Signalanalyse
Übungsblatt 2

Nikolaus Hammler

13. Februar 2009

Inhaltsverzeichnis

1 Aufgabe 1 - Vergleich von PSD Schätzern

1.1 Aufgabenstellung

Ziel dieser Aufgabe ist es, verschiedene Methoden zur Schätzung der PSD (Power Spectrum Density) zu vergleichen. Dazu wird aus einem gaussverteiltem, weissen Rauschprozess $X[n]$ ein $AR(6)$-Prozess $Y[n]$ erstellt. Der weisse Rauschprozess ist mittelwert frei mit Varianz 25, also $X[n] \sim \mathcal{N}(0, 25)$. Der zugrunde liegende $AR(6)$-Prozess ist vorgegeben:

$$H(z) = \frac{1}{1 - 2,6934z^{-1} + 2,4748z^{-2} - 0,6627z^{-3} + 0,1519z^{-4} - 0,5061z^{-5} + 0,2801z^{-6}} \tag{1}$$

Folgende zwei PSD Schätzer sollen mit einer analytischen Lösung (aufbauend auf Gleichung 1) verglichen werden:

- **Nicht-parametrisch**: Eine Periodogramm-Mittelung nach Welch mit 50% Überlappung und Hammingfenster
- **Parametrisch**: Eine Schätzung des AR-Modells nach der Methode von Burg mit Anzahl an Parametern $m = \{1 \dots 10\}$.

Es soll eine Frequenzauflösung von mindestens $\Delta\omega \leq \frac{\pi}{1024}$ verwendet werden. Die Methoden sollen mittels MSE (Mean-Squared-Error) verglichen werden.

1.2 Implementierung

Der weisse Rauschprozess wird durch die Funktion `randn` erstellt:

```
sigma_x2 = 25;
x = sqrt(sigma_x2)*randn(1,10^6);
```

Der $AR(6)$-Prozess nach Gleichung 1 wird mittels `filter` erstellt, woraus sich $Y[n]$ ergibt:

```
B = [1];
A = [1 -2.6934 2.4748 -0.6627 0.1519 -0.5061 0.2801];
y = filter(B, A, x);
```

Zum Berechnen der analytischen Lösung habe ich eine Funktion `squaredspectrum` geschrieben, die von einen LTI-Filter (spezifiziert durch die Filterkoeffizienten A und B) das quadrierte Betragsspektrum berechnet:

```
function H = squaredspectrum(B, A, N)
    H = zeros(N, 1);
```

```
    w = linspace(0, 2*pi, N);

    for i=1:N
        H(i) = abs(sum(B.*exp(j*(0:length(B)-1)*w(i))) ...
            / sum(A.*exp(j*(0:length(A)-1)*w(i))))^2;
    end
end
```

Die Autokorrelationsfolge eines weissen Rauschprozesses hat genau einen Peak bei Null (bei $k = 0$ ist die einzige Selbstähnlichkeit), da die Samples von weissem Rauschen per Definition untereinander statistisch unabhängig sind. Da also bei $k = 0$ alle Samples des gesamten Signals quadriert sind, entspricht dies genau der Signalleistung σ_x^2. Dementsprechend gilt für die (theoretische) Autokorrelationsfolge von $X[n]$:

$$\phi_{XX}[n] = \sigma_x^2 = 25 \tag{2}$$

Für ein LTI System gilt zwischen der PSD des Eingangssignals und der PSD des Ausgangssignals folgende Beziehung:

$$P_{YY}(\omega) = P_{XX}(\omega) \cdot |H(\omega)|^2 \tag{3}$$

Um nun die PSD von $\phi_{XX}[n]$ zu bestimmen, wird die DTFT angewandt (Wiener Chinchien Theorem). Für die Fouriertransformation gibt es unterschiedliche Definitionen die sich um einen Normierungsfaktor voneinander unterscheiden. So ist die DTFT im Skriptum wie folgt definiert:

$$X(\omega) = \sum_{n=\infty}^{\infty} x[n] \cdot e^{-j\omega n}$$
$$x[n] = \frac{1}{2\pi} \int_0^{2\pi} X(\omega) e^{j\omega n} \, \mathrm{d}\omega$$

Eine andere Definition verwendet jedoch MATLAB, die die Normierung genau umgekehrt verwendet:

$$X(\omega) = \frac{1}{2\pi} \sum_{n=\infty}^{\infty} x[n] \cdot e^{-j\omega n} \tag{4}$$

Beide Varianten sind äquivalent, solange sie konsistent verwendet werden. Das bedeutet konkret: Verwendet man die Definition von MATLAB, muss auch Gleichung 4 für die analytische Gleichung verwendet werden. Verwendet man die Definition aus dem Skriptum, so muss man die Normierung bei den Ergebnissen von MATLAB rückgängig machen und die Ergebnisse von MATLAB mit 2π multiplizieren. Ich habe mich für die Definition von MATLAB entschieden, muss also auch für die Berechnung der analytischen PSD die Definiton von Gleichung 4 verwenden. So berechnet sich meine analytische PSD vom Eingangssignal mit Gleichung 2 und 4 zu:

$$P_{XX} = \frac{1}{2\pi} \sum_{n=\infty}^{\infty} \phi_{XX}[n] \cdot e^{j\omega n} = \frac{1}{2\pi} \sum_{n=\infty}^{\infty} 25\delta[n] \cdot e^{j\omega n} = \frac{25}{2\pi} \tag{5}$$

Gleichungen 3 und 5 wurden im MATLAB-Code wie folgt implementiert und daraus die analytische PSD für $Y[n]$ bestimmt:

```
P_xx = sigma_x2/(2*pi);
P_analytic = P_xx * squaredspectrum(B, A, N);
```

Als geeignete Frequenzauflösung habe ich die gleiche gewählt wie bei den anderen Schätzern, damit diese untereinander vergleichbar sind.

