ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Signal and Information Processing Laboratory
Institut für Signal- und Informationsverarbeitung

9. Februar 2017

Fachpraktikum Signalverarbeitung

SV2: Digitale Filter und Konvertierung der Abtastrate

1 Einführung

Ein *zeitdiskretes* ("digitales") *Signal* $x[k]$ ist nur zu Zeitpunkten $k \in \mathbb{Z}$ definiert (z.B. ein Audio-File auf dem Computer). Oft wird ein solches Signal dazu verwendet, ein kontinuierliches Signal $x(t)$ zu beschreiben ($t \in \mathbb{R}$). In diesem Fall ist eine *Abtastrate* f_s bzw. ein *Abtastintervall* $T_s = 1/f_s$ definiert.

Die *z-Transformation* von $x[k]$ ist für die komplexe Zahl z definiert als

$$X(z) = \sum_{k=-\infty}^{\infty} x[k] z^{-k} \quad . \tag{1}$$

Das *Spektrum* (der Frequenzgang) von $x[k]$ ist $X(e^{i\Omega})$, d.h. die z-Transformierte für auf dem Einheitskreis liegende z, wobei die *digitale Frequenz* Ω dem Argument von z entspricht und meist von $-\pi$ bis π betrachtet wird, da sie 2π-periodisch ist. Die entsprechende zeitkontinuierliche Frequenz ist $f = \Omega f_s / 2\pi$.

In diesen Versuchen werden zeitdiskrete Signale mit linearen Filtern manipuliert und die Abtastrate konvertiert.

2 Lineare Filter

Ein *lineares Filter* verarbeitet ein Eingangssignal $x[k]$ zu einem Ausgangssignal $y[k]$, indem es eine lineare Kombination von vergangenen Eingangswerten und Ausgangswerten bildet:

$$\begin{aligned} y[k] = a_0\, x[k] + a_1\, x[k-1] + a_2\, x[k-2] + \ldots + a_N\, x[k-N] \\ + b_1\, y[k-1] + b_2\, y[k-2] + \ldots + b_M\, y[k-M]. \end{aligned} \tag{2}$$

Ein *IIR* (infinite impulse response) *Filter* hat mindestens einen Koeffizienten $b_m \neq 0$, während bei einem *FIR* (finite impulse response) *Filter* alle Koeffizienten $b_m = 0$ sind. Die *Ordnung* eines Filters ist definiert als die grössere der Zahlen N und M. Das FIR-Filter kann mit Hilfe der *Impulsantwort*[1] $h[k]$ charakterisiert werden, da dessen Impulsantwort gerade den Koeffizienten $a_n = h[n]$ entspricht. Achtung: diese Impulsantwort hat die Länge $L = N + 1$.

Für IIR- und FIR-Filter wird die z-transformierte Impulsantwort $H(z)$ *Übertragunsfunktion* genannt und hat die allgemeine Form:

$$H(z) = \frac{Y(z)}{X(z)} = \frac{a_0 + a_1\, z^{-1} + a_2\, z^{-2} + \ldots + a_N\, z^{-N}}{1 + b_1\, z^{-1} + b_2\, z^{-2} + \ldots + b_M\, z^{-M}}. \tag{3}$$

In diesen Versuchen betrachten wir oft den *Amplitudengang* $\left|H(e^{i\Omega})\right|$ eines Filters.

[1] Die Impulsantwort ist das Ausgangssignal für das Eingangssignal $x[k] = (1, 0, 0, \ldots)$.

