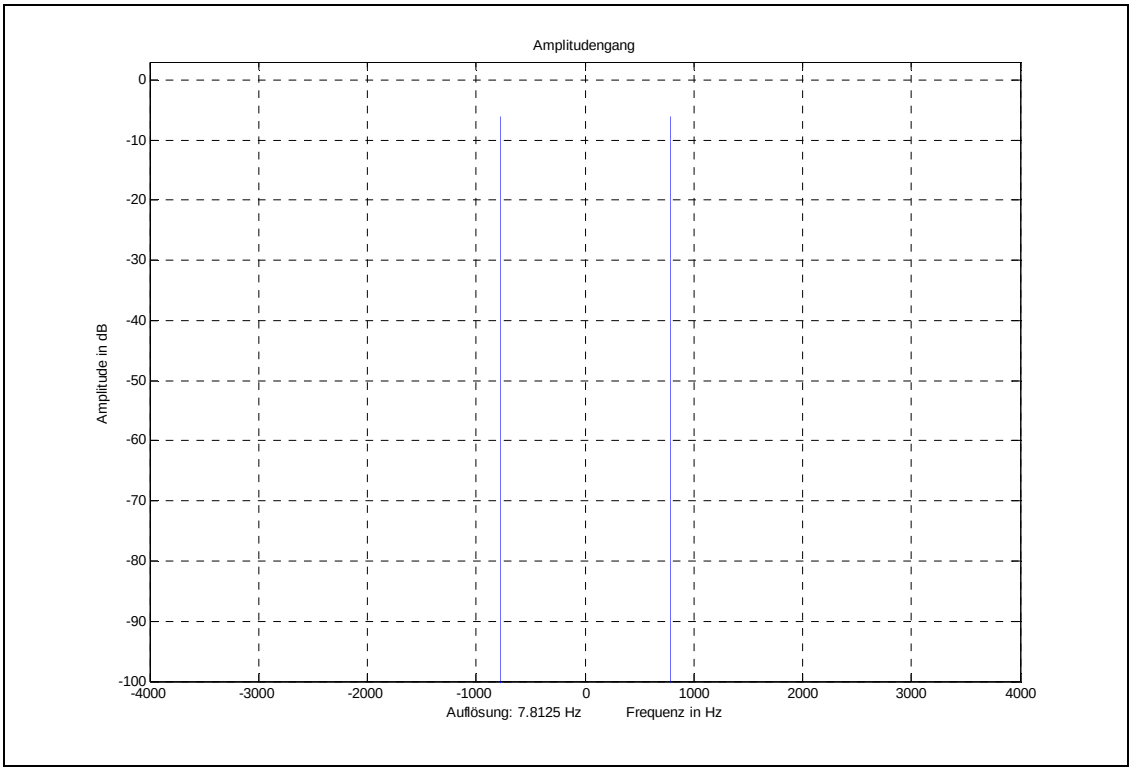

Abbildung 3.3: Amplitudengang (logarithmisch)

Fortsetzung der 1. Beispieluntersuchung:

```
% Darstellung des interessierenden Frequenzbereichs des
% Amplitudengangs (0...fn) und
% daran angepasste Amplitudenskalierung (Normierung auf N/2):
amplitudengang = [amplH(1)/N amplH(2:N/2)/(N/2)];  % DC-Bin auf N normieren!

fig = figure(fig+1);
stem(x_fn, amplitudengang, 'b.-')
axis([0 fn 0 a*1.1])
title('Amplitudengang')
ylabel('Amplitude')
xlabel(['Auflösung: ',num2str(df),' Hz          Frequenz in Hz'])
grid
```

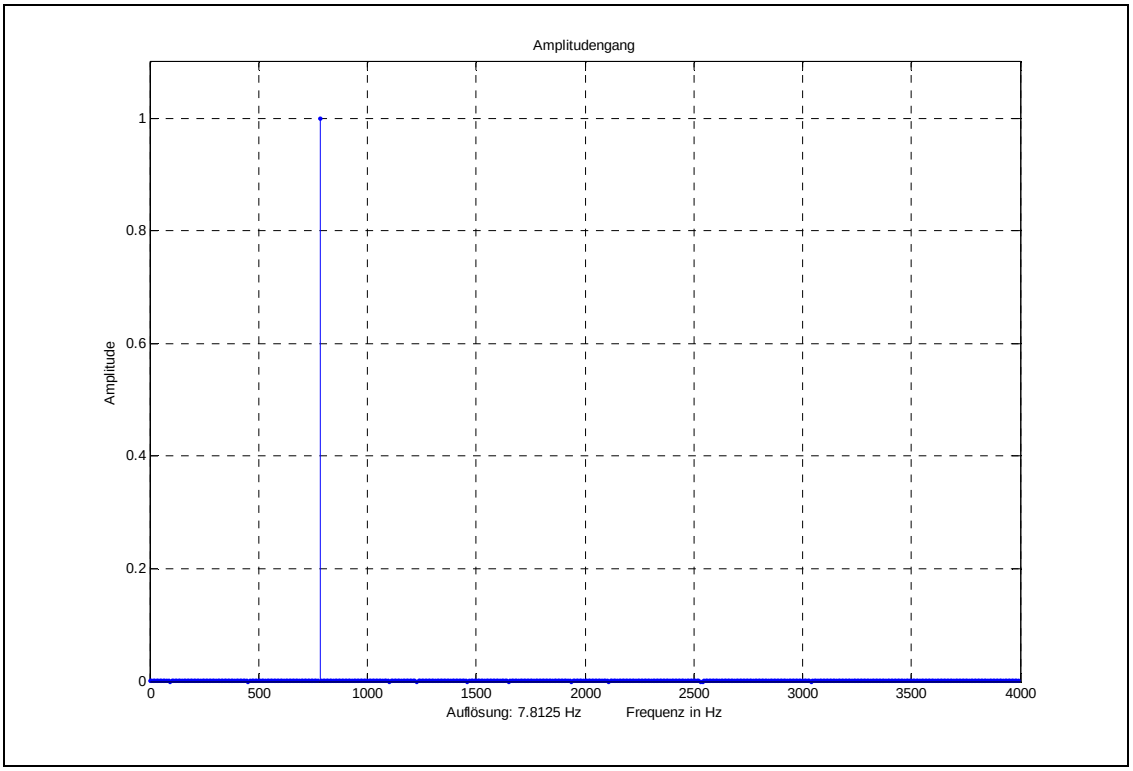

Abbildung 3.4: Amplitudengang (linear)

Fortsetzung der 1. Beispieluntersuchung:

```
fig = figure(fig+1);
plot(x_fn, 20*log10(amplitudengang))
axis([0 fn -100 20*log10(a)+3])
title('Amplitudengang')
ylabel('Amplitude in dB')
xlabel(['Auflösung: ',num2str(df),' Hz          Frequenz in Hz'])
grid
```

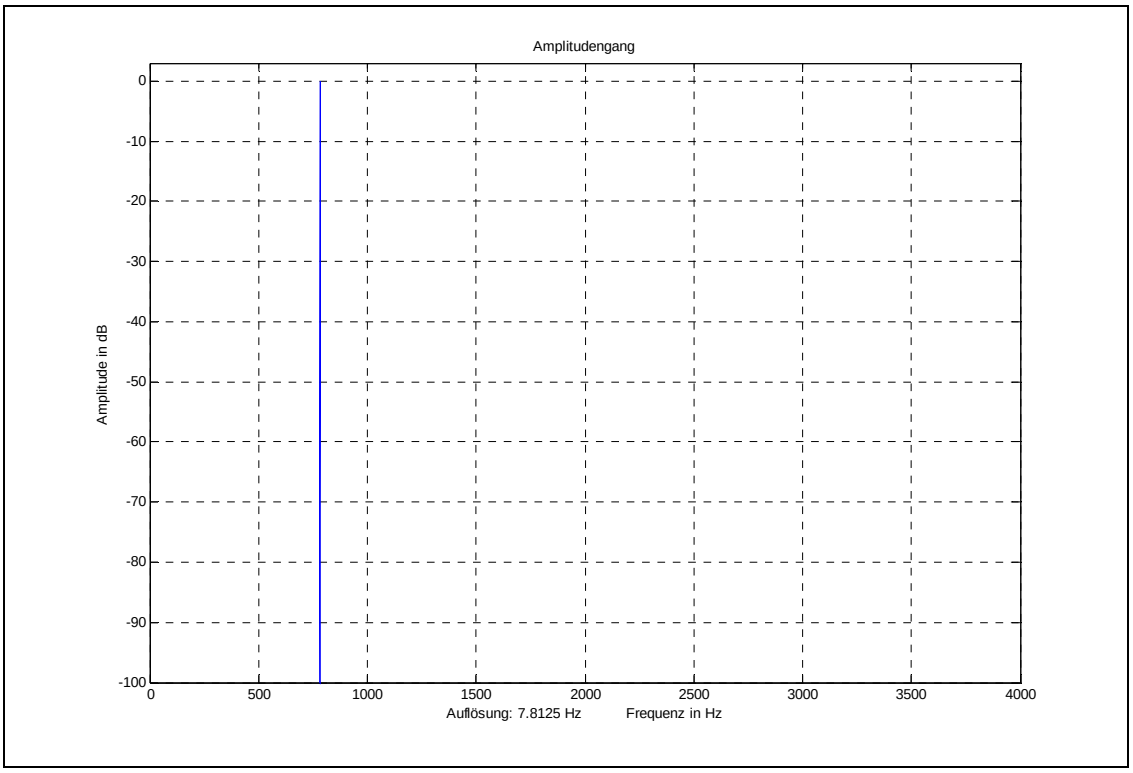

Abbildung 3.5: Amplitudengang (logarithmisch)

3.3 Zweite Beispieluntersuchung

3.3.1 Signalgenerierung

Die zweite Beispieluntersuchung soll die Auswirkungen aufzeigen, die sich ergeben, wenn die Frequenz des abgetasteten periodischen Zeitsignals so gewählt ist, dass sich der Datensatz durch Aneinanderreihung nicht periodisch ohne Sprungstelle (Unstetigkeitsstelle) fortsetzen lässt. Dazu ist nur eine geringfügige Änderung der Frequenzvorgabe notwendig.