Die nicht-parametrische Schätzung der PSD nach Welch wurde mit folgendem MATLAB-Code erstellt:

```
window = window(@hamming, N);
[P_nonparam, W] = pwelch(y, window, length(window)/2, N, 'twosided');
```

Anforderungen Dieser Befehl kommt der Anforderung nach dem Hammingfenster und der Überlappung von 50% nach. Die Anforderung an die Frequenzauflösung ist direkt gekoppelt mit der Länge des Fensters. Denn die Periodogramm-Methode mittelt ganz einfach nur über mehrere Datenfenster und die FFT eines Datenfensters hat natürlich die gleiche Länge wie das Fenster selbst. Deswegen ist die Länge eines Datenfensters gleich zu wählen wie die Anzahl an darstellbaren Frequenzkomponenten. Diese soll mindestens 1024 betragen. Ich habe deshalb $N = 4096$ gewählt, was einer Frequenzauflösung von $\Delta\omega = \frac{1}{4096}$ entspricht. Die Anzahl der FFT Punkte habe ich explizit noch einmal angegeben. Ich habe also einen vierfach höheren Wert gewählt als die Anforderung weil ich eine höhere Frequenzauflösung und damit eine größere Länge des Datensignals gegenüber der Anzahl der Mittelungen bevorzuge (die standardmäßige Anzahl an Mittelungen bei `pwelch` ist gar nur 8).

Zweiseitiges Spektrum Da die Autokorrelationsfunktion stets symmetrisch ist und daher die PSD als Fouriertransformierte ein symmetrisches Spektrum um π hat, gibt MATLAB automatisch ein **ein**seitiges Spektrum aus. Das genügt, da ein zweiseitiges Spektrum lediglich Redundanzen beinhaltet. Allerdings sind beim einseitigen Spektrum ein paar Spezialfälle zu beachten: Da nur die Hälfte vom Spektrum dargestellt wird, muss das einseitige Spektrum verdoppelt werden damit das Spektrum die gleiche Leistung besitzt. Da jedoch der Gleichanteil bei beiden Spektrumtypen vorkommt, darf dieser nicht verdoppelt werden. Und damit nicht genug: Abhängig von der Größe des Spektrums (gerade/ungerade) darf auch das letzte Sample des einseitigen Spektrums nicht mit 2 multipliziert werden, da er ebenfalls in beiden Spektrumstypen vorkommen würde. Um diesen ganzen Unannehmlichkeiten aus dem Weg zu gehen habe ich mich von vornherein für ein **zweiseitiges** Spektrum entschieden.

Um die parametrischen Schätzer nach der Methode von Burg zu erstellen, habe ich die Resultate einfach in einer Matrix `P_param` zusammengefasst und mittels einer `for`-Schleife erstellt:

```
for m=1:10
    tic;
    HBurg = spectrum.burg(m);
    HSpec = psd(HBurg, y, 'NFFT', N, 'SpectrumType','twosided');
    P_param(:,m) = HSpec.Data;
    times(m) = toc;
end
```

Um die Performance der PSD Schätzer vergleichen zu können, werden diese jeweils mit der analytischen Lösung verglichen. Der Vergleich soll anhand des MSE (Mean-Squared-Error) bestimmt werden, der wie folgt definiert ist:

$$\mathrm{mse}(m) = \frac{1}{2\pi} \int_{-\pi}^{\pi} \left\| \hat{P}_{XX}^{\mathrm{param}}(\omega, m) - \hat{P}_{XX}(\omega) \right\|^2 \mathrm{d}\omega \tag{6}$$

Im Endeffekt sagt Gleichung 6 nichts anderes aus, als dass die Differenz jeder Frequenz quadiert wird, zusammengezählt wird und durch die Anzahl an Frequenzen normiert wird. Um dies einfach zu implementieren, habe ich die analytische Lösung mittels `repmat` erweitert und kann so die analytische Lösung von *allen* nicht-parametrischen Lösungen auf einmal abziehen:

```
P = repmat(P_analytic, 1, 10);
mse = sum(abs(P_param - P).^2)/(size(P,1));
```

Schließlich soll noch der MSE zwischen der analytischen Lösung und der nicht-parametrischen Lösung bestimmt werden. Analog wird Gleichung 6 wie folgt implementiert:

```
mse_nonparam = sum(abs(P_nonparam - P_analytic).^2)/size(P,1);
```

1.3 Ergebnisse und Interpretation

Als Benchmark dient dient die analytische Lösung, die logarithmisch in dB in Abbildung 1 dargestellt ist. Obwohl ich im Abschnitt 1.2 beschrieben habe, dass ich zweitige Spektren verwende, habe ich jeweils nur den Bereich von 0 bis π geplottet.

In Abbildung 2 ist die nicht-parametrische Schätzung nach Welch zu sehen. Der Verlauf ist gleich wie die der analytischen Lösung in Abbildung 1, jedoch erkennt man Fluktuationen. Dieser Umstand kommt daher, dass einfach nicht das zugrundeliegende Modell des $AR(6)$-Prozesses geschätzt wird sondern vielmehr eine Art analytisches Spektrum durch die FFT bestimmt wird und danach gemittelt wird.