2.1 Filterentwurf

Eine einfache FIR-Filter Entwurfsmethode beginnt mit einer Spezifikation des Amplitudenganges $|H(e^{i\Omega})|$. Ein ideales Tiefpassfilter hat folgende Form:

$$H(e^{i\Omega}) = \begin{cases} 1 & |\Omega| < \Omega_c \\ 0 & \Omega_c \leq |\Omega| \leq \pi \end{cases} \tag{4}$$

Mit der Rücktransformation in den Zeitbereich erhalten wir die gewünschte Impulsantwort:

$$h[k] = \frac{\sin(\Omega_c k)}{\pi k} = \frac{\Omega_c}{\pi}\,\mathrm{sinc}\left(\frac{\Omega_c}{\pi}k\right) \tag{5}$$

Diese klingt beidseitig im Unendlichen ab und wird deshalb mit einer *Fensterfunktion* $w[k]$ multipliziert. Im einfachsten Fall ist dies ein Rechteckfenster mit der Länge L:

$$w[k] = \begin{cases} 1 & |k| \leq L/2 \\ 0 & |k| > L/2 \end{cases} \tag{6}$$

Dadurch entsteht ein Filter das nicht mehr dem idealen Frequenzgang von Gleichung (4) entspricht sondern einen gleitenden Übergang zwischen Sperr- und Passband sowie Unebenmässigkeiten im Frequenzgang in diesen Bereichen mit sich bringt.

Es existieren kompliziertere Fensterfunktionen (z.B. von Blackman und Harris) welche unterschiedliche Aspekte dieser Frequenzgangabweichungen besser gegeneinander abwägen als das Rechteckfenster.

Um die so modifizierte Impulsantwort kausal zu machen, wird sie um $L/2$ Zeitschritte verschoben: $a_k = h[k-L/2]\,w[k-L/2]$. Die resultierende Verzögerung von $L/2$ Zeitschritten wird in Kauf genommen.

Eine mögliche Entwurfmethode für IIR-Filter bestimmt zuerst ein zeitkontinuierliches Filter und diskretisiert dieses.

3 Konvertierung der Abtastrate

Ein Signal $x[k]$ mit Abtastrate $f_{s1} = 1/T_{s1}$ soll umgerechnet werden in ein Signal $y[k]$ mit Abtastrate $f_{s2} = 1/T_{s2}$ ($\neq f_{s1}$). Dies entspricht einer Neuabtastung des zeitkontinuierlichen Signals $x(t)$ auf dem Abtastraster $k\,T_{s2}$.

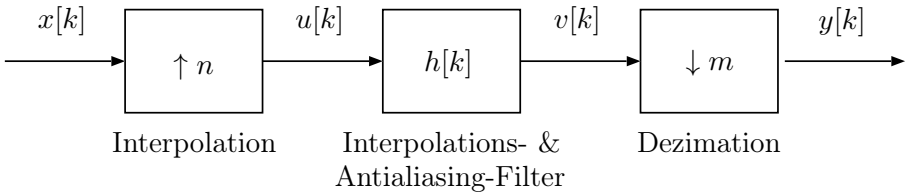

Abbildung 1: System zur Umrechnung der Abtastrate

Für einen rationalen Faktor $\frac{f_{s2}}{f_{s1}} = \frac{n}{m}$ ($n, m \in \mathbb{N}$) kann das System von Abbildung 1 angewendet werden. Es besteht aus einem Interpolationsblock, einem Tiefpassfilter und einem Dezimationsblock.

$\boxed{\uparrow n}$ Im *Interpolationsblock* werden zwischen zwei Werten $x[k]$ und $x[k+1]$ zusätzlich $n-1$ Nullen eingefügt (für jedes k). Die Abtastrate wir somit auf $f_{s(up)} = n\,f_{s1}$ erhöht.

$$u[k] = \begin{cases} x[k/n] & k/n \in \mathbb{Z} \\ 0 & k/n \notin \mathbb{Z} \end{cases} \tag{7}$$