2. Beispieluntersuchung:

```
% Einzige Änderung in der bereits erfolgten Untersuchung
% ------------------------------------------------------

% Frequenzvorgabe in Hz als ganzzahlig Vielfaches der Frequenzauflösung der DFT/FFT:
%f1 = df*100;  % bei fa = 8000 Hz und N = 1024 beträgt df = 7,8125 Hz und
               % f1 damit 781,25 Hz

% Frequenzvorgabe in Hz als nicht ganzzahlig Vielfaches der
% Frequenzauflösung der DFT/FFT:
f1 = 784;
```

3.3.2 Berechnung der FFT und Darstellung des Amplitudengangs

Die Berechnung des komplexen Spektrums und die graphische Darstellung des Amplitudengangs erfolgt genau wie in der ersten Beispieluntersuchung.
Vergleichen Sie die folgenden Darstellungen jeweils mit denen der ersten Beispieluntersuchung.

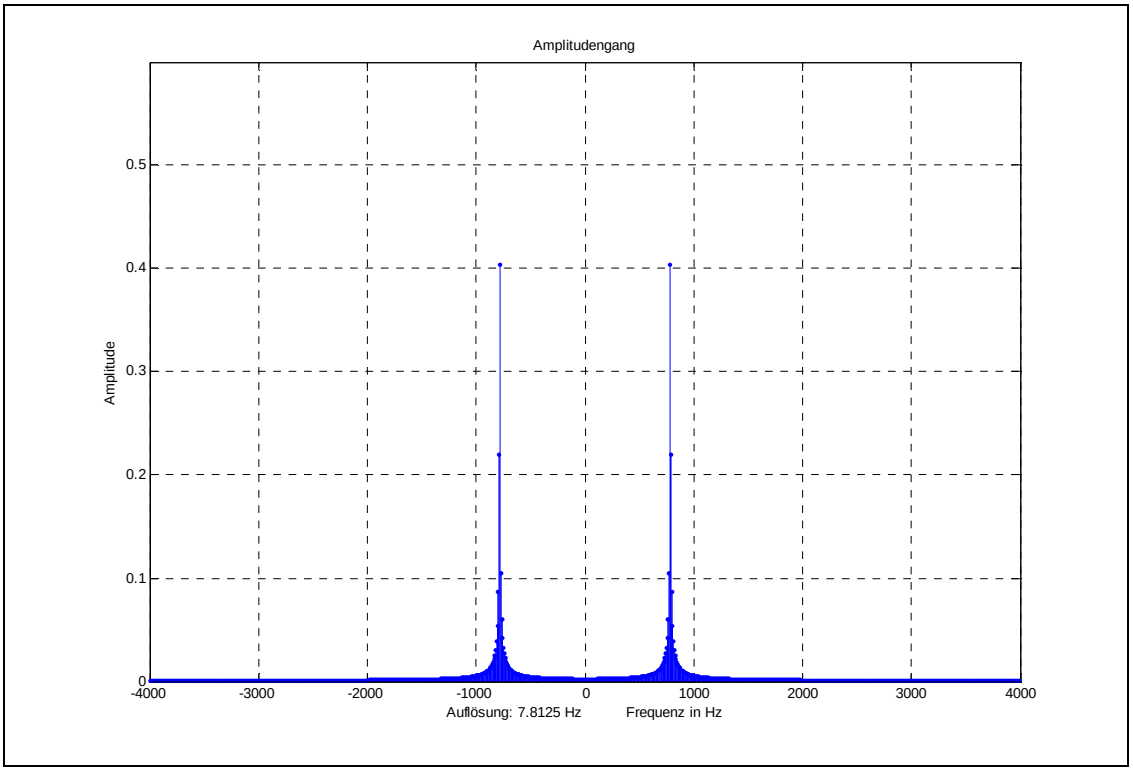

Abbildung 3.6: Amplitudengang (linear)

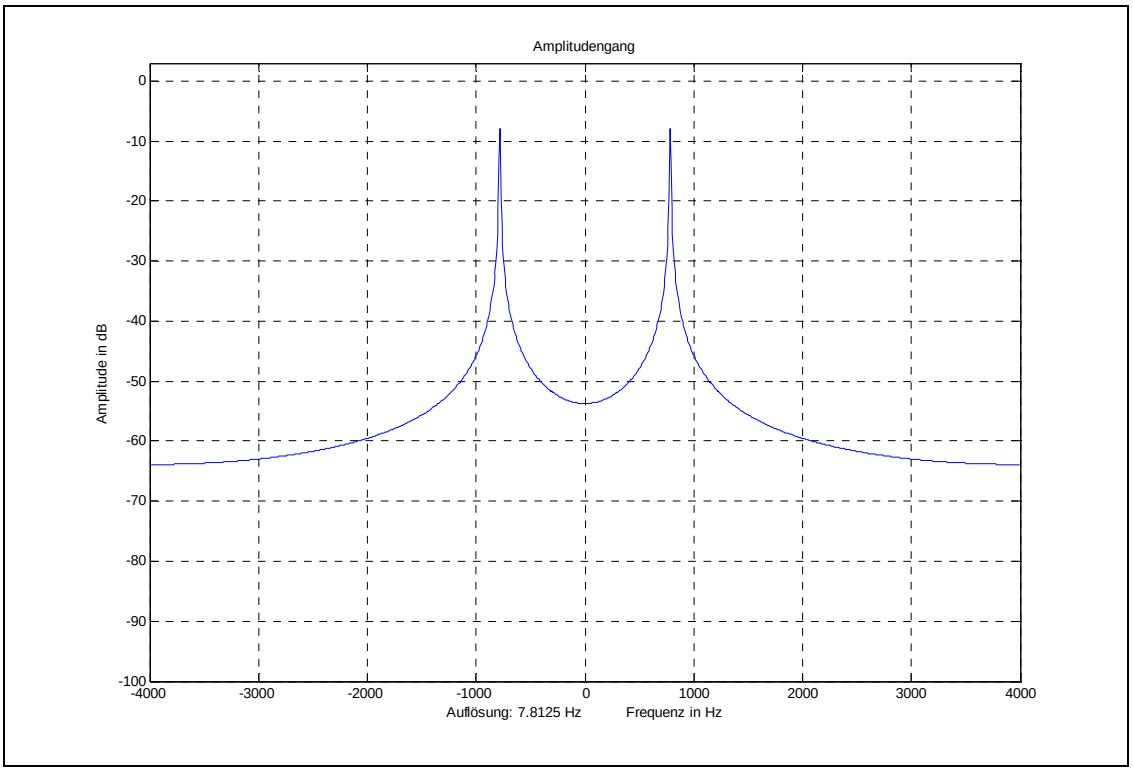

Abbildung 3.7: Amplitudengang (logarithmisch)

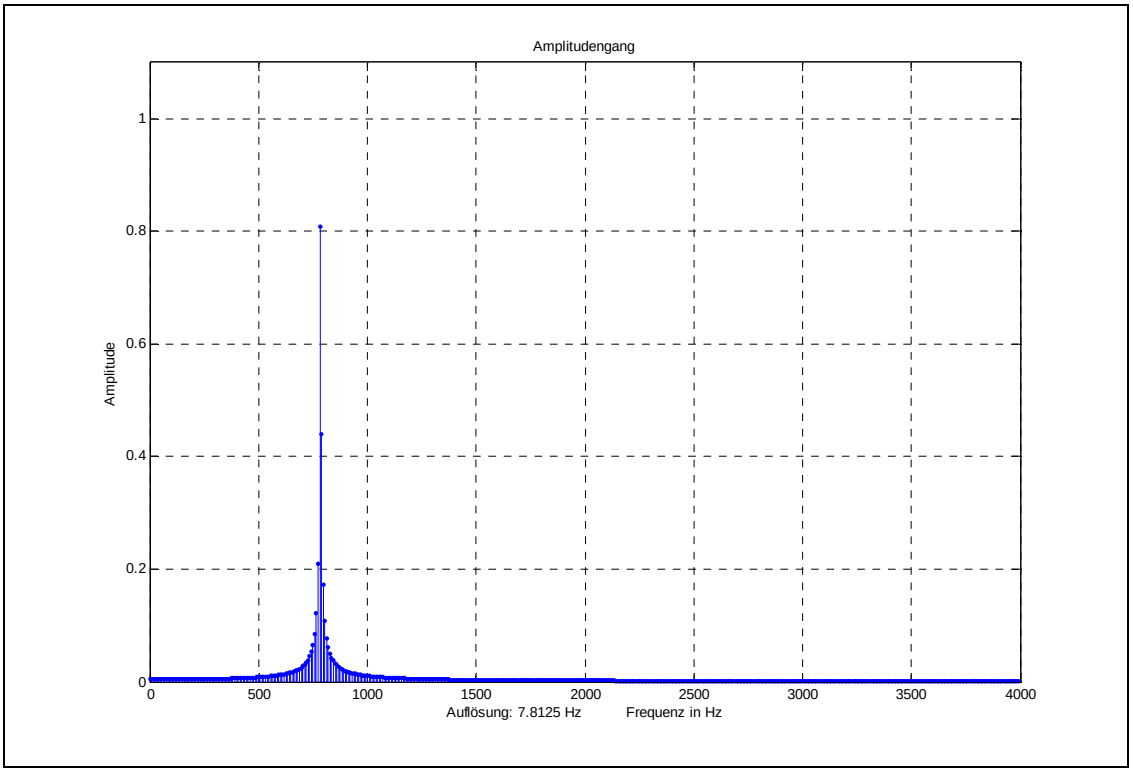

Abbildung 3.8: Amplitudengang (linear)

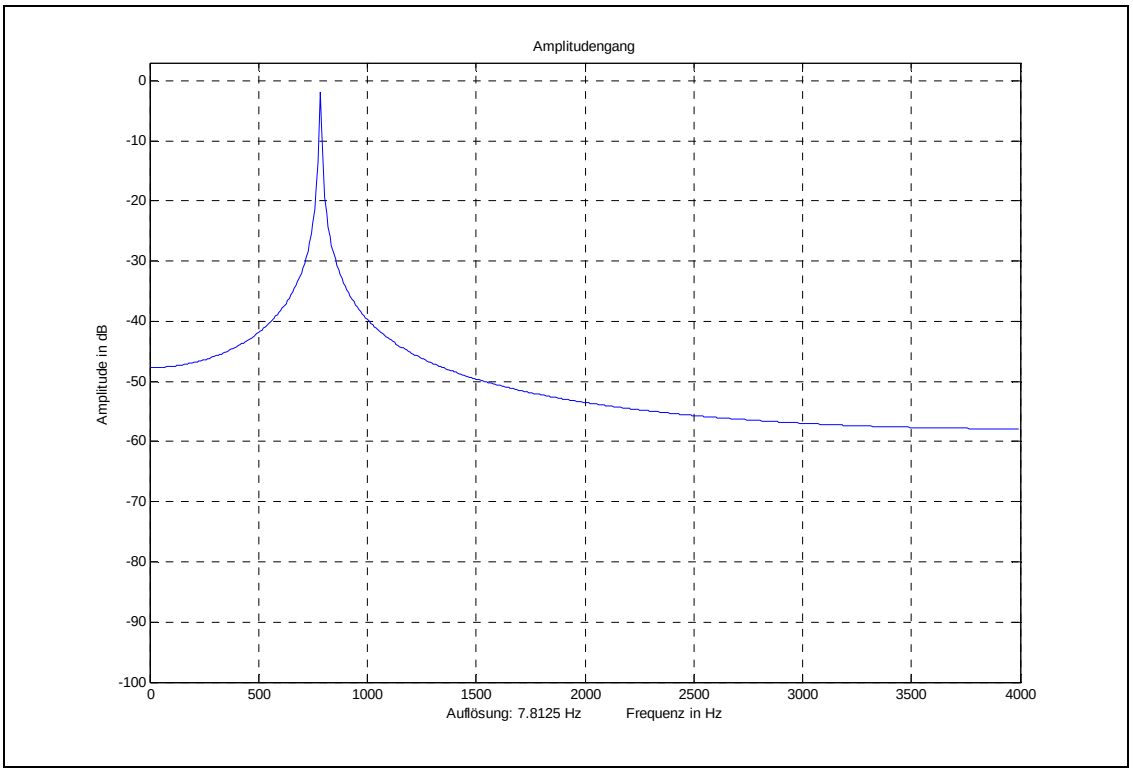

Abbildung 3.9: Amplitudengang (logarithmisch)