In Abbildung 3 sind die Ergebnisse der parametrischen Methode von Burg dargestellt. Ich habe alle Ordnungen übereinander dargestellt damit man sie gut vergleichen

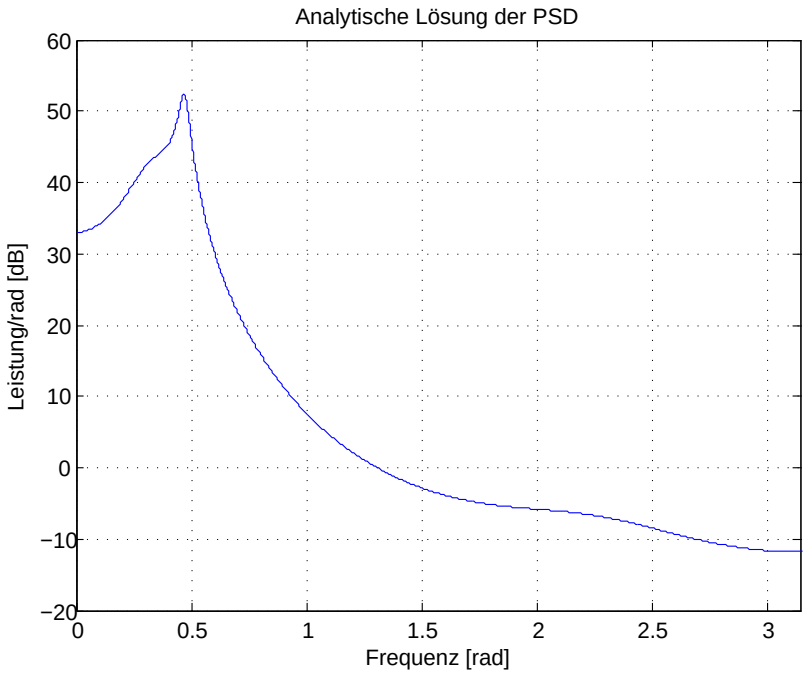


Abbildung 1: Analytische Lösung der PSD

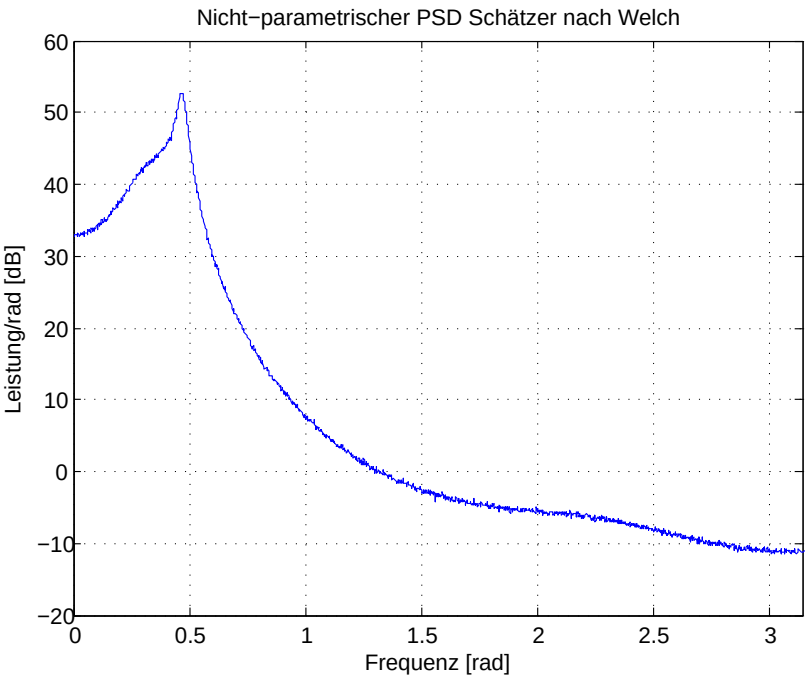


Abbildung 2: Nicht-parametrische Schätzung nach Welch

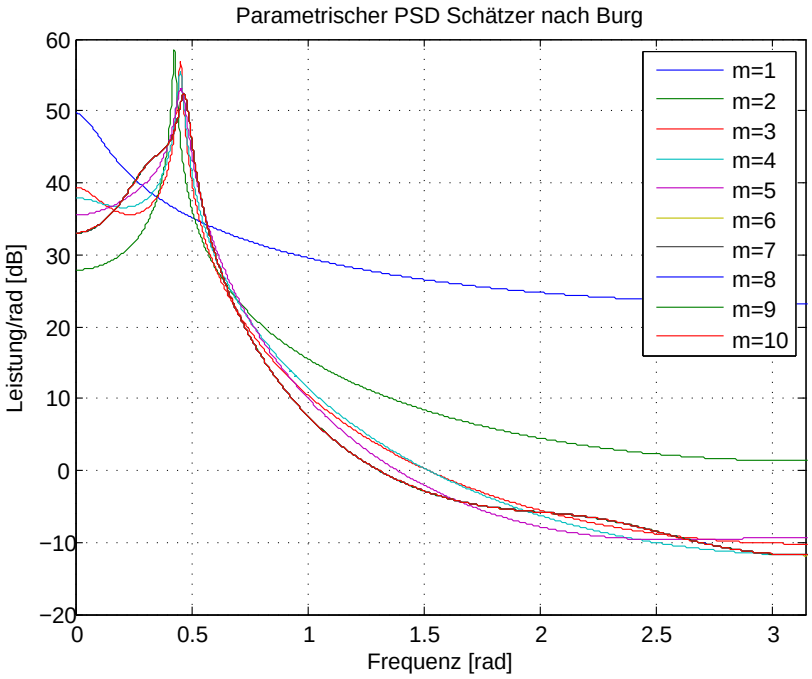


Abbildung 3: Parametrische Schätzungen nach Burg

kann. Das erste was sofort im Vergleich zur nicht-parametrischen Methode (Abbildung 2) auffällt ist, dass alle Kurven sehr glatt sind und keine Fluktuationen aufweisen. Das liegt daran, dass die Methode nicht die PSD schätzt, sondern das zugrundeliegende AR-Modell selbst. Sehr markant sieht man, dass die Kurve für $m = 1$ sehr weit von der analytischen Lösung (Abbildung 1) entfernt ist. Das liegt daran, dass für die Schätzung des Modells nur ein einziger Koeffizient verwendet wird, d.h. der Schätzer hat nur einen Freiheitsgrad obwohl der echte Prozess 6 Koeffizienten verwendet.

