5.2 Entwurf digitaler Filter

Zur Demonstration eines rekursiv implementierten Tiefpasses (FIR – Finite Impulse Response bzw. IIR – Infinite Impulse Response) soll dieses Beispiel dienen. Eingangsgröße wird ein Rechtecksignal sein, welches von der im DSP integrierten PWM-Einheit oder einem externen Funktionsgenerator erzeugt wird.
Die Implementierung wird so ausgelegt, dass das Variieren von Filtertyp und Koeffizienten ohne Probleme möglich ist. Die Aufnahme des Signals erfolgt durch den AD-Wandler. Nach der Verarbeitung wird das gefilterte Signal über den externen DA-Wandler ausgegeben.

1. Filterentwurf:

Ein für den Filterentwurf sehr geeignetes Hilfsmittel ist Matlab. Mit dessen Hilfe sind ein einfacher Filterentwurf sowie dessen Simulation möglich.

Bei digitalen Filtern kann grob zwischen FIR und IIR-Filtern unterschieden werden. IIR Filter zeichnen sich durch eine zusätzliche Rückkopplung des Ausgangs auf den Eingang aus.

FIR-Filter:

Als Basis für das zu entwerfende Filter wird das Hamming-Fenster dienen. Aufgrund der folgenden Funktion gestaltet sich der Entwurf einfach:

```
b = fir1 (n, Wn);
b = fir1 (15,1000/5000);
```

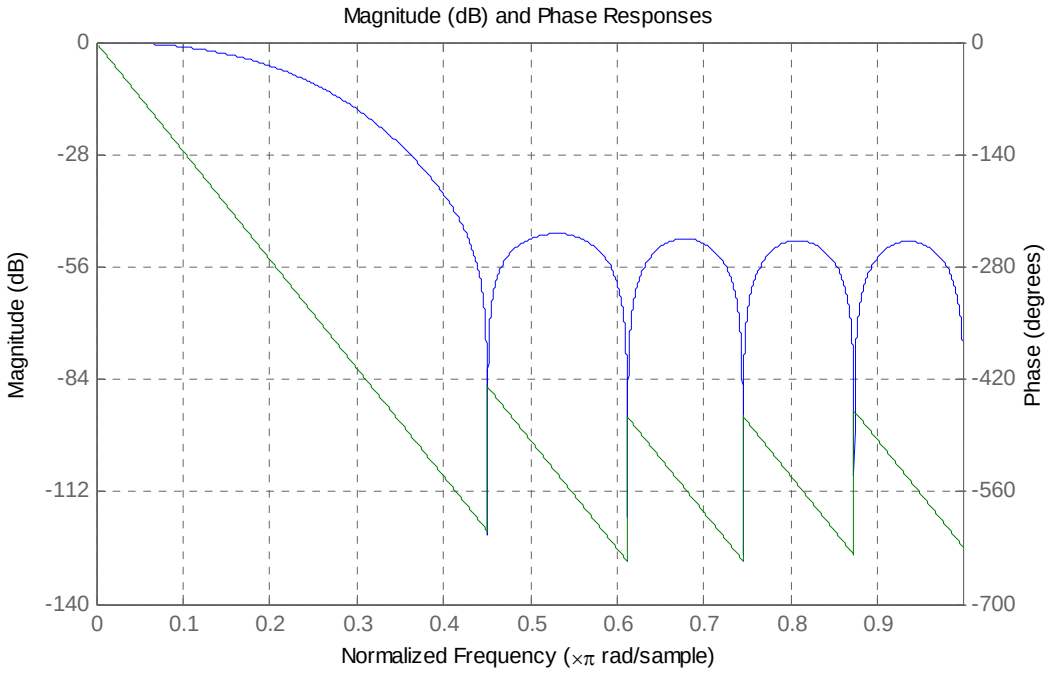
Hierbei ist n die Ordnung des Filters und Wn die normierte Schnittfrequenz. Man erhält die folgenden Werte für die b_i:

```
-0.0035  -0.0049  -0.0042  0.0089  0.0442  0.1002  0.1601
 0.1991   0.1991   0.1601  0.1002  0.0442  0.0089 -0.0042
-0.0049  -0.0035
```

Die Simulation erfolgt nun mit einem 1kHz Rechtecksignal. Theoretisch sollte der Filter sämtliche Oberwellen genügend stark dämpfen, so dass nur noch die Hauptschwingung erhalten bleibt. Die Abtastfrequenz (=Ausgabefrequenz) wird mit 10kHz festgesetzt, so dass eine genügend genaue Nachbildung des erwartetem Sinus am Ausgang erfolgen kann.