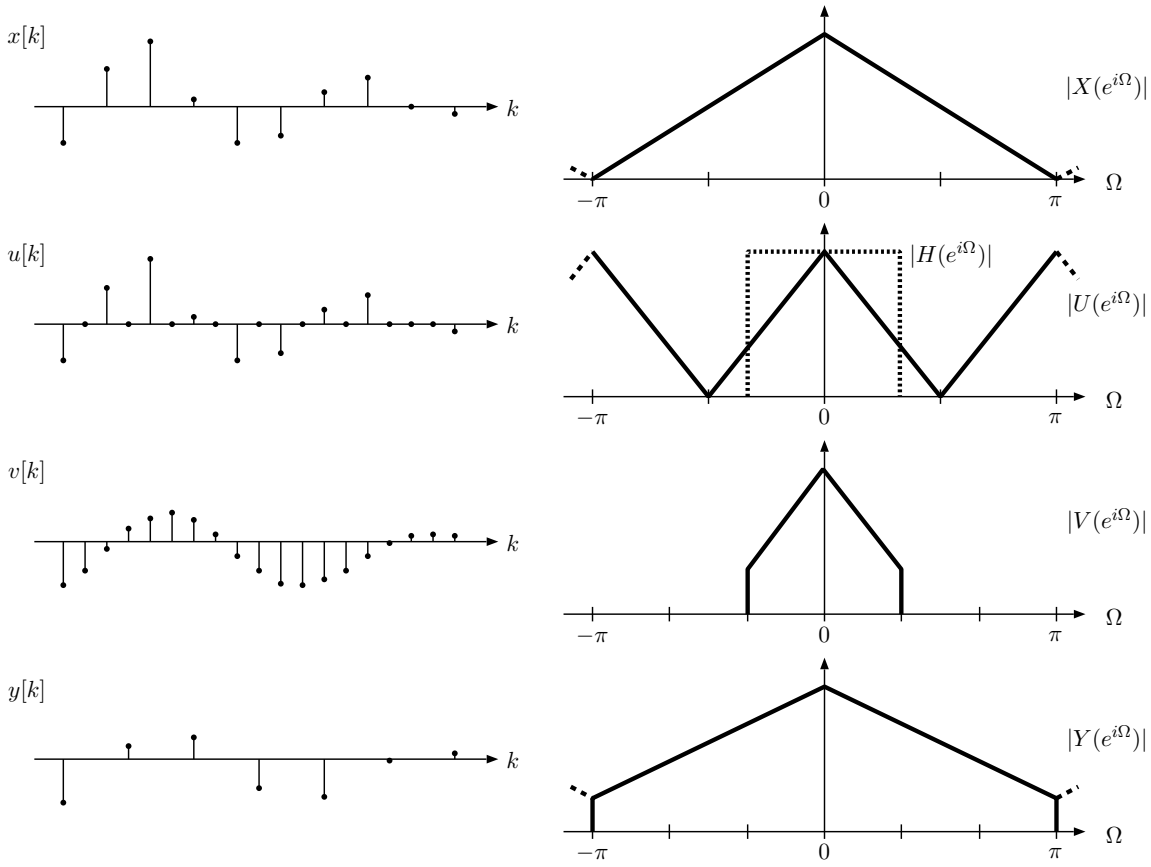

Abbildung 2: Signale und Amplitudengänge bei einer Konvertierung mit $\frac{n}{m} = \frac{2}{3}$

Im Frequenzbereich wird dadurch das Spektrum entlang der Frequenzachse um den Faktor n gestaucht (von $[-\pi, \pi]$ nach $[-\pi/n, \pi/n]$) und $n-1$ mal "kopiert". Als Beispiel für $n = 2$ vergleiche die Amplitudengänge $|X(e^{i\Omega})|$ und $|U(e^{i\Omega})|$ in Abbildung 2.

$\boxed{\downarrow m}$ Der *Dezimationsblock* pickt jeden m-ten Wert aus $v[k]$ heraus und ignoriert die dazwischenliegenden Werte. Die Abtastrate wird somit auf $f_{s2} = f_{s(up)}/m$ abgesenkt.

$$y[k] = v[mk] \tag{8}$$

Dadurch findet im Frequenzbereich eine Streckung um den Faktor m statt (von $[-\pi/m, \pi/m]$ nach $[-\pi, \pi]$). Vergleiche $|V(e^{i\Omega})|$ und $|Y(e^{i\Omega})|$ in Abbildung 2 für $m = 3$.