Weiters erkennt man, dass bereits mit zwei Koeffizienten der Peak annähernd richtig getroffen wird. Mit steigendem m nähert sich der Verlauf immer mehr dem analytischen aus Abbildung 1.

In Abbildung 4 ist der absolute Fehler pro Anzahl an Koeffizienten der parametrischen Schätzung geplottet. Man erkennt deutlich den monoton fallenden Verlauf des Fehlers der die Aussage von Abbildung 3 unterstreicht: Je mehr Koeffizienten vorhanden sind, desto besser kann das AR-Modell geschätzt werden. Das beste Ergebnise bekommt man ab $m = 6$. Das ist intuitiv sehr leicht nachvollziehbar, denn der echte Prozess hat genau 6 Koeffizienten. Ab 6 Koeffizienten gibt es ein *Overmodeling*. Die Performance kann dadurch natürlich nicht weiter steigen. Es ist allerdings zu beachten, dass der Fehler keinesfalls ab $m = 6$ Null wird, wie die Grafik in Abbildung 4 vermuten lassen würde: Der Unterschied zwischen den Modellen ist zwar markant, jedoch ist der Fehler ab $m = 6$ noch immer in einer Größenordung von 100. Das liegt daran, dass das Modell auch nur eine Schätzung anhand von endlich vielen Stichproben darstellt.

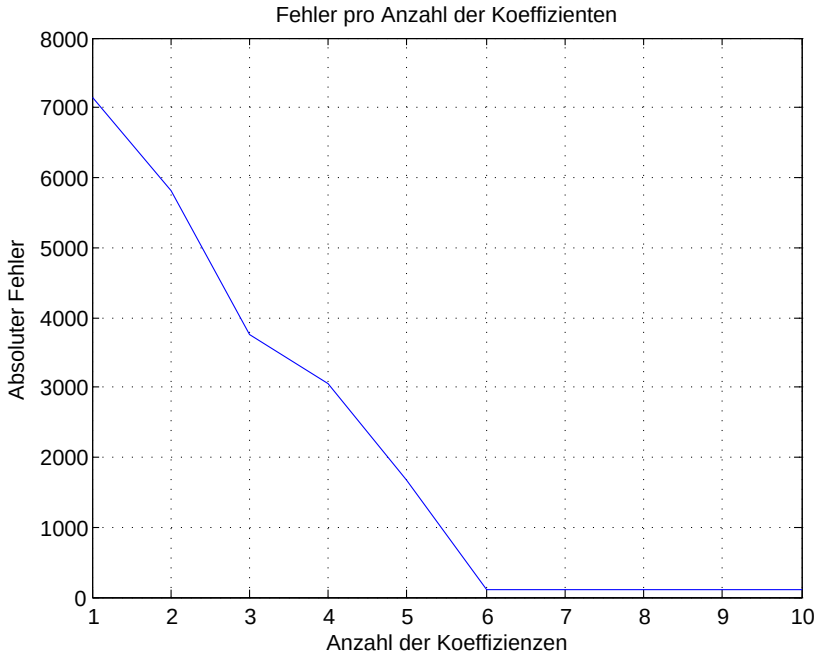
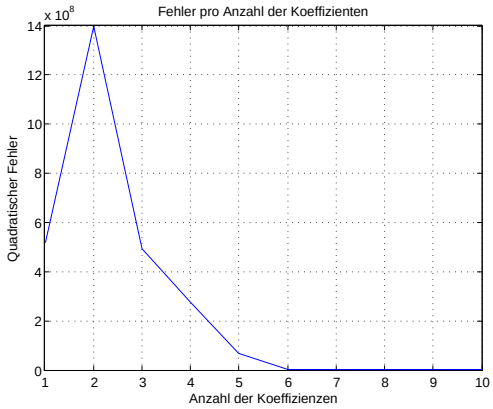


Abbildung 4: Fehler pro Anzahl der Koeffizienten

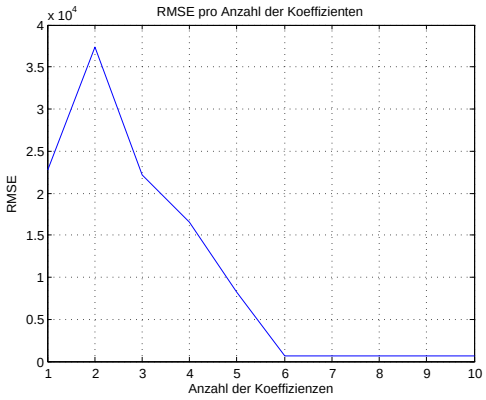
Ein sehr interessantes Phänomen tritt jedoch zu Tage, wenn man sich nicht den absoluten Fehler ansieht, sondern den laut Angabe geforderten quadratischen Fehler. Der MSE ist in Abbildung 5(a) dargestellt. Damit man die gleiche Dimension wie den normalen Fehler enthält, kann man noch die Wurzel ziehen und kommt zum RMSE, der in Abbildung 5(b) dargestellt ist. Es ist deutlich zu erkennen dass der quadratische Fehler bei $m = 2$ viel größer ist als bei $m = 1$! Betrachtet man nur die Linien in Abbildung 3, so ist dies nicht einfach erklärbar.