3.4 Fensterfunktionen

Um den in der vorangegangenen Untersuchung deutlich gewordenen Leakage-Effekt zu verringern, kann der Datensatz vor der Transformation mit einer Fensterfunktion multipliziert werden (siehe Kapitel 1.2.3).

In MATLAB stehen dafür eine Reihe von Fensterfunktionen zur Verfügung:

```
bartlett - Bartlett window
barthannwin - Modified Bartlett-Hanning window
blackman - Blackman window
blackmanharris - Minimum 4-term Blackman-Harris window
bohmanwin - Bohman window
chebwin - Chebyshev window
flattopwin - Flat Top window (vgl. FFT-Function Tektronix Oscilloscope TDS 210)
gausswin - Gaussian window
hamming - Hamming window
hann - Hann window (vgl. FFT-Function Tektronix Oscilloscope TDS 210)
kaiser - Kaiser window
nuttallwin - Nuttall defined minimum 4-term Blackm.-Harris window
parzenwin - Parzen (de la Valle-Poussin) window
rectwin - Rectangular window
tukeywin - Tukey window
triang - Triangular window
```

Die folgende Abbildung zeigt vier solcher Fenster, nämlich das Hamming-, Hann-, Blackman- und Bartlett-Window.

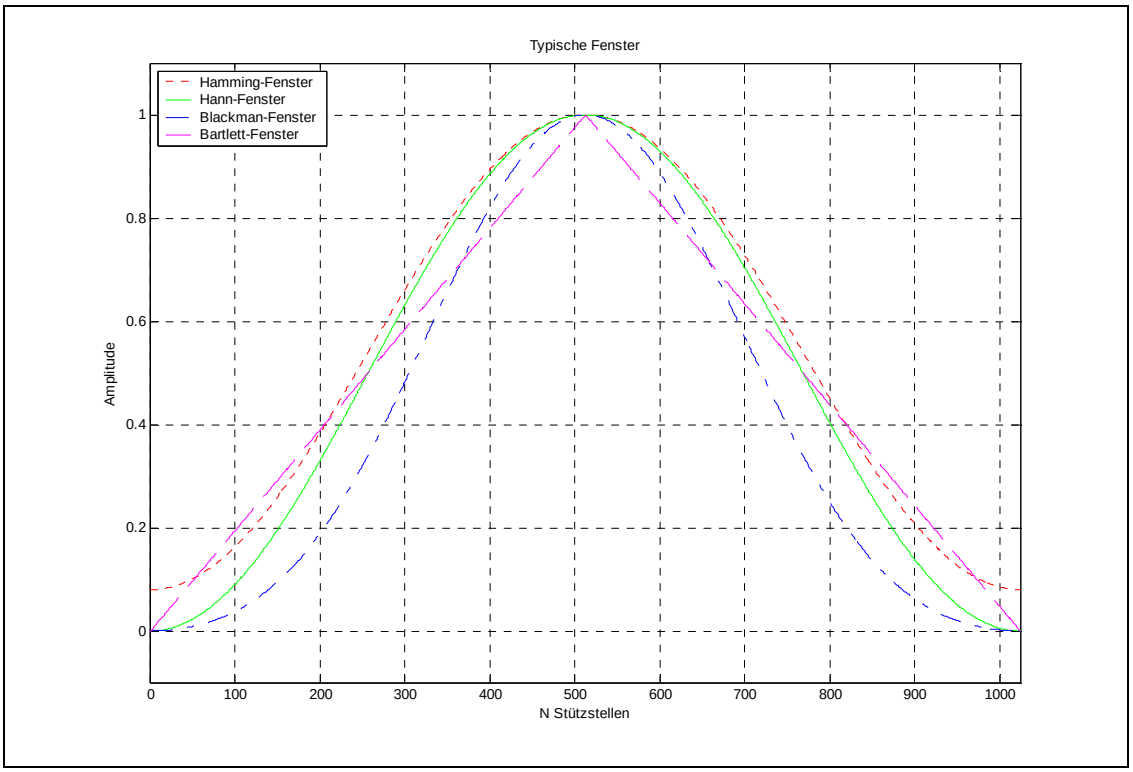

Abbildung 3.10: Typische Fenster

Erläuterung:

Da die Multiplikation im Zeitbereich (Fensterung) bekanntlich einer Faltung im Frequenzbereich entspricht, kann das Ergebnis der DFT auch als aus der Faltung des Zeitsignalspektrums *H(f)* mit dem Spektrum der Fensterfunktion und anschließender Abtastung im Raster Δf entstanden aufgefasst werden. Die Fourier-Transformierte der Fensterfunktion hinterlässt so anschaulich ihre Spuren.

Faltung: $$G_{kont.}(f) = H(f) * WIN(f) = \int_{-\infty}^{\infty} H(\tau) \cdot WIN(f-\tau)\, d\tau \qquad (3.1)$$

Abtastung: $$G_{diskret}(f) = G_{kont.}(f) \cdot \sum_{n=0}^{N-1} \delta(f - n \cdot \Delta f) = \sum_{n=0}^{N-1} G_{kont.}(n \cdot \Delta f) \cdot \delta(f - n \cdot \Delta f) \qquad (3.2)$$

genauer betrachtet muss das zweiseitige diskrete Spektrum wie folgt zusammengesetzt werden:

$$G_{diskret}(f) = \sum_{n=0}^{\frac{N}{2}} G_{kont.}(n \cdot \Delta f) \cdot \delta(f - n \cdot \Delta f) + \sum_{n=-\frac{N}{2}+1}^{-1} G_{kont.}(n \cdot \Delta f) \cdot \delta(f - n \cdot \Delta f) \qquad (3.3)$$

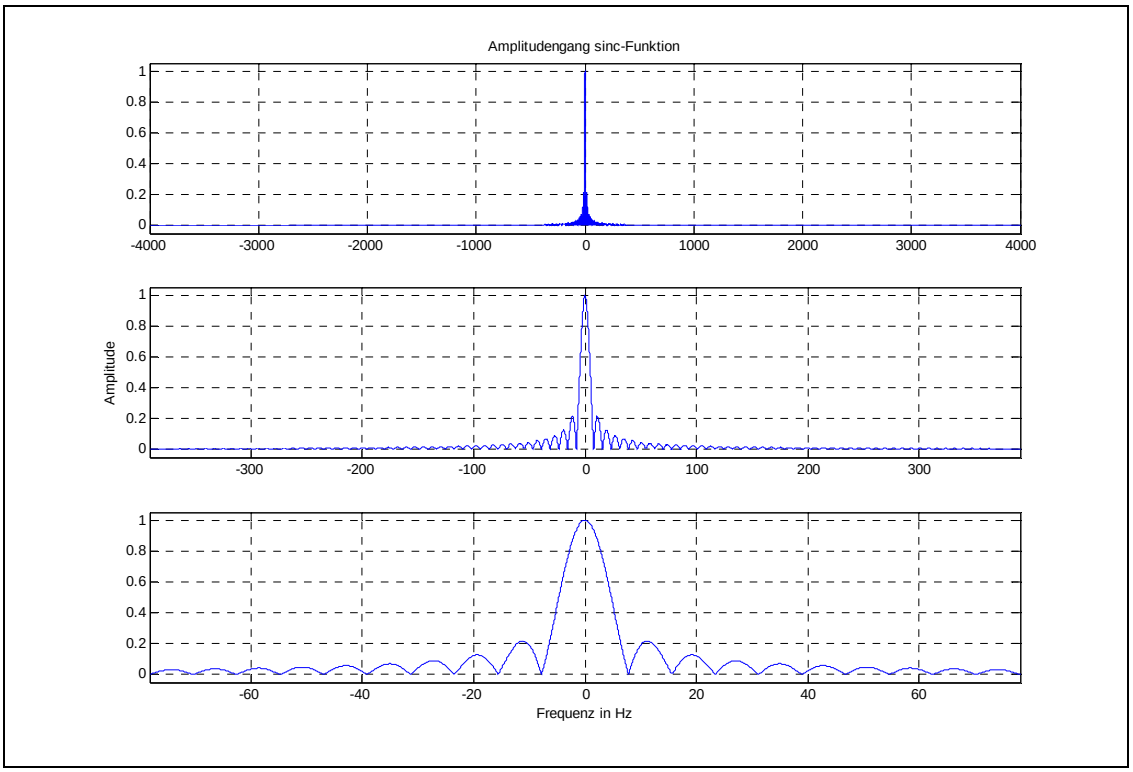

Abbildung 3.11: Amplitudengang der Fourier-Transformierten des Rechteckfensters (linear)