$\boxed{h[k]}$ Das *Tiefpassfilter* erfüllt zwei Aufgaben gleichzeitig:

- Als *Interpolationsfilter* unterdrückt es in $U(e^{i\Omega})$ die unerwünschten Kopien. Somit bleibt das ursprüngliche Spektrum theoretisch vollständig erhalten.
- In $U(e^{i\Omega})$ muss das Signal auf den Frequenzbereich $-\pi/m \ldots \pi/m$ beschränkt werden. Andernfalls werden bei der Dezimation Signalanteile ausserhalb dieses Intervalles über π (unter $-\pi$) hinausgestreckt. Weil jedes digitale Spektrum periodisch ist, werden diese Anteile quasi in das Spektrum "zurückgefaltet". Dieser Effekt wird *Aliasing* genannt und das Filter $h[k]$ wirkt deshalb auch als *Antialiasingfilter*. Wie in Abbildung 2 ersichtlich ist es dabei möglich, dass die ursprüngliche Form des Spektrums nicht erhalten bleibt.

Damit das kombinierte Interpolations- und Antialiasing-Filter $h[k]$ beide Anforderungen erfüllt wird die Grenzfrequenz

$$\Omega_c = \min\{\pi/n, \pi/m\}. \tag{9}$$

Weiterführende Texte

H.-A. Loeliger, *Zeitdiskrete und statistische Signalverarbeitung*, Kapitel 1 und 2, Vorlesungsskript 2012.

J. G. Proakis und D. G. Manolakis, *Digital Signal Processing*, Prentice Hall, 1996.

4 Versuche

(Eingaben auf der Systemkommandozeile (shell) werden mit ">" gekennzeichnet, Eingaben auf der Matlab-Kommandozeile hingegen mit ">>".)

Kopiere `/home/isistaff/glf/fachprak_isi/SV2` in dein Homeverzeichnis:

```
> cp -irL /home/isistaff/glf/fachprak_isi/SV2 ./
> cd SV2/matlab
> matlab &
```

In den Versuchen geht es darum, Matlab-Skript-Files (.m-Files) im Ordner `SV3/matlab` zu komplettieren und in Matlab mit dem "Filter Analyse- und Designtool" (`fdatool`) zu spielen. Zur Veranschaulichung benutzen wir Audiosignale[2] die alle als wave-Files im Ordner `SV2/matlab/signals` abgelegt werden.

4.1 Lineare Filter

1. Höre dir das Rauschsignal `sig_original.wav` an. Es dauert eine Sekunde und ist mit 44.1 kHz abgetastet.
   ```
   > play signals/sig_original.wav
   ```

2. Im File `fir_window_lp.m` wird Gleichung (5) benutzt, um die Impulsantwort eines FIR-Tiefpass-Filters zu gewinnen. Generiere die Impulsantwort `h` und die mit einer Blackman-Fensterfunktion multiplizierte Impulsantwort `h_w` für eine frei gewählte Filterlänge und Grenzfrequenz:
   ```
   >> f_s = 44100; L = 43; f_c = 8000; window = blackman(L);
   >> h = fir_window_lp(f_s, L, f_c);
   >> h_w = fir_window_lp(f_s, L, f_c, window);
   ```

Weil das FIR-Filter sind, sind in Gleichung (2) alle Koeffizienten $b_m = 0$ und die Matlab-Vektor `h` und `h_w` enthalten die Koeffizienten a_n. [3]

3. Benutze das Matlab-eigene "Filter Visualization Tool" `fvtool` um den Amplitudengang beider Filter anzuzeigen: [4]

[2] Am besten schreibt man zur Ausgabe der Audiodateien eine kurze Matlab Funktion (`play.m`):

```
function play(audiofile)
[y, fs] = wavread(audiofile);
sound(y, fs)
end
```

[3] In der Matlab Dokumentation heissen die Koeffizienten im Zähler von Gleichung (3) b_n und diejenigen im Nenner a_n.