Da Abbildung 3 logarithmisch geplottet ist, sind die Ergebnisse für $M = \{1, 2\}$ sowie die analytische Lösung noch einmal nicht-logarithmisch in Abbildung 6 dargestellt. Man erkennt deutlich, dass die Höhe des Peaks für $m = 2$ markant höher ist als das Ergebnis für $m = 1$. Und genau hier liegt die Erklärung: Der MSE bzw. RMSE ist so etwas wie eine Norm, genauer eine L_2-Norm (im Gegensatz zum normalen Fehler, der einer L_1-Norm entsprechen würde). Konkret bedeutet dies, dass höhere Fehler viel stärker gewichtet werden als kleine Fehler, da sich das Quadrat auf höhere Zahlen viel stärker auswirkt als auf kleinere Zahlen. Und genau das sagt Abbildung 6 aus: Der gesamte Fehler ab ca. 0,6 rad wirkt sich quadriert nicht so stark aus wie der völlig überhöhte Peak von $m = 2$.

Intuitiv könnte sich der Sachverhalt wie folgt interpretieren: Mit einem Freiheitsgrad wird die tendentielle Richtung abgeschätzt (ein Freiheitsgrad). Der Fehler ist zwar sehr groß, aber dadurch dass der Peak nicht geschätzt wird kann der quadratische Fehler auch nicht ins Unermessliche wachsen. Mit zwei Freiheitsgraden könnte man intuitiv zwei Dinge schätzen: Die Stelle des Peaks und dessen Höhe. Ein Koeffizient könnte die



(a) MSE



(b) RMSE

Abbildung 5: Quadratische Fehler

Stelle geschätzt haben (und schon relativ gut getroffen haben), die Höhe wurde jedoch falsch geschätzt. Da ab $m = 3$ immer mehr Koeffizienten zur Systembeschreibung zu Verfügung stehen und die Höhe des Peaks wieder besser getroffen wird, sinkt auch der quadratische Fehler (bzw. der RMSE) wieder.

Zu guter Letzt soll noch der MSE zwischen der analytischen Lösung und der nicht-parametrischen Lösung nach Welch verglichen werden. Dieser beträgt:

```
1.0241e+006
```

Hier sind die Ergebnisse aus Abbildung 5(a) noch einmal als Zahlenwerte angegeben:

m	MSE $\times 10^9$
1	0.5192
2	1.3965
3	0.4935
4	0.2757
5	0.0678
6	0.0004
7	0.0004
8	0.0004
9	0.0004
10	0.0004

Anhand dieser Werte erkennt man dass der Fehler bei den parametrischen Schätzungen prinzipiell kleiner ist, jedoch nur wenn man die richtige Modellordnung erwischt hat! Und dann ist der Fehler auch nicht um Zehnerpotenzen niedriger sondern „lediglich“ um den Faktor 2,5. Das ist dadurch erklärbar, dass mit der Periodogrammschätzung die PSD direkt geschätzt wird. Dadurch entstehen sehr viele Fluktuationen und ein

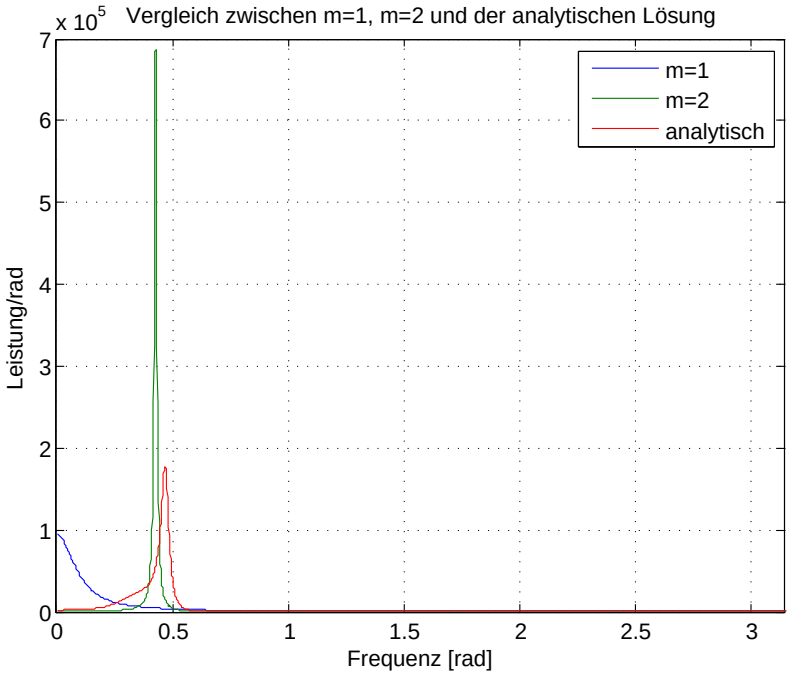


Abbildung 6: Schätzung nach Burg für die ersten 2 Ordnungen

nicht-glatter Verlauf. Die prinzipielle Tendenz der PSD wird jedoch gut getroffen. Beim parametrischen Schätzer wird ein ganzes Modell geschätzt was eine glatte PSD erzeugt. Dadurch ist der quadratische Fehler kleiner, sofern die Modellordnung mindestens so groß ist wie der AR-Prozess selbst.

Um den Fehler jedoch zu relativieren, kann der quadratische Fehler in dB bestimmt werden:

$$\mathrm{mse}(m)|_{\mathrm{dB}} = 20 \cdot \log\left(\frac{P_{\mathrm{nonparam}}}{P_{\mathrm{analytic}}}\right)$$

Dies wurde in MATLAB mit folgendem Code umgesetzt:

```
mse_db = sum(abs(20*log10(P_param ./ P)))/size(P,1);
mse_nonparam_db = sum(abs(20*log10(P_nonparam ...
./ P_analytic)))/size(P,1);
```

Das Ergebnis ist in Abbildung 7 zu sehen: Der Fehler im Vergleich zur analytischen Lösung relativiert sich hier; er beträgt ab $m = 6$ ca. Null dB. Im Vergleich dazu beträgt die die Abweichung zum nicht-parametrischen Schätzer ca. **0,5 dB**.