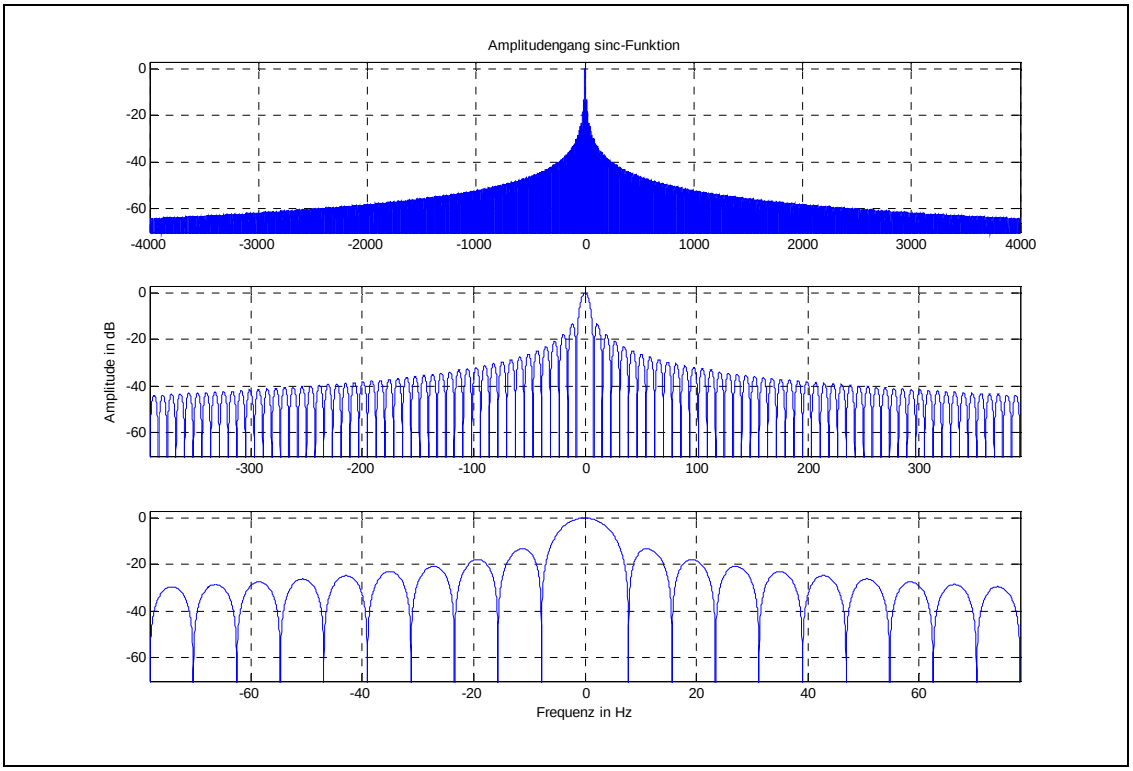

Abbildung 3.12: Amplitudengang der Fourier-Transformierten des Rechteckfensters (logarithmisch)

Damit lässt sich auch eine andere Sicht auf den Leck-Effekt angeben:

Als Zeitsignal soll zunächst eine Sinusschwingung der Frequenz f_{sin}, genau wie in der ersten Beispieluntersuchung, angenommen werden. Sie erfüllt also die in Kapitel 1.1 genannten Voraussetzungen.
Das Ausschneiden des Datensatzes der Länge N aus dem abgetasteten Zeitsignal entspricht der Fensterung des abgetasteten Zeitsignals mit einem Rechteckfenster der Breite N.
Da die Fourier-Transformierte dieses Rechteckfensters eine $si(f)$-Funktion ist (auch $sinc(f) = sin(f)/f$), die bei Vielfachen von Δf Nullstellen hat (außer beim Hauptzipfel), liefert die der Faltung der $si(f)$-Funktion mit der Fourier-Transformierten $H(f)$ der Sinusschwingung folgende Abtastung im Raster Δf auch nur einen diskreten Wert an der Stelle f_{sin}.
(Betrachtet man das Faltungsergebnis nur bei den diskreten Frequenzen im Raster Δf, dann fällt hier die Frequenzlinie von $H(f)$ an der Stelle f_{sin} – außer beim Hauptzipfel (main lobe) – mit den Nullstellen der $si(f)$-Funktion zusammen, weil die Frequenz der Sinusschwingung ein ganzzahlig Vielfaches der Frequenzauflösung der N-Punkte-DFT ist: $f_{sin} = n \cdot \Delta f$)

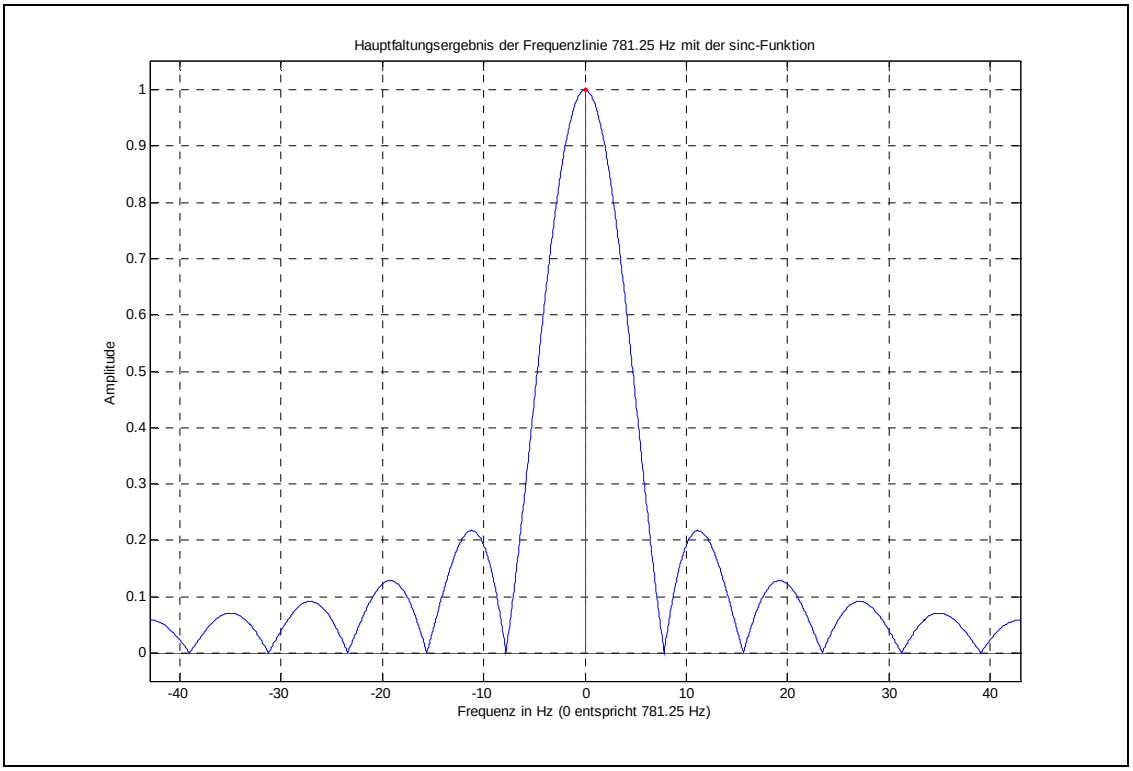

Abbildung 3.13: Veranschaulichung des wesentlichen Faltungsergebnisses

Hingegen liefert die Kombination aus der Faltung und der anschließenden Abtastung der $si(f)$-Funktion mit der Fourier-Transformierten einer Sinusschwingung, die nicht die in Kapitel 1.1 genannten Voraussetzungen erfüllt, überall Werte ungleich Null.
Die Frequenzlinie von $H(f)$ an der Stelle f_{sin} – und damit auch das Maximum des Faltungsergebnisses – liegt neben dem bei der anschließenden Abtastung benutzten Raster. Die maximale Amplitude wird nicht erreicht, sondern es ergeben sich zwei Frequenzmaxima mit geringerer, möglicherweise gleicher Amplitude (vlg. Abbildung 3.14).

Damit liegen auch die Nullstellen der $si(f)$-Funktion – also die minimalen Faltungsergebnisse – neben dem Raster. Man erhält also überall Werte ungleich Null (man beachte die doch recht geringe Dämpfung der Nebenkeulen (side lobes) der $si(f)$-Funktion im Vergleich zur Hauptkeule in Abbildung 3.12 !).

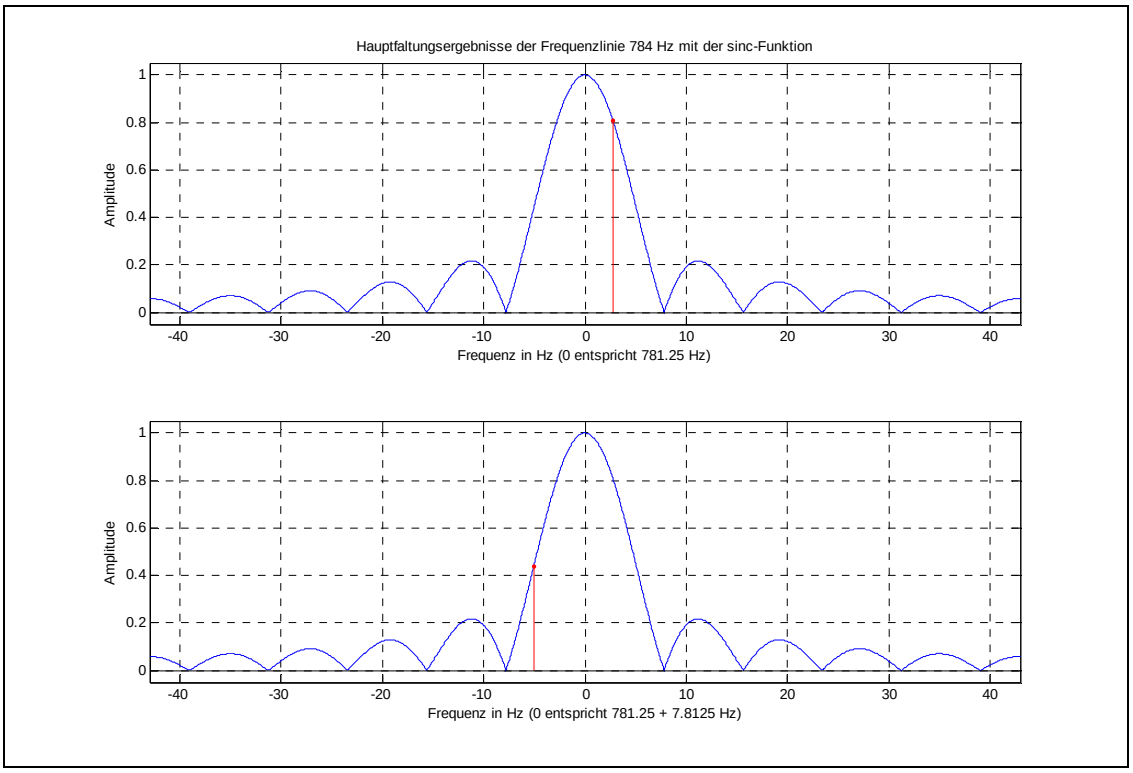

Abbildung 3.14: Veranschaulichung der beiden wesentlichen Faltungsergebnisse, die man nach der Abtastung im Raster Δf erhält

Das Zerlaufen des wahren Spektrums über sämtliche Indexwerte lässt sich über die Faltung der Fourier-Transformierten von Zeitsignal und Fensterfunktion und erst anschließend erfolgender Abtastung also leicht veranschaulichen (siehe dazu auch Literaturangabe [1] in Kapitel 2).