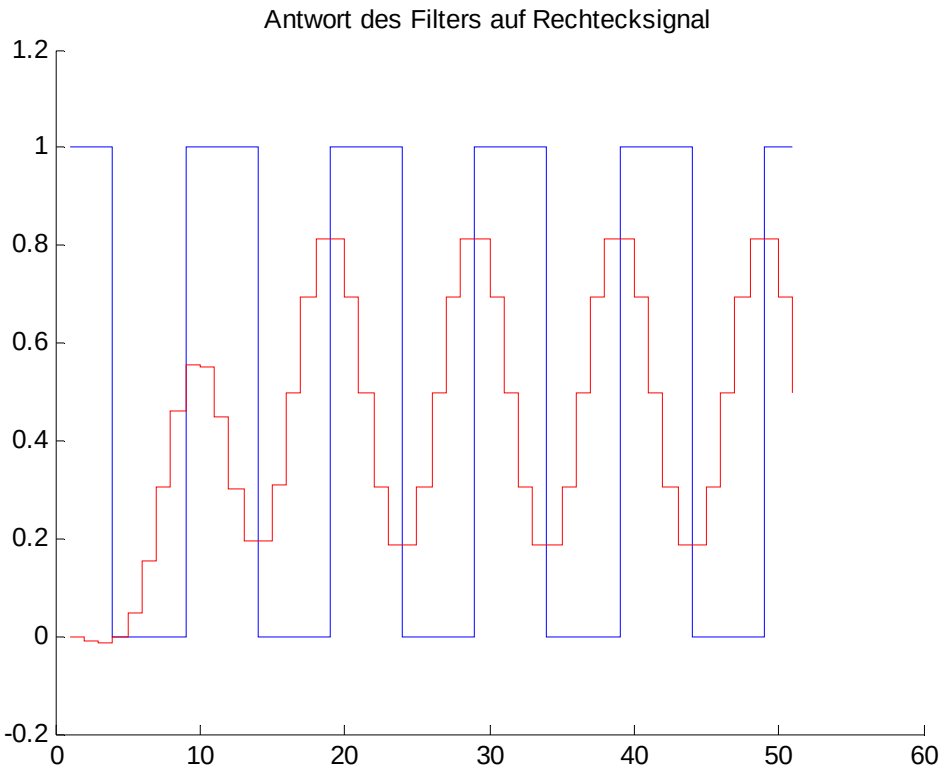
Zunächst soll der Frequenzgang dargestellt werden.

```
hd = dfilt.dffir(b);
freqz(hd);
```



Es ist ersichtlich, dass Frequenzen größer 1kHz zunehmend stärker gedämpft werden. Die Flankensteilheit ist sicherlich nicht perfekt, soll aber für diesen Fall genügen. Besser Charakteristiken lassen sich mittels IIR-Filtern erzeugen, wie später gezeigt wird.

Eine Simulation mit Simulink liefert für das Ausgabesignal das folgende Verhalten:



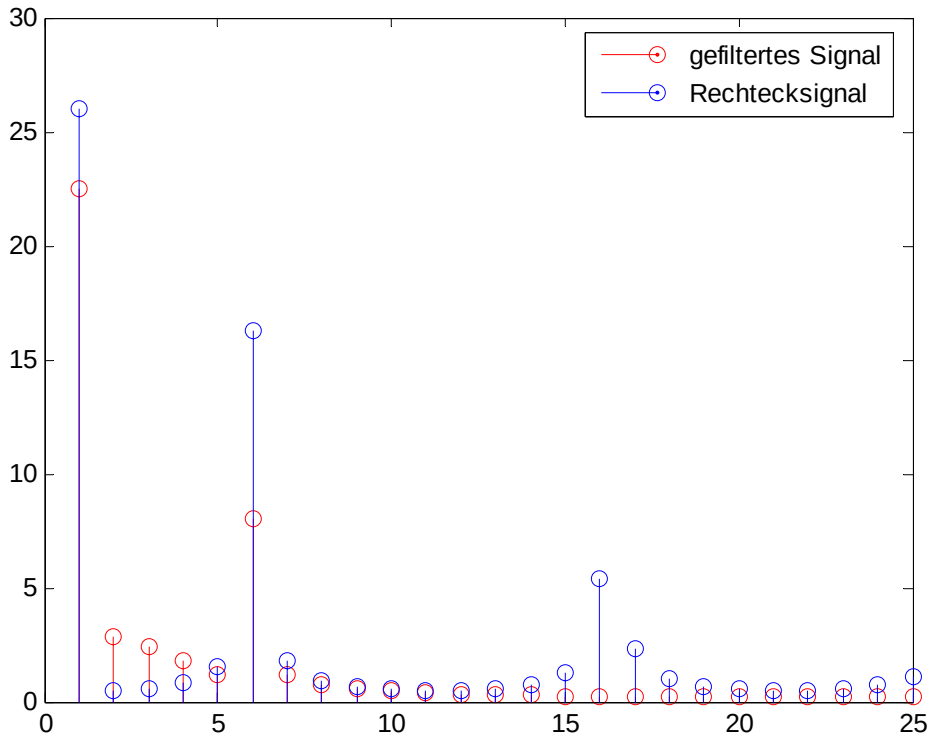
Der Filter verrichtet also seine Arbeit wie gedacht, die Phasenverschiebung ist prinzipbedingt.