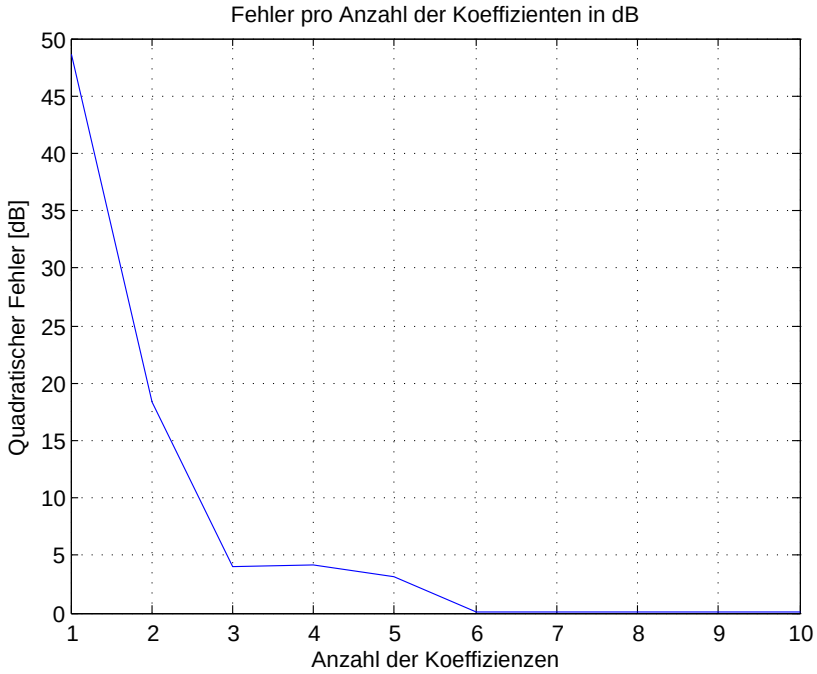


Abbildung 7: Quadratischer Fehler in dB

m	Fehler [dB]
1	48.5926
2	18.4051
3	4.0723
4	4.1715
5	3.1907
6	0.0611
7	0.0624
8	0.0680
9	0.0684
10	0.0684

Die parametrische PSD-Schätzung ist also viel leistungsstärker und sofern die Modellordnung bekannt ist sollte ein parametrischer Schätzer verwendet werden. Nachteilhaft ist jedoch die Laufzeit, die linear mit der Modellordnung anwächst. Abbildung 8 zeigt die Zeit pro Modellordnung. Der lineare Zusammenhang ist gut erkennbar. Ausserdem ist die Performance des parametrischen Schätzers *viel* schlechter als die des nichtparametrischen Schätzers, sofern die Modellordnung **nicht** getroffen wird. Folglich sollte man den parametrischen Schätzer nicht verwenden wenn die Modellordnung nicht bekannt ist.

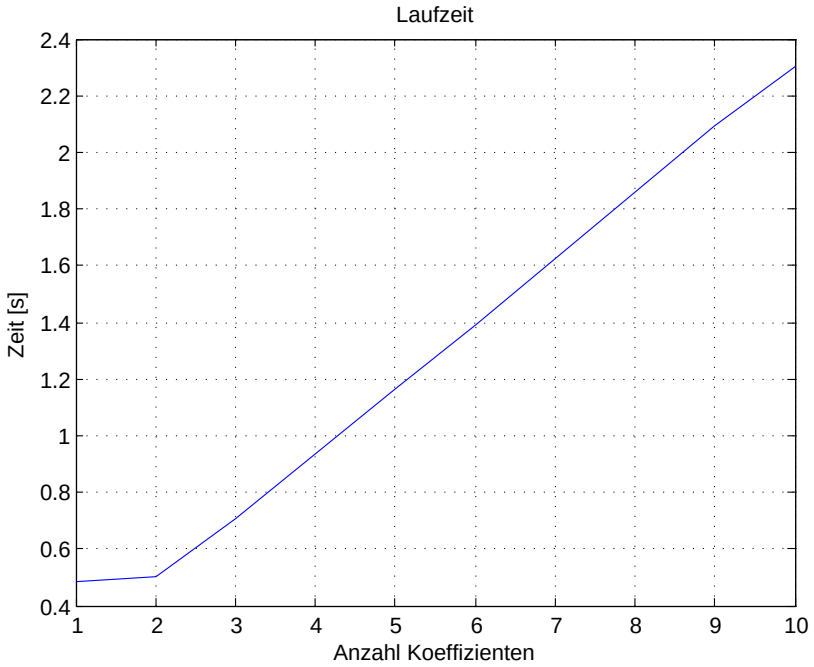


Abbildung 8: Laufzeit pro Modellordnung

2 Aufgabe 2 - Entwurf eines zeitdiskreten Tiefpassfilters

2.1 Aufgabenstellung

Ein Signal $x(t)$ besteht aus einem 1 Hz Nutzsignal und aus einem 5 Hz Störsignal:

$$x(t) = 5 \cdot \sin(2\pi t) + 2 \cdot \sin(10\pi t) \tag{7}$$

Dieses Signal wird mit 50 Hz abgetastet und nun soll ein zeitdiskreter Tiefpassfilter entworfen werden, der diese beiden Signalkomponenten trennt:

- Der Tiefpassfilter soll aus einem zeitkontinuierlichen Butterworthfilter gebaut werden
- Die Dämpfung der Nutzfrequenz soll maximal 1 dB betragen
- Die Dämpfung der Störfrequenz soll mindestens 35 dB betragen

Das Design muss durch die Anforderung an den Butterworthfilter im Analogen geschehen und durch die Bilineartransformation in einen diskreten Filter übergeführt werden. Zum Schluss soll die Übertragungsfunktion des entstandenen Tiefpassfilters im z-Bereich sowie die Differenzengleichung, der Signalflussgraph und die Anzahl der benötigten Rechenschritte ermittelt werden.