3.5 Dritte Beispieluntersuchung

3.5.1 Fensterung des Datensatzes

Im Folgenden soll die zweite Beispieluntersuchung wiederholt werden, diesmal aber mit Fensterung des Datensatzes vor der FFT-Berechnung mit dem (periodischen) Hann-Fenster. (Die angegebenen MATLAB-Befehle können einfach an die bereits bestehenden angehängt werden.)

3. Beispieluntersuchung:

```
% Anhang an die bereits erfolgte Untersuchung
% ----------------------------------------

win = hann(N, 'periodic')';
%y_win = y.*win;  % Fensterung ohne Amplitudenkorrektur
```

```
y_win = y.*win*N/sum(win);  % Fensterung mit Amplitudenkorrektur

max_y = max(abs(y_win))*1.1;

fig = figure(fig+1);
plot(y_win)
axis([0 N -max_y max_y])
title('Datensatz nach Fensterung mit Hann-Fenster')
ylabel('Amplitude')
xlabel('N Stützstellen')
grid
```

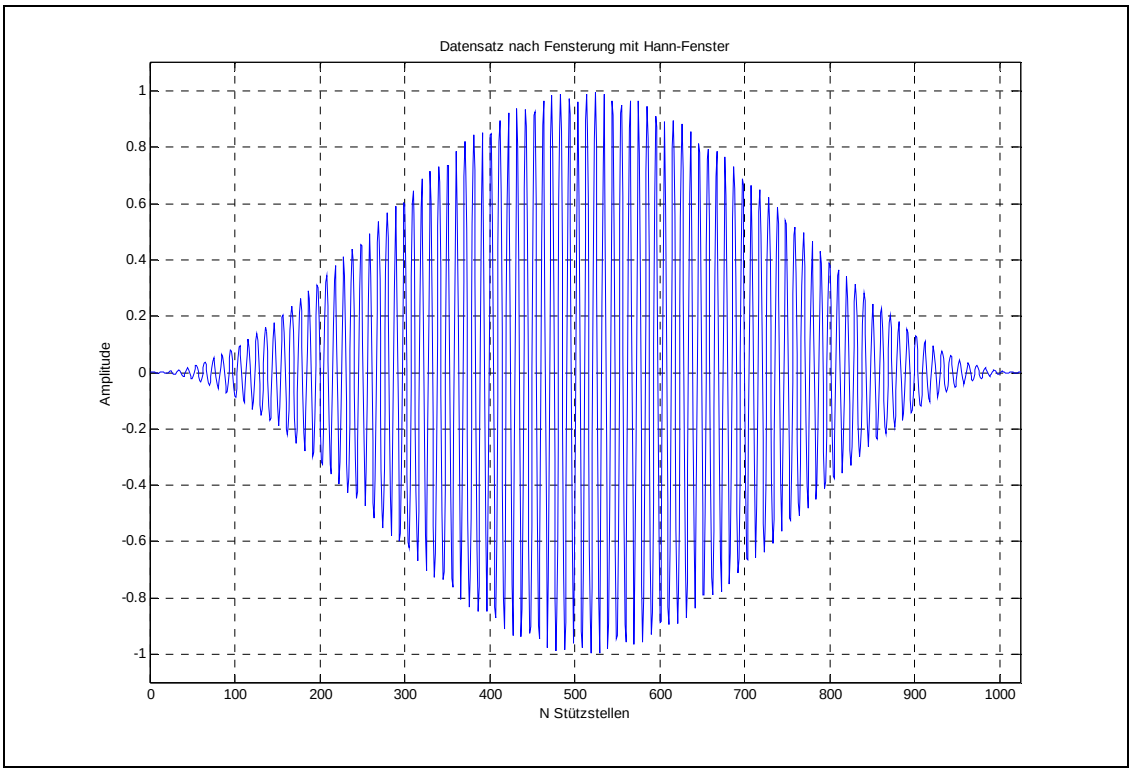

Abbildung 3.15: Datensatz der Länge N nach Fensterung (ohne Amplitudenkorrektur)

3.5.2 Berechnung der FFT und Darstellung des Amplitudengangs

Die Berechnung des komplexen Spektrums und die graphische Darstellung des Amplitudengangs erfolgt analog zu den vorangegangenen Beispieluntersuchungen.
Vergleichen Sie die sich ergebenden Darstellungen jeweils mit denen der ersten beiden Beispieluntersuchungen (insbesondere hinsichtlich der Frequenzverschmierung und der Amplitude).

Damit Sie sich das zustande kommende Ergebnis auch über die Faltung der Fourier-Transformierten von Zeitsignal und Fensterfunktion erklären können, ist in Abbildung 3.16 zunächst der Betrag der Fourier-Transformierten des Hann-Fensters dargestellt. Zu beachten ist dabei insbesondere der im Vergleich zum Betrag der Fourier-Transformierten des Rechteckfensters (Abbildung 3.17) doppelt so breite Main-Lobe und die geringere Amplitude der Side-Lobes.

Um den interessanten Bereich der Amplitudengänge um den Hauptlappen herum gut erkennen zu können, wurde nur ein sehr enger Frequenzbereich hoch aufgelöst dargestellt.

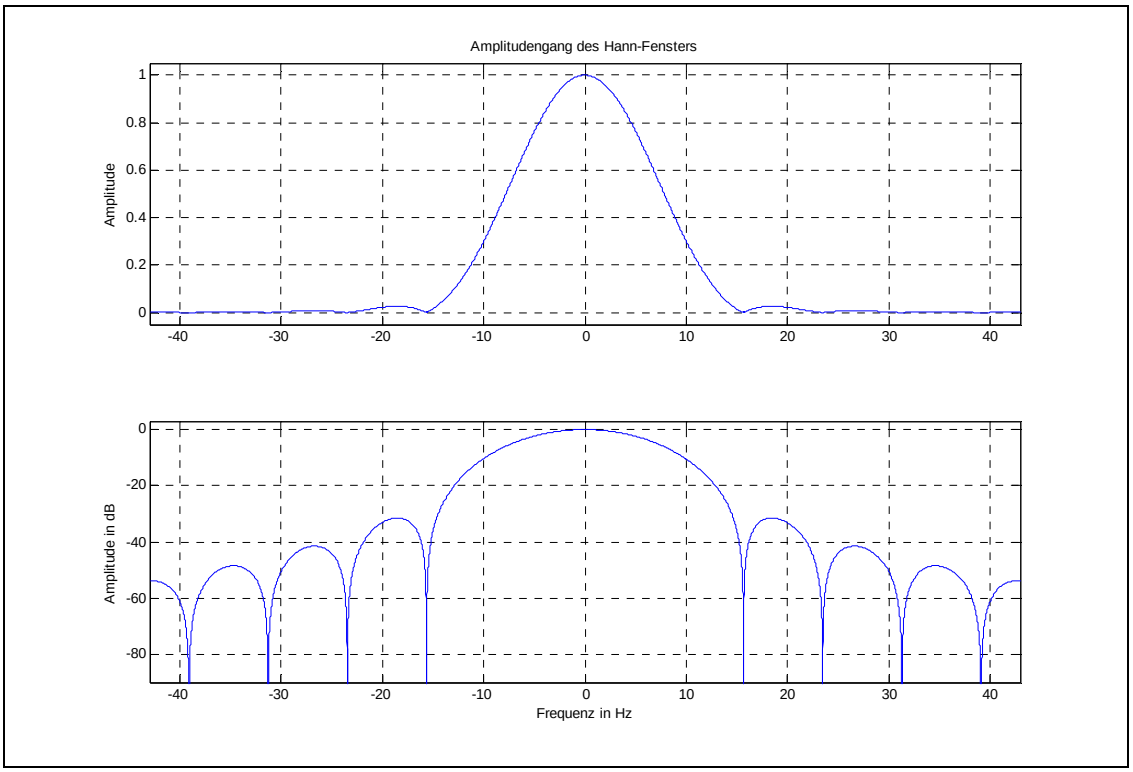

Abbildung 3.16: Amplitudengang der Fourier-Transformierten des Hann-Fensters

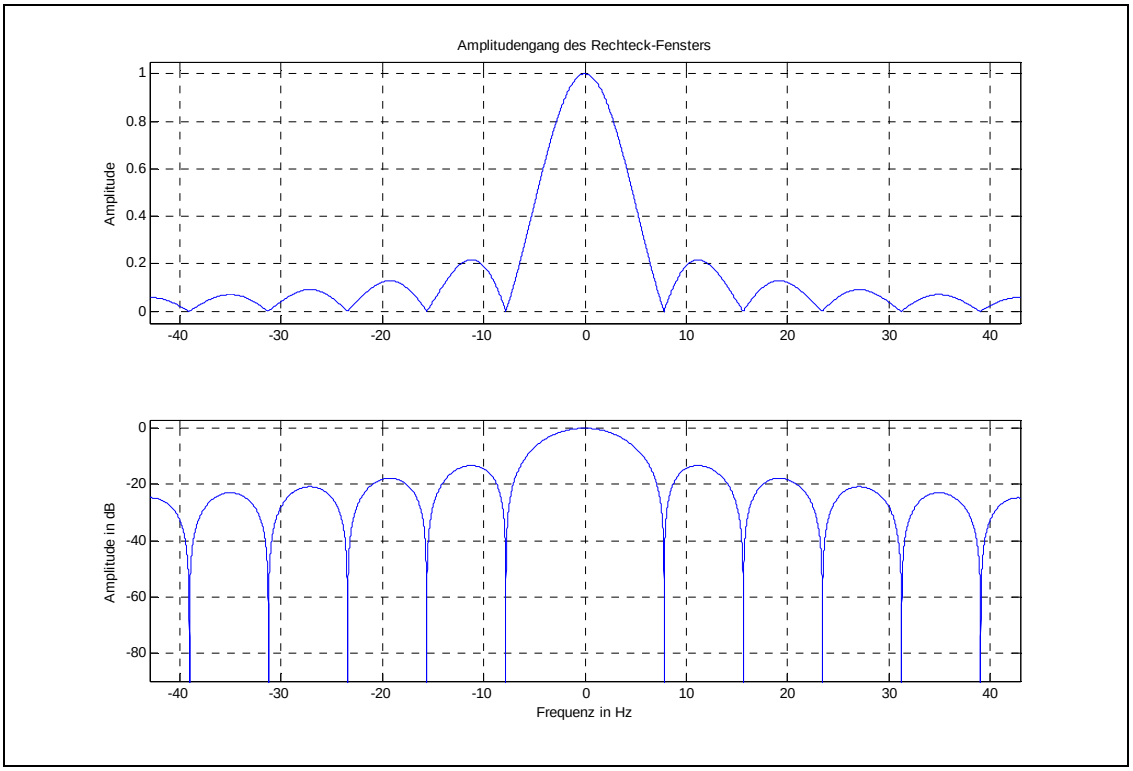

Abbildung 3.17: Amplitudengang der Fourier-Transformierten des Rechteckfensters

Fortsetzung der 3. Beispieluntersuchung:

```
% Berechnung der FFT
% ------------------

H = fft(y_win, N);

% Berechnung des Amplitudengangs aus dem komplexen Frequenzvektor H:
amplH = abs(H);

% Amplitudenskalierung (Normierung auf N) und verschieben der Elemente des
% Amplitudenvektors, so dass die Darstellung des Amplitudengangs von -fn...0...fn
% erfolgen kann:
amplitudengang = fftshift(amplH/N);

% Graphische Darstellung
% ----------------------

fig = figure(fig+1);
stem(x_fa-fn, amplitudengang, 'b.-')
axis([-fn fn 0 a/2*1.1])
title('Amplitudengang nach Fensterung')
ylabel('Amplitude')
xlabel(['Auflösung: ',num2str(df),' Hz          Frequenz in Hz'])
grid
```