[4] Im `fvtool`-Fenster kann die Abtastrate im Menu `Analysis -> Sampling Frequency` eingestellt werden. Dies bewirkt eine korrekte Beschriftung der Frequenzachse.

`>> fvtool(h, 1, h_w, 1);`
$|H_w(e^{i\Omega})|$ (rote Kurve) unterscheidet sich in folgenden Merkmalen von $|H(e^{i\Omega})|$ (blaue Kurve): Der Übergang zwischen Pass- und Stopband ist weniger scharf, dafür ist die Stopbanddämpfung grösser und der Rippel im Passband kleiner.

4. Schau dir die Impulsantwort und die Schrittantwort an. Klicke dazu im `fvtool`-Fenster auf die entsprechenden Toolbartasten. Hier ist sichtar, dass $h_w[k]$ (rot) ein glatteres Ein- und Ausschwingverhalten aufweist als $h[k]$ (blau).

Mit der Methode von Abschnitt 2.1 kann man auch Hochpass-, Bandpass- und Bandstopfilter entwerfen. Die entsprechenden Funktionen sind in `fir_window_hp.m`, `fir_window_bp.m` und `fir_window_bs.m` implementiert.

5. Implementiere in `fir_window_bs.m` die fehlende Umsetzung der folgenden Gleichung (Diese kann analog zu Gleichung (5) hergeleitet werden):

$$h[k] = \mathrm{sinc}(k) - \frac{\Omega_{c1}}{\pi}\ \mathrm{sinc}\left(\frac{\Omega_{c1}}{\pi}k\right) - \frac{\Omega_{c2}}{\pi}\ \mathrm{sinc}\left(\frac{\Omega_{c2}}{\pi}k\right) \tag{10}$$

6. Berechne analog zum Tiefpassfilterbeispiel in Punkt 2 die Impulsantwort eines Bandstopfilters (z.B: `>> f_c = [5000 15000];`) und plotte den Amplitudengang mit dem `fvtool`.

7. Die Matlab-Funktion `filter` berechnet Gleichung (2), sie filtert also ein Signal. Schau dir `filter_signal.m` an und führe es aus (`>> filter_signal()`), um vier gefilterte Versionen von `sig_original.wav` zu erhalten.

8. Höre dir die generierten wave-Files an. Wenn du möchtest, kannst du `filter_signal.m` so verändern, dass alle Filter ein Rechteckfenster verwenden (lösche das Argument `window` in den Funktionsaufrufen `fir_window_...`) und die Signale unter anderem Namen gespeichert werden. Hörst du einen Unterschied?

Im File `noisy.wav` wird ein Nutzsignal von einem schmalbandigen Störsignal (2.2 - 2.8 kHz) überlagert. Letzteres versuchen wir mit einem Bandstopfilter zu entfernen, wobei das Nutzsignal natürlich auch verändert wird.

9. Höre dir das Nutzsignal `sound.wav`, das Störsignal `noise.wav` und die Addition von beiden `noisy.wav` an.
`> play signals/sound.wav signals/noise.wav signals/noisy.wav`

10. Komplettiere den entsprechend markierten Abschnitt in `denoise_signal.m` so dass ein Bandstopfilter mit Hilfe von `fir_window_bs` angewendet wird.

11. Führe nun das Skript aus: `>> denoise_signal();` (evt. für unterschiedliche Filterlängen `L`) und höre dir das Resultat an: `> play signals/denoised_fir_window.wav`

Matlab unterstützt viele weitere Filterentwurfsmethoden, auch für IIR-Filter. Die meisten kann man mit dem `fdatool` durchführen. `fvtool` ist in `fdatool` integriert.

Wir benutzen nun die genaue Spezifikation in Tabelle 1, um ein elliptisches IIR-Filter und ein Equiripple FIR-Filter zu entwerfen. Insbesondere lassen wir Matlab die erforderliche Filterordnung berechnen.