2.2 Vorarbeiten

Prinzipiell soll ein zeitdiskreter Filter entworfen werden. Deswegen müssen die benötigten Frequenzen zuerst in ein diskretes Spektrum übergeführt werden. Dieses ist jedoch von der Abtastfrequenz abhängig und wird dadurch in einen Bereich zwischen 0 und 2π übergeführt. In diesem Bereich entsprechen die kontinuierlichen Frequenzen diskreten Frequenzen. Die Umrechnung geschieht nach folgender Formel:

$$\omega = \frac{\Omega}{f_s} = \frac{2\pi f}{f_s} \tag{8}$$

Nach Formel 8 folgt für die Frequenzen 1 Hz und 5 Hz:

$$\begin{aligned} \text{Für 1 Hz:} \quad & \omega_1 = \frac{2\pi}{50} \approx 0.3142\,[\text{rad}] \\ \text{Für 5 Hz:} \quad & \omega_2 = \frac{10\pi}{50} \approx 0.6283\,[\text{rad}] \end{aligned} \tag{9}$$

Um den analogen Filter nun designen zu können, müssen die Werte aus Gleichungen 9 auch wieder in einen analogen Bereich transformiert werden. Die Bilineartransformation ist allerdings eine stark nichtlineare Abbildung, da sie die komplette imaginäre Achse von $-\infty$ bis ∞ auf den Einheitskreis (der Länge 2π) abbildet, bzw. die linke komplexe Ebene innerhalb des Einheitskreises abbildet. Aus diesem Grund müssen die diskreten Frequenzen auch auf eine nicht-lineare Art und Weise in eine kontinuierliche Frequenz

übergeführt werden. Dieser Vorgang wird *Pre-Warping* genannt. Er geschieht nach der Formel:

$$\Omega = \frac{2}{T} \arctan\left(\frac{\omega}{2}\right) \tag{10}$$

Für die Werte aus Gleichungen 9 entspricht dies (mit $T = 1$):

$$\begin{aligned} \text{Für } \omega_1\text{:} \quad & \Omega_{1_W} = 2 \arctan\left(\frac{\pi}{50}\right) \approx 0.1258 \\ \text{Für } \omega_2\text{:} \quad & \Omega_{2_W} = 2 \arctan\left(\frac{\pi}{10}\right) \approx 0.6498 \end{aligned} \tag{11}$$

2.3 Entwurf eines analogen Filters

Ein Butterworthfilter ist die Verallgemeinerung des bekannten RC-Tiefpasses auf höhere Ordnungen. Er hat den Vorteil dass er weder im Durchlassbereich noch im Sperrbereich eine Welligkeit aufweist und im Zwischenbereich monoton ist. Nachteilhaft ist, dass die Steilheit nicht so hoch ist, wie bei anderen Filtern (Tchebeycheff, Cauer, ...). Je steiler die Flanke sein soll, desto höher muss die Ordnung sein (desto größer die Polynome, desto größer der Rechenaufwand und desto größer die Gruppenlaufzeit in einer zeitdiskreten Implementierung).

Die Ordnung des Butterworthfilters kann anhand der Spezifikationen (Dämpfungen im Durchlass- und Sperrbereich sowie zwei Eckfrequenzen) bestimmt werden und errechnet sich lt. Formel im Skriptum mit:

$$n = \frac{\log\left(\frac{10^{\frac{-K_1}{10}} - 1}{10^{\frac{-K_2}{10}} - 1}\right)}{2 \log \frac{\Omega_1}{\Omega_2}} \tag{12}$$

Um einen analogen Filter zu bauen, setzt man die gewünschten Spezifikationen in Gleichung 12 ein:

- $\Omega_1 = 1\,\text{Hz}$
- $\Omega_2 = 5\,\text{Hz}$
- $K_1 = -1\,\text{dB}$ (maximale Dämpfung im Durchlassbereich)
- $K_2 = -35\,\text{dB}$ (minimale Dämpfung im Sperrbereich)

Eingesetzt im Gleichung 12 ergibt dies:

$$n = \frac{\log\left(\frac{10^{\frac{1}{10}} - 1}{10^{\frac{35}{10}} - 1}\right)}{2 \log \frac{1}{5}} \approx 2.9234$$

Sinnvollerweise ist die Ordnung eine ganze Zahl, also runde ich auf 3 auf. Dementsprechend gilt für meine Ordnung: $N = 3$.

Um den Filter nun vollständig zu spezifizieren, muss noch die Grenzfrequenz bestimmt werden. Dies geschieht nach einer der folgenden Formeln:

$$\begin{aligned}\Omega_c &= \frac{\Omega_1}{\sqrt{\left(10^{-\frac{K_1}{10}}-1\right)^{\frac{1}{n}}}} \\ \Omega_c &= \frac{\Omega_2}{\sqrt{\left(10^{-\frac{K_2}{10}}-1\right)^{\frac{1}{n}}}}\end{aligned} \tag{13}$$

Ich wähle die erste Variante aus Gleichung 13 und erhalte als Grenzfrequenz:

$$\Omega_c = \frac{1}{\sqrt{(10^{\frac{1}{10}}-1)^{\frac{1}{2.9234}}}} \approx 7.9168\,[\text{Hz}]$$

Schließlich kann der Butterworth-Filter entworfen werden. Dazu kann entweder das Butterworthpolynom (ausmultipliziert) gebildet werden und und die Koeffizienten aus einer Tabelle abgelesen werden (Skriptum) oder aber die Koeffizienten nach einer bestimmten Vorschrift selbst ermittelt werden.