Der Matlab-Code hat die folgende Form:

```
ts = .005;                  % Länge der Sequenz
fs = 10000;                 % Abtastrate
fr = 1000;                  % Rechtecksignalfrequenz
p1  = 1/fr/2;               % Pulsweite 50%

ts = 0:1/fs:ts;
d  = 0:1/fr:1;

yr = pulstran (ts, d, 'rectpuls', p1);

y = filter (b, 1, yr);
figure;
hold on;
stairs (yr, 'b');
```

```
stairs (y, 'r');
title ('Antwort des Filters auf Rechtecksignal');
hold off;
```

Der Beweis kann auch im Frequenzbereich geführt werden.



IIR-Filter:

Die Wahl ist hier auf einen Butterworth-Tiefpass gefallen. Dieser zeigt im Durchlassbereich einen flachen Amplitudengang. Die Sprungantwort schwingt jedoch vergleichsweise stark über.

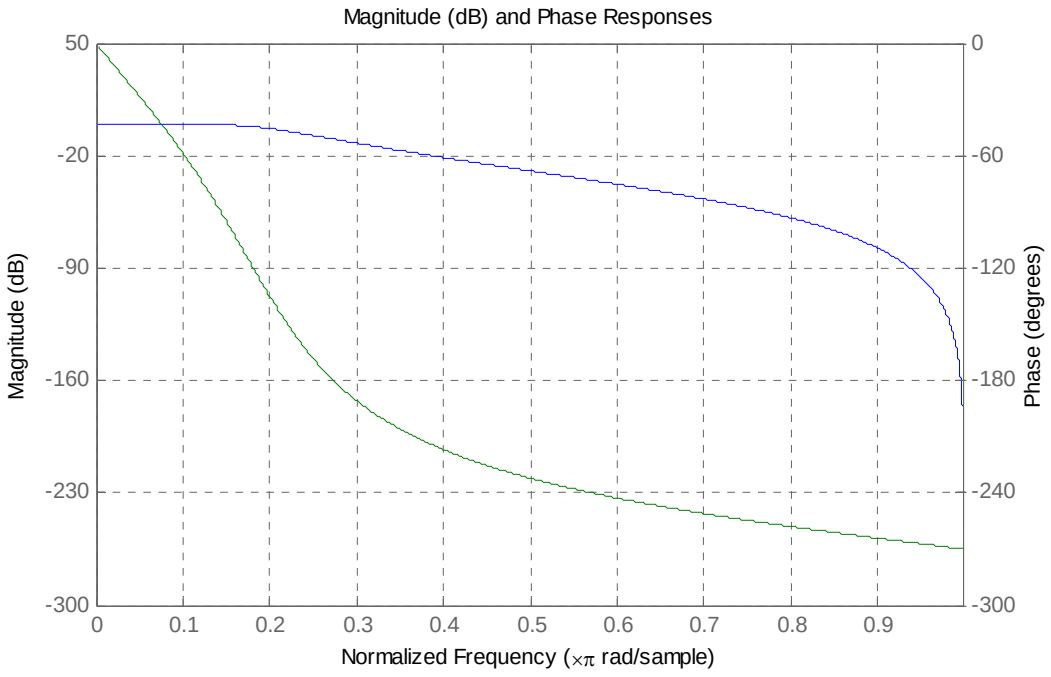
```
[b, a] = butter (n, Wn);
[b, a] = butter (3, 1000/5000);
```

Für die Koeffizienten ergibt sich:

```
ai:
1.0000   -1.7600    1.1829   -0.2781
bi:
0.0181    0.0543    0.0543    0.0181
```

Eine Simulation des Frequenzganges ergibt das folgende Verhalten:

```
hd = dfilt.df1 (b, a);
freqz (hd);
```