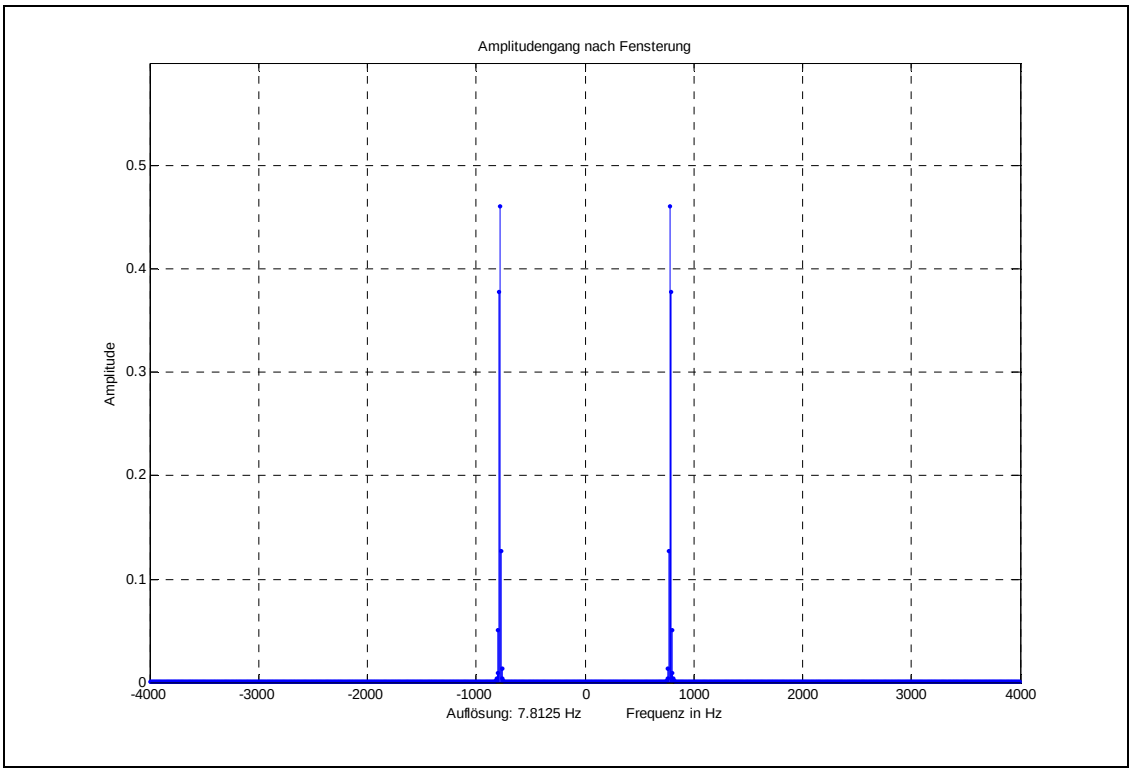

Abbildung 3.18: Amplitudengang (linear)

Fortsetzung der 3. Beispieluntersuchung:

```
fig = figure(fig+1);
plot(x_fa-fn, 20*log10(amplitudengang))
%axis([-fn fn -100 20*log10(a/2)+3])
axis([-fn fn -100 3])
title('Amplitudengang nach Fensterung')
ylabel('Amplitude in dB')
xlabel(['Auflösung: ',num2str(df),' Hz          Frequenz in Hz'])
grid
```

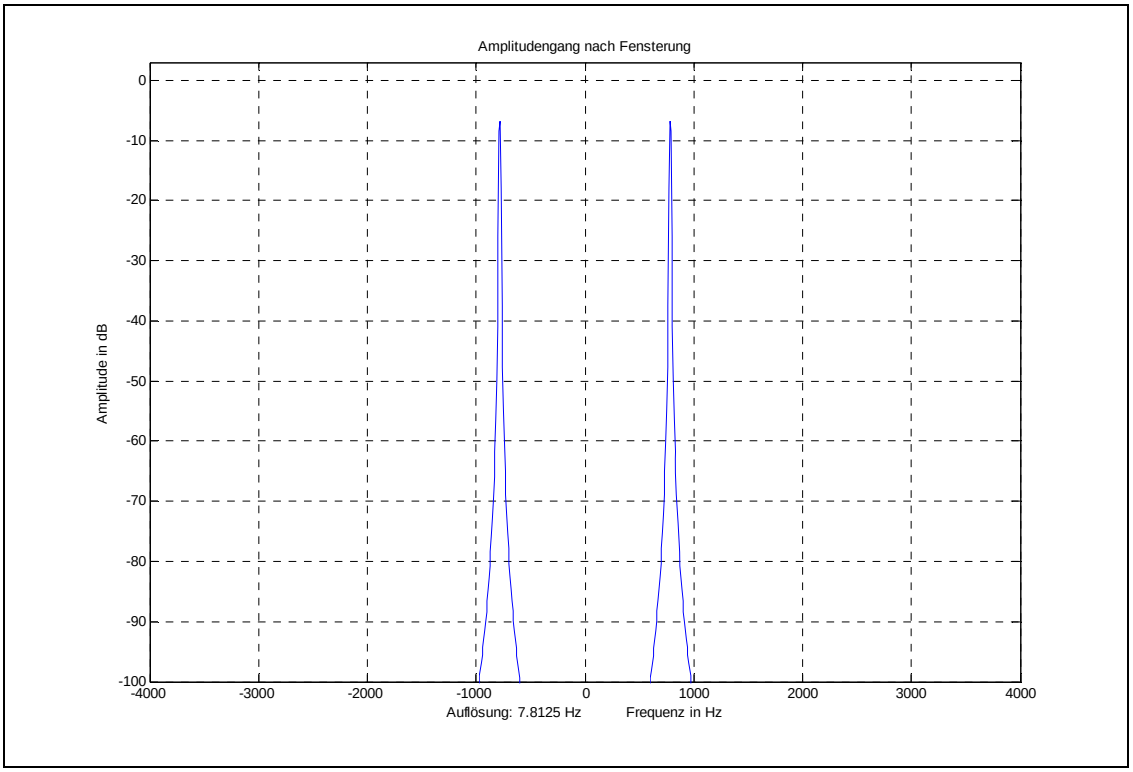

Abbildung 3.19: Amplitudengang (logarithmisch)

3.6 Vorbereitung, Durchführung, Aufgaben und Fragen

3.1 Vorbereitung:

Übernehmen Sie die in den Beispieluntersuchungen angegebenen MATLAB-Befehle in ein m-file und vollziehen Sie die Untersuchungen nach.
Sofern beim Lesen noch nicht geschehen, sollten Sie sich dabei auch in die Funktionsweise des Matlab-Scripts einarbeiten, d.h. den Sinn der einzelnen Befehle, deren Ergebnisse und das Zusammenwirken erfassen und verstehen.

Vergleichen Sie den Amplitudengang aus Abbildung 3.8 mit der Darstellung der Faltungsergebnisse in Abbildung 3.14, indem Sie Ihre erzeugte MATLAB-Grafik entsprechend vergrößern (zoomen mit der Maus).
Testen Sie einige verschiedene Frequenzvorgaben bei der Datensatzerzeugung und auch die Wirkung anderer Fensterfunktionen.

3.2 Testen Sie die Auswirkungen der Fensterung mit z.B. dem Hann-Fenster bei Vorgabe eines Datensatzes, der die in Kapitel 1.1 genannten Voraussetzungen einhält. Was fällt Ihnen dabei auf? Erklären Sie Ihre Beobachtungen über die Faltung.

3.3 Ist eine Fensterung des Datensatzes in jedem Fall sinnvoll? Begründen Sie Ihre Antwort.

3.4 Durchführung:

Führen Sie eine Untersuchung mit dem wie folgt erzeugten Datensatz durch:

```
% Frequenzvorgabe in Hz als ganzzahlig Vielfaches der
% Frequenzauflösung der DFT/FFT:
f1 = df*100;  % bei fa = 8000 Hz und N = 1024 beträgt df = 7,8125 Hz und
              % f1 damit 781,25 Hz

% Frequenzvorgabe in Hz als nicht ganzzahlig Vielfaches der
% Frequenzauflösung der DFT/FFT:
%f1 = 784;

f2 = f1*2;
f3 = f1*3;
f4 = f1*4;
f5 = f1*5;

a1 = 1;  % Amplitudenvorgaben
a2 = 1;
a3 = 1;
a4 = 1;
a5 = 1;

y = a1*sin(2*pi*f1*t) ...
  + a2*sin(2*pi*f2*t) ...
  + a3*sin(2*pi*f3*t) ...
  + a4*sin(2*pi*f4*t) ...
  + a4*sin(2*pi*(f4+2*df)*t) ...
  + a5*sin(2*pi*f5*t);
```

Ändern Sie dazu auch die Ermittlung der maximalen Amplitude für die graphische Darstellung wie folgt ab:

```
% max. Amplitude zur Skalierung der graphischen Darstellung feststellen:
a = max([a1, a2, a3, a4, a5]);
%a = a1;
```

Vergleichen Sie bei der Untersuchung wieder die Ergebnisse bei Vorgabe der Frequenzen als ganzzahlig Vielfaches der Frequenzauflösung der N-Punkte-DFT und als nicht ganzzahlig Vielfaches (Änderung von `f1`).