12. Starte das Tool mit `fdatool`.

13. Wähle "Response Type" > "Bandstop" und "Design Method" > "FIR Equiripple"

Abtastfrequenz f_s	44.1 kHz
Unteres Passband	0 - 1.5 kHz
Stopband	2.2 - 2.8 kHz
Oberes Passband	3.5 - 22.05 kHz
Maximaler Rippel im Passband	0.8 dB
Stopbanddämpfung	80 dB

Tabelle 1: Spezifikation des Bandstopfilters

14. Gib die Parameter aus Tabelle 1 in die Abschnitte "Frequency Specifications" und "Magnitude Specifications" ein.[5]

15. Klicke auf die "Design Filter" Taste. Links oben wird die berechnete Filterordnung angezeigt.

16. Ändere die "Design Method" auf "IIR Elliptic" und klicke abermals auf "Design Filter". Achte auf die Filterordnung.

An dieser Stelle könnte man ein Filter exportieren. [6] Ebenso ist es möglich ein .m-File zu generieren um die aufgerufenen Matlab-Funktionen anzuschauen. Der Entwurf dieser zwei Filter ist auch in `denoise_signal.m` implementiert.

17. Schliesse das `fdatool`-Fenster. Die drei Filter werden in `denoise_signal.m` entworfen und in das `fvtool` geladen mit:
    `>> denoise_signal(1);`
    Im `fvtool`-Fenster werden drei Plots gezeigt:

    Blau $\leftrightarrow$ FIR Filter durch Fenstermethode mit selbstgewählter Filterordnung.
    Grün $\leftrightarrow$ FIR Equiripple Filter.
    Rot $\leftrightarrow$ IIR elliptisches Filter.

18. Höre dir die Resultate `denoised_fir_window.wav`, `denoised_fir_equirip.wav` und `denoised_iir_elliptic.wav` an.

Folgende Beobachtungen und Bemerkungen können gemacht werden:

- Das Nutzsignal hat durch die Filterung hörbare Veränderungen erfahren. Diese Signaltrennung ist nicht optimal aber dafür einfach durchzuführen.
- Das IIR-Filter benötigt eine deutlich kleinere Filterordnung als die FIR-Filter, um die Spezifikation in Tabelle 1 zu erfüllen.
- Der Phasengang beider FIR Filter ist linear. Die Gruppenlaufzeit ist somit für diese beiden Filter konstant. Diese Eigenschaft ist jedoch kaum hörbar.

4.2 Konvertierung der Abtastrate mit rationalem Faktor

19. Höre dir `umrechnen.wav` an. Es enthält ein Sprachsignal welches mit $f_{s1} = 16$ kHz abgetastet wurde. Wir möchten die Abtastrate nun konvertieren auf $f_{s2} = 6$ kHz.

[5] Das untere Ende des unteren Passbandes und das obere Ende des oberen Passbandes müssen nicht angegeben werden, da diese Werte automatisch 0 und die Nyquistfrequenz $f_s/2$ sind.

[6] Falls du das möchtest, wähle für das elliptische Filter zuerst das Menu "Edit > Convert to Single Section" um die Filterkoeffizienten gemäss Gleichung (3) zu erhalten.

20. Berechne das entsprechend gekürzte Verhältnis $\frac{n}{m}$ und führe die Konvertierung mit Hilfe des Skriptes `resample_signal.m` aus:
    `>> resample_signal('./signals/umrechnen.wav', n, m);`
    Höre dir das Resultat `umrechnen6000_wrong.wav` an.

Die Störgeräusche, die du hörst, sind Aliasing und die durch die Interpolation "kopierten" Spektren. Da die Konvertierung ohne Interpolations- und Antialiasingfilter arbeitet, wird das Spektrum wie im Abschnitt 3 beschrieben verzerrt. Dies ist natürlich im Zeitsignal hörbar.