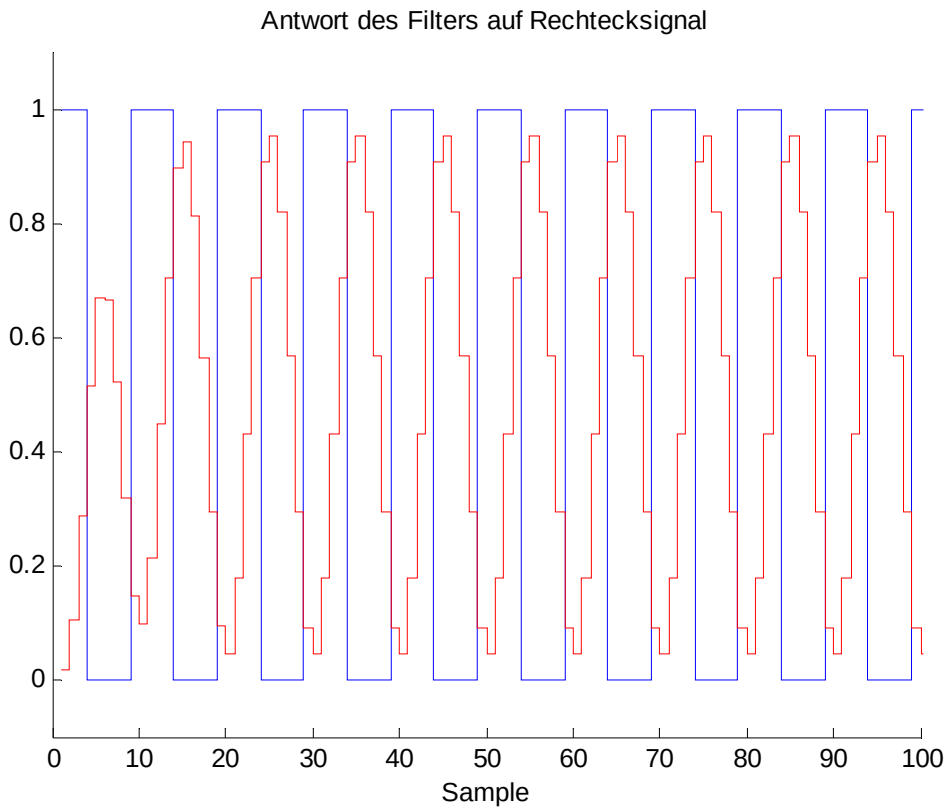


Man sieht, dass sich das gewünschte Verhalten eingestellt hat.

Generell lässt sich bei IIR-Filtern mit geringerem Aufwand (weniger Koeffizienten) normalerweise ein besseres Ergebnis als mit FIR-Filtern erzielen. Erkaufen tut man sich das Ganze mit möglichen Unstabilitäten bedingt durch die Rückkopplung sowie einem nichtlinearen Phasenverlauf.
In der Praxis geht der Trend zum Einsatz von FIR-Filtern. Dem erhöhten Rechenaufwand wird durch die Struktur der Arithmetikeinheiten der DSPs Rechnung getragen, welche Multiply&Accumulate Operationen in einem Zyklus absolvieren können.

Zu guter letzt soll der Filter noch seine Fähigkeiten beim simulierten Filtern zeigen.

```
ts = .01;                % Länge der Sequenz
fs = 10000;              % Abtastrate
fr = 1000;               % Rechtecksignalfrequenz
p1  = 1/fr/2;            % Pulsweite 50%

ts = 0:1/fs:ts;
d  = 0:1/fr:1;

yr = pulstran (ts, d, 'rectpuls', p1);

y = filter (b, a, yr);
figure;
hold on;
stairs (yr, 'b');
stairs (y, 'r');
title ('Antwort des Filters auf Rechtecksignal');
hold off;
```



Wie ersichtlich ist zeigt sich das gewünschte Verhalten.

Es soll abschließend zur Simulation noch gesagt werden, dass Matlab eine weitere Funktion zur Verfügung stellt, mit der sich diverse Filtereigenschaften auf einmal überblicken lassen. Hierzu zählen Betrag, Phase, Sprungantwort, Impulsantwort, usw.
Die Syntax hat die folgende Form:

```
fvtool (b, a);      % Filter Visualization Tool
```

Das mächtigste Filterentwurfswerkzeug innerhalb Matlabs stellt das fdatool dar. Es fasst alle Entwurfsverfahren und Einstellungen unter einer grafischen Oberfläche zusammen. Gestartet wird es über einen einfachen Kommandozeilenaufruf.

```
fdatool;            % Filter Design & Analysis Tool
```