Zoomen Sie jeweils interessante Bereiche der dargestellten Graphen in den Graphik-Fenstern heraus. Insbesondere ist der Bereich um die vierte Frequenzkomponente von Interesse, da direkt daneben eine weitere, um `2*df` verschobene Komponente platziert wurde.

3.5 Erzeugen Sie einige Ausdrucke Ihrer Untersuchungsergebnisse, welche die Eigenschaften und Effekte der DFT aufzeigen und kommentieren / erläutern Sie die Graphen.

3.6 Führen Sie eine Untersuchung an einer Sprachprobe durch. Das Wave-File *bbauer8.wav* kann in MATLAB wie folgt eingelesen, dargestellt und bearbeitet werden:

```
% Sprachfile einlesen:
% --------------------
wav_file = 'bbauer8';  % Dateiname
[wave_seq, fs, w] = wavread(wav_file);

% Amplitudennormierung
% Die Wave-File-Amplituden liegen zwar bereits im Bereich -1.0 ... +1.0,
```

```
% um aber garantiert Vollaussteuerung zu erreichen trotzdem normieren!
wave_seq = wave_seq / max(abs(wave_seq));

disp(' ')
disp(['File: ' wav_file '.wav'])
disp(['fs = ' num2str(fs) ' Hz'])
disp(['Lineare PCM mit ' num2str(w) ' Bit Wortbreite'])

fig = figure(fig+1);
plot(wave_seq)
axis([1 length(wave_seq) -1.1 1.1])
title('Sprachprobe')
ylabel('Amplitude')
xlabel('Samples')
grid

disp(' ')
disp(['Wiedergabe von ' wav_file ' läuft !'])
sound(wave_seq, fs, w)

pause(ceil(length(wave_seq)/fs))  % Abspieldauer abwarten

% Datensatz der Länge N ausschneiden (Erstes Wort der Sprachprobe: Ja)
start_sample = 83;
wave_seq_part = wave_seq(start_sample : start_sample+N-1);

fig = figure(fig+1);
plot(wave_seq_part)
axis([1 N -1.1 1.1])
title('Datensatz aus Sprachprobe')
ylabel('Amplitude')
xlabel('N Stützstellen')
grid
```

Aus dem Datensatz `wave_seq_part` kann dann wie gehabt das Spektrum berechnet und anschließend dargestellt werden.
Benutzen Sie für die logarithmische Darstellung des Amplitudengangs den Befehl `semilogx()`, so dass auch die Frequenzachse eine logarithmische Einteilung erhält. Die Darstellung erfolgt dann insgesamt also doppeltlogarithmisch. Stellen Sie den Amplitudengang von 10 Hz bis `fn` dar. Erzeugen Sie einen Ausdruck des Amplitudengangs in doppeltlogarithmischer Darstellung.

(Die Fragen sind handschriftlich in der Versuchsvorbereitung bzw. Ihrem Versuchsprotokoll zu beantworten!)

Optionale Zusatzaufgaben:

3.7 Wiederholen Sie die erste Beispieluntersuchung (Kapitel 3.2) mit der Frequenzvorgabe `f1 = 2*df` zunächst ohne und danach mit einer Phasenverschiebung von 90° bzw. $\pi/2$.

```
phase = pi/2;  % Phasenverschiebung
y = a1*sin(2*pi*f1*t + phase);  % y-Vektor
```

Welche Analyseergebnisse erwarten Sie?

Stimmen die tatsächlichen Ergebnisse mit Ihren Erwartungen überein?

3.8 Zur Anpassung der Datensatzlänge an eine Zweierpotenz oder zur Erhöhung der Frequenzauflösung bzw. Stützpunktdichte der DFT/FFT werden vor der Transformation i.d.R. Nullen an den Datensatz angehängt.
Untersuchen Sie die datensatzabhängigen Begleiterscheinungen dieser Vorgehensweise indem Sie die beiden Untersuchungen von 3.7 wiederholen, aber jeweils nur die erste Hälfte des Datensatzes übernehmen bzw. die zweite Hälfte auf Null setzen.

```
y = [y(1:N/2) zeros(1, N/2)];
```

Welche Analyseergebnisse erwarten Sie?

Stimmen die tatsächlichen Ergebnisse mit Ihren Erwartungen überein?

Welche Voraussetzungen muss der Datensatz Ihrer Meinung nach erfüllen, damit ein Anhängen von Nullen nahezu ohne Nebenwirkung auf den Amplitudengang bleibt?

Halbiert eine Datensatzlängenverdopplung durch Anhängen von Nullen die im Datensatz analysierbare tiefste Frequenz (sog. Grundfrequenz der harmonischen Analyse)? Begründen Sie Ihre Antwort.

Ab hier folgt der optionale zweite Teil des Laborversuchs (Eigenstudium):

4. Realisierung (Eigenstudium)

4.1 Überführung der Definitionsgleichung in eine algorithmische Beschreibung

Zerlegt man die komplexe DFT-Summenformel (1.1) für die Realisierung in zwei reelle Gleichungen, so wird deutlich, dass die Berechnung von Cosinus- und Sinusfunktionswerten eine wichtige Rolle spielt.

Mit

$$\underline{X}(n) = \mathrm{Re}\{\underline{X}(n)\} + j\,\mathrm{Im}\{\underline{X}(n)\} \qquad mit \quad n = 0,\ 1,\ ...,\ N\text{-}1 \tag{4.1}$$

und

$$\underline{x}(k) = \mathrm{Re}\{\underline{x}(k)\} + j\,\mathrm{Im}\{\underline{x}(k)\} \qquad mit \quad k = 0,\ 1,\ ...,\ N\text{-}1 \tag{4.2}$$

erhält man

$$\mathrm{Re}\{\underline{X}(n)\} = \frac{1}{N}\sum_{k=0}^{N-1}\left(\mathrm{Re}\{\underline{x}(k)\}\cdot\cos\left(\frac{2\pi}{N}kn\right) + \mathrm{Im}\{\underline{x}(k)\}\cdot\sin\left(\frac{2\pi}{N}kn\right)\right) \tag{4.3}$$

und

$$\mathrm{Im}\{\underline{X}(n)\} = \frac{1}{N}\sum_{k=0}^{N-1}\left(\mathrm{Im}\{\underline{x}(k)\}\cdot\cos\left(\frac{2\pi}{N}kn\right) - \mathrm{Re}\{\underline{x}(k)\}\cdot\sin\left(\frac{2\pi}{N}kn\right)\right) \tag{4.4}$$

Für die N komplexen Zeit- und Frequenzwerte werden Arrays vereinbart, die Cosinus- und Sinusfunktionswerte werden sinnvoller weise nicht permanent in Echtzeit berechnet, sondern in Tabellen abgelegt. Diese Tabellen können dann entweder zur Laufzeit vom Zielprozessor berechnet (gefüllt) werden oder sie werden bereits vorab auf einem anderen System (z.B. PC) berechnet und nur in das Projekt mit eingebunden.

C-Beispielcode zur Berechnung der Twiddle-Faktoren (Cosinus- und Sinusfunktionswerte) für eine 512-Punkte-DFT:

```
#define PI  3.1415926  // defined in c6713dsk.h
#define N  512

// twiddle factor arrays:
float Wr[N];
float Wi[N];
```

```
void main()
{
  // often used values:
  // ------------------
  // 2*pi/N:
  const float PI2N = PI * (float)2 / (float)N;
  // or:
  //float PI2N;
  float PI2Nk;
  int k;

  //PI2N = PI * (float)2 / (float)N;
  for (k=0; k<N; k++)
  {
    PI2Nk = PI2N * (float)k;
    Wr[k] = cos(PI2Nk);
    Wi[k] = sin(PI2Nk);
  }
}
```

4.2 DFT reellwertiger Eingangsfolgen $x(k)$

Da die komplexe DFT-Summenformel (1.1) von einer komplexen Eingangsfolge $\underline{x}(k)$ ausgeht, die durch Abtastung des Zeitsignals gewonnene Eingangsfolge aber reellwertig ist und man damit durch Ausschneiden ebenfalls einen reellen Datensatz $x(k)$ erhält, kann der Imaginärteil der Eingangsfolge $\mathrm{Im}\{\underline{x}(k)\}$ in den Summenformeln (4.3) und (4.4) prinzipiell auf zwei (bzw. drei) verschiedene Arten behandelt werden:

1.) Man setzt den Imaginärteil gleich Null

Nachteil:
Die Berechnungen mit dem Imaginärteil werden völlig unnötig durchgeführt und kosten nur Rechenzeit.

2.) Man nutzt den Imaginärteil auch zur Auswertung eines reellen Datensatzes:

a) indem gleichzeitig zwei Datensätze transformiert werden

oder

b) indem mit einer N-Punkte-DFT ein Datensatz der Länge $2 \cdot N$ transformiert wird.
(anders ausgedrückt: Die effektiv nötige DFT-Länge zur Transformation des Datensatzes halbiert sich und die Frequenzauflösung Δf der N-Punkte-DFT beträgt trotzdem $f_a/(2 \cdot N)$.)

Allerdings erfordert die Ausnutzung des Imaginärteils zusätzliche Rechenschritte, so dass die effektive Ersparnis geringer ausfällt.

Im Folgenden werden die nötigen Schritte zur Berechnung der N-Punkte-DFT eines Datensatzes der Länge $2 \cdot N$ beschrieben (siehe dazu auch Literaturangabe [1] in Kapitel 2):

$$DFT\{\underline{x}(k)\} = DFT\{x_1(k) + j \cdot x_2(k)\} \qquad mit \quad k = 0, 1, ..., N\text{-}1 \quad (4.5)$$

Zunächst muss der Datensatz der Länge $2 \cdot N$ also in zwei Hälften zerlegt werden. Und zwar eine mit den Abtastwerten mit geraden Indizes $x_1(k) = g(2 \cdot k)$ und die andere mit den ungeraden

Indizes $x_2(k) = g(2 \cdot k+1)$. Sinnvollerweise geschieht dies bereits beim Ablegen der Abtastwerte im Speicher. Man benötigt also zwei Sample-Arrays (even und odd) jeweils der Größe N.