21. Berechne die Grenzfrequenz ($f_c = \Omega_c f_{s(up)}/2\pi$) des benötigten Tiefpassfilters für das Beispiel von Punkt 20. (Achtung: Das Filter arbeitet bei einer Abtastrate von $f_{s(up)} = n\, f_{s1}$.)

22. Implementiere das Tiefpassfilter in `resample_signal.m` an der vorgegebenen Stelle im File. Benutze dazu entweder `fir_window_lp.m` mit selbst gewählter Ordnung oder implementiere ein Equiripple FIR-Filter (oder ein elliptisches IIR-Filter) wie in `denoise_signal.m`.

23. Führe nun nochmals die Konvertierung durch. Das zusätzliche vierte Argument (`1`) schaltet die Filterung ein:
    `>> resample_signal('./signals/umrechnen.wav', n, m, 1);`
    Höre dir das Resultat `umrechnen6000_right.wav` an.[7]

Das Sprachsignal hat zwar durch die Tiefpassfilterung eine Veränderung erfahren, die Aliasing-Geräusche sind aber kaum mehr hörbar. Je tiefer die Ordnung des Filters gewählt wird, desto stärker ist diese Tiefpassfilterung hörbar. Gänzlich unhörbar wird sie auch bei hoher Filterordnung nicht, da das Signal Frequenzanteile oberhalb von 6 kHz aufweist.

Wird die Abtastfrequenz erhöht z.B. auf 24 kHz, so wird – bei genügend hoher Filterordnung – das Sprachsignal kaum eine wahrnehmbare Veränderung erfahren. Probiere es aus wenn du möchtest.

Zum Schluss schauen wir uns die Amplitudengänge für die Abtastenkonvertierung des Signals `triang.wav` an. Da es sich um ein stationäres Signal handelt, können wir das Spektrum mit der diskreten Fouriertransformation abschätzen.

24. Lade das Signal und plotte den Amplitudengang:
    ```
    >> [sig, f_s] = wavread('./signals/triang.wav');
    >> spec = fftshift(fft(sig));
    >> mag_spec = abs(spec); f = linspace(-f_s/2, f_s/2, length(sig));
    >> plot(f, mag_spec);
    ```
    Der Amplitudengang entspricht etwa dem in Abbildung 2. [8]

25. In `resample_signal.m` werden folgende Amplitudengänge für positive Frequenzen f geplottet, falls ein zusätzliches fünftes Argument (`1`) übergeben wird:
    - Amplitudengang des Originalsignals $|X(f)|$
    - Amplitudengang des interpolierten Signals $|U(f)|$ und des Filters $|H(f)|$
    - Amplitudengang des dezimierten Signals $|Y(f)|$

[7] Eventuelle verbleibende Störgeräusche können durch den nicht-idealen Charakter des Filters, durch Quantisierungseffekte oder durch erneute Abtastratenkonvertierung in der Audioausgabe des Betriebssystems erklärt werden.

[8] Oft wird der Amplitudengang in Dezibel umgerechnet. Der "schönen" Dreiecksform wegen lassen wir diesen Schritt hier weg.

26. Führe nun `resample_signal.m` mit den in Punkt 20 berechneten Werten für n und m aus, zuerst ohne Filter:
    `>> resample_signal('./signals/triang.wav', n, m, 0, 1);`
    Betrachte die Plots und schalte nun das Filter dazu:
    `>> resample_signal('./signals/triang.wav', n, m, 1, 1);`
    Höre dir die entsprechenden Signale an.

27. Wähle nun $n = 3$ und $m = 2$ um die Abtastrate auf 24 kHz zu erhöhen. Führe nochmals `resample_signal` aus (evt. musst du die Eckfrequenz f_c des Filters anpassen), mit und ohne Filter. Betrachte die Spektren und höre dir das Resultat an.

Gratulation! Du hast es bis ans Ende geschafft. Falls noch Zeit übrigbleibt, spiele mit den benutzten Matlab-Funktionen und den Audiosignalen.