Danach erfolgt die DFT-Berechnung getrennt nach Realteil

$$X_r(n) = \frac{1}{N} \sum_{k=0}^{N-1} \left(x_1(k) \cdot \cos\left(\frac{2\pi}{N} kn \right) + x_2(k) \cdot \sin\left(\frac{2\pi}{N} kn \right) \right) \qquad (4.6)$$

und Imaginärteil

$$X_i(n) = \frac{1}{N} \sum_{k=0}^{N-1} \left(x_2(k) \cdot \cos\left(\frac{2\pi}{N} kn \right) - x_1(k) \cdot \sin\left(\frac{2\pi}{N} kn \right) \right) \qquad (4.7)$$

jeweils mit: $n = 0, 1, ..., N\text{-}1$

Um nun aus dem Zwischenergebnis $\underline{X}(n)$ das eigentlich gewünschte Ergebnis, nämlich die diskrete Fourier-Transformierte des Datensatzes $g(k)$ also $\underline{G}(n)$ zu erhalten, ist die folgende zusätzliche Berechnung notwendig:

$$\underline{G}(n) = \underline{X}(n) \cdot \underline{A}(n) + \underline{X}(N-n) \cdot \underline{B}(n) \qquad (4.8)$$

getrennt nach Real- und Imaginärteil:

$$G_r(n) = X_r(n) \cdot A_r(n) - X_i(n) \cdot A_i(n) + X_r(N-n) \cdot B_r(n) + X_i(N-n) \cdot B_i(n)$$

$$G_i(n) = X_i(n) \cdot A_r(n) + X_r(n) \cdot A_i(n) + X_r(N-n) \cdot B_i(n) - X_i(N-n) \cdot B_r(n)$$

mit den zuvor berechneten und in Arrays abgelegten Koeffizienten $\underline{A}(n)$ und $\underline{B}(n)$:

$$A_r(n) = 1 - \sin(\pi \cdot n / N)$$

$$A_i(n) = -\cos(\pi \cdot n / N)$$

$$B_r(n) = 1 + \sin(\pi \cdot n / N)$$

$$B_i(n) = \cos(\pi \cdot n / N)$$

jeweils mit: $n = 0, 1, ..., N\text{-}1$ und $\underline{X}(N) = \underline{X}(0)$

Zu beachten ist, dass mit (4.8) nur die erste Hälfte der Frequenzwerte berechnet wird. Die zweite Hälfte kann, falls erforderlich, wegen der bestehenden Symmetrie aus der ersten Hälfte gewonnen werden.

C-Beispielcode zur Berechnung der Koeffizienten:

```
#define PI  3.1415926  // defined in c6713dsk.h
#define N  512

// coefficient arrays:
float Ar[N];
float Ai[N];
float Br[N];
float Bi[N];

void main()
{
  int k;
  float arg, cos_arg, sin_arg;

  for(k=0; k<N; k++)
  {
```

```
    arg = PI * (float)k / (float)N;
    cos_arg = cos(arg);
    sin_arg = sin(arg);
    Ar[k] = 1.0 - sin_arg;
    Ai[k] = -cos_arg;
    Br[k] = 1.0 + sin_arg;
    Bi[k] = cos_arg;
  }
}
```

C-Beispielcode zur Berechnung einer $2N$-Punkte-Transformation mittels einer N-Punkte-DFT:

```
#define N  512

// even and odd sample_buffers:
float even[N];
float odd[N];

// twiddle factor arrays:
float Wr[N];
float Wi[N];

// intermediate frequency sequence:
float Xr[N+1];
float Xi[N+1];

// coefficient arrays:
float Ar[N];
float Ai[N];
float Br[N];
float Bi[N];

// resulting frequency sequence
float Gr[N];
float Gi[N];
float absG[N];  // amplitudes

void main()
{
  register int k, m, n;
  const float N2 = N << 1;
  register float tempXr, tempXi;
  register float tempWr, tempWi;
  register float temp_even, temp_odd;
  register float tempAr, tempAi, tempBr, tempBi;
  register float tempGr, tempGi;

  // dft calculation:
  for(k=0; k<N; k++)
  {
    tempXr = even[0];
    tempXi = odd[0];
    m = 0;
    for(n=1; n<N; n++)  // starts with n=1
    {
      m += k;
      if (m>=N)
        m -= N;
      tempWr = Wr[m];
      tempWi = Wi[m];
      temp_even = even[n];
      temp_odd = odd[n];
      tempXr += temp_even * tempWr + temp_odd * tempWi;
      tempXi += temp_odd * tempWr - temp_even * tempWi;
```

```
    }
    Xr[k] = tempXr;
    Xi[k] = tempXi;
    // with normalization
    //Xr[k] = tempXr/N;
    //Xi[k] = tempXi/N;
  }

  // additional computations to get the final result:
  Xr[N] = Xr[0];
  Xi[N] = Xi[0];
  for(k=0; k<N; k++)
  {
    tempAr = Ar[k];
    tempAi = Ai[k];
    tempBr = Br[k];
    tempBi = Bi[k];
    tempGr = Xr[k] * tempAr - Xi[k] * tempAi + Xr[N-k] * tempBr + Xi[N-k] * tempBi;
    tempGi = Xi[k] * tempAr + Xr[k] * tempAi + Xr[N-k] * tempBi - Xi[N-k] * tempBr;
    Gr[k] = tempGr;
    Gi[k] = tempGi;

    // compute magnitude:
    // no scaling:
    //absG[k] = sqrt(tempGr * tempGr + tempGi * tempGi);
    // scale with 2*N due to the dft optimization for real input sequence of 2*N:
    absG[k] = sqrt(tempGr * tempGr + tempGi * tempGi)/N2;
    // if absolut value is not necessary:
    //absG[k] = tempGr * tempGr + tempGi * tempGi;
  }
}
```

5. Implementierung der DFT auf dem DSK6713 (Eigenstudium)

5.1 Quell-Code-Vorgaben

Für diesen Laborversuch stehen Ihnen folgende Quellcode-Dateien im Projektordner 6 zur Verfügung:

\Project6\dft.c	C-Quellcode
\Project6\init_mcbsp_codec.c	Initialisierung der McBSPs und des On-Board Codec's TLV320AIC23B
\Project6\vectors3.asm	Interrupt-Vektor-Tabelle
\Project6\c6713dsk.h	Include-Datei (C6713 und DSK Definitionen)
\Project6\dsk6713.cmd	Linker-Konfigurationsdatei

Die **Abtastfrequenz des Codec's** TLV320AIC23B wurde gegenüber den bisherigen Projekten durch eine Änderung in *init_mcbsp_codec.c* von 32 kHz auf **8 kHz** reduziert.

5.2 Die Windows-Audio-Software

Auf den Labor-PCs steht Ihnen neben dem Windows Media Player zum Abspielen von CDs und Wave-Dateien auch wieder das Programmpaket AudioTester zur Verfügung, mit dessen Funktionsgenerator waveGen sich über die eingebaute Soundkarte des PC's u.a. sinus- und rechteckförmige Signale einstellbarer Frequenz und Amplitude erzeugen lassen.

5.3 Durchführung, Aufgaben und Fragen

5.1 Erstellen Sie mit Code Composer Studio ein neues Projekt für die DSK6713-Plattform. Fügen Sie die zur Verfügung gestellten Quellcode-Dateien (bis auf die Header-Files) dem Projekt hinzu. Nehmen Sie die nötigen Einstellungen bei den Build Options vor (rts-Library, Allocation Order, Heap Size, Stack Size; vgl. Laborversuch 2, Absatz 3.2). Analysieren Sie den Quellcode, so dass Ihnen danach die Funktionsweise des Programms klar ist.

Anmerkung:

Die Abtastwerte (genauer gesagt jeweils die Monosumme aus linkem und rechtem Kanal) werden durch die Receive-Interruptserviceroutine als Gleitkommazahlen abwechselnd in den getrennten Ringspeichern `buf_e` und `buf_o` abgelegt. (Also getrennt nach geraden und ungeraden Indizes der Eingangsfolge.)
Die Berechnung der $2N$-Punkte-Transformation in `main()` wird gestartet, sobald durch den ersten DIL-Schalter die Freigabe dazu erfolgte und die beiden Ringspeicher gerade mit den neuesten Daten gefüllt sind.

5.2 Compilieren/Linken und starten Sie das Programm. Legen Sie mit Hilfe des (Software-) Funktionsgenerators ein Sinussignal geeigneter Frequenz und Amplitude an den Eingang des Codecs an.

Starten Sie eine DFT-Berechnung durch kurzes Betätigen des ersten DIL-Schalters (`USER_SW0`). Stoppen Sie danach den Programmablauf und überprüfen Sie die Aussteuerung des A/D-Wandlers im Codec, indem Sie sich die Sinusschwingung im Speicher des DSP mit Hilfe der Graph-Funktion von CCS ansehen. Lassen Sie sich ebenfalls das Ergebnis der DFT-Berechnung darstellen. Wählen Sie für dieses Grafikfenster an Stelle der Liniendarstellung über das Graph-Property-Dialog-Fenster die Balkendarstellung aus (Im Graph-Property-Dialog-Fenster beim Eintrag 'Data Plot Style' die Option `Bar` auswählen!). Korrigieren Sie ggf. die Aussteuerung.

Anmerkung:

Wenn Sie im C-Code nach der Amplitudenberechnung an der Stelle `k=0;` einen SW-Breakpoint setzen, dann zeigen die Grafikfenster nach jedem Erreichen des Breakpoints die jeweils aktuellen Daten an. Sie sparen sich damit das wiederholte Stoppen des Programms und das Aktualisieren der Grafikfenster-Inhalte (Wiederanlauf mit Shift F11).

Testen Sie einige verschiedene Frequenzeinstellungen. Stimmen die DFT-Ergebnisse mit der eingestellten Frequenz und Amplitude überein? Beachten Sie dabei die vorgenommene Normierung der Amplitude im C-Code. Erläutern Sie, wie Sie das DFT-Ergebnis überprüfen.

5.3 Stellen Sie am (Software-) Funktionsgenerator eine Sinusschwingung mit einer Frequenz von $16 \cdot \Delta f$ ein und wählen Sie den Ausgangspegel so, dass der A/D-Wandler des Codecs zu ca. 2/3 ausgesteuert wird. Schalten Sie danach den Funktionsgenerator von Sinus auf Rechteck um und sehen Sie sich das DFT-Ergebnis an. Welche Änderungen ergeben sich in der Frequenzdarstellung? Stimmen diese mit Ihrer Erwartung überein?

Ändern Sie danach die Frequenzeinstellung des Funktionsgenerators um + 4 Hz und überprüfen Sie das DFT-Ergebnis.

Ändern Sie jetzt die Frequenz um nur 2 Hz und überprüfen Sie erneut. Beachten Sie dabei insbesondere auch die Amplitudenveränderung der beiden Hauptmaxima.

(Die Fragen sind handschriftlich in der Versuchsausarbeitung zu beantworten!)