Ich wähle den zweiten Weg. Die allgemein Übertragungsfunktion des normierten Butterworthfilters (d.h. mit Grenzfrequenz bei 1 Hz) lautet:

$$H(s) = \frac{A_0}{\prod_i (1 + a_i s + b_i s^2)}$$

Dabei gibt A_0 die Verstärkung an, die in unserem Falle 1 ist. Die Bestimmung der Koeffizienten a_i und b_i erfolgt für ungerade Ordnungen nach folgender Vorschrift (siehe Wikipedia):

$$\begin{aligned} i &= 1 \cdots \frac{(n+1)}{2} \\ a_1 &= 1 \\ b_1 &= 0 \\ a_i &= 2\cos\left(\frac{(n+1)\pi}{2}\right) \\ b_i &= 1 \end{aligned}$$

Für eine Filterordnung von 3 ergibt sich $i = 2$, d.h. es braucht nur der Wert a_2 berechnet werden:

$$a_2 = 2\cos\left(\frac{1\cdot\pi}{2}\right) = 1$$

Somit hat die Übertragungsfunktion die Form:

$$H(s) = \frac{1}{(1+s)(1+s+s^2)}$$

Multipliziert man diese Gleichung aus, so kommt auf die Form:

$$H(s) = \frac{1}{1+2s+2s^2+s^3} \tag{14}$$

Die Koeffizienten 2 und 2 hätten sich auch direkt aus der Koeffiziententabelle (siehe Skriptum) ablesen lassen.

Gleichung 14 ist nun im Laplacebereich. Um nun in den Frequenzbereich zu gelangen, muss man $s = \Omega$ setzen (das ergibt sich direkt aus dem Zusammenhang zwischen Laplace- und Fouriertransformation). Nun soll der Tiefpass noch auf die eigentliche Grenzfrequenz Ω_c normiert werden. Dazu setzt man $\Omega = \frac{\Omega}{\Omega_c}$:

$$H(\Omega) = \frac{1}{1+2\left(\frac{\Omega}{\Omega_c}\right)+2\left(\frac{\Omega}{\Omega_c}\right)^2+\left(\frac{\Omega}{\Omega_c}\right)^3} \tag{15}$$

Der analoge Prototyp des Tiefpassfilters ist nun fertiggestellt. In MATLAB verwende ich folgenden Code, um die Übertragungsfunktion des Filters im Bereich 0 bis 10 Hz darzustellen:

```
W = linspace(0, 20*pi, 10000);
F = W/(2*pi);
Hc = 1./(1 + 2*(j*W/Wc_unwrapped) + ...
2*(j*W/Wc_unwrapped).^2 + (j*W/Wc_unwrapped).^3);
```

Das Resultat ist in Abbildung 9 zu sehen. Anhand der Marker bei 1 Hz und 5 Hz erkennt man, dass die Spezifikationen genau eingehalten werden.

2.4 Der zeitdiskrete Tiefpassfilter

Um nun aus dem analogen Prototypen in Gleichung 15 einen zeitdiskreten Filter zu erstellen, muss dieser mittels der Bilineartransformation transformiert werden. Diese nicht-lineare Transformation stellt einen Zusammenhang zwischen dem s-Bereich (Laplace, kontinuierlich) und dem z-Bereich (zeitdiskret) her:

$$s = \frac{2}{T}\left(\frac{1-z^{-1}}{1+z^{-1}}\right) = \frac{2}{T}\left(\frac{z-1}{z+1}\right) \tag{16}$$

Der Parameter T ist frei wählbar. Ist er bei allen Parametern gleich gewählt, so kürzt er sich heraus und ist somit für das Ergebnis unbedeutend. Ich habe immer $T = 1$ gesetzt.

Problematisch ist jedoch, dass der analoge Prototyp anhand der analogen Frequenzen erzeugt wurde. Deswegen muss die gesamte Prozedur noch einmal wiederholt werden. Um die Ordnung n zu bestimmen, setze ich nun die Werte aus Gleichungen 11 in Gleichung 12 ein:

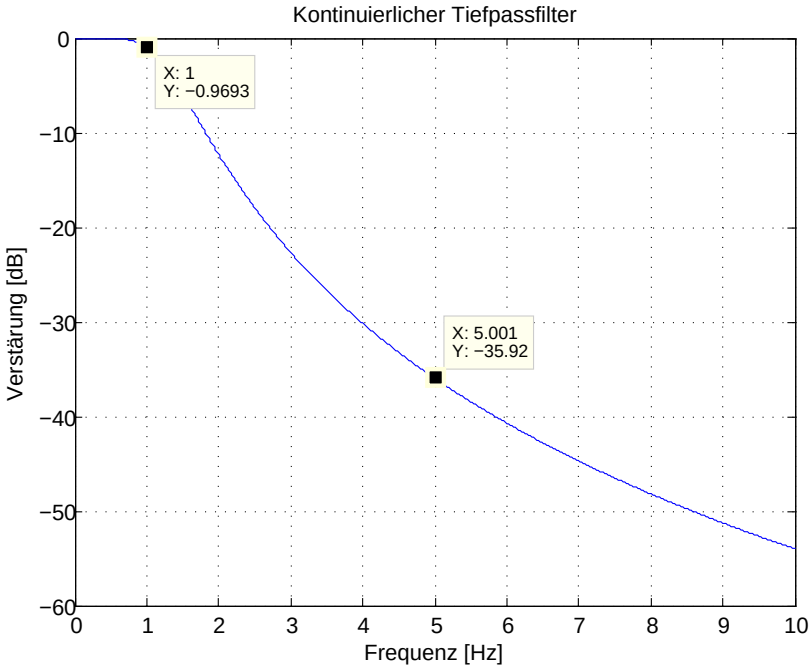


Abbildung 9: Übertragungsfunktion des analogen Prototypen

$$n = \frac{\log\left(\frac{10^{\frac{1}{10}}-1}{10^{\frac{35}{10}}-1}\right)}{2\log\frac{0.1258}{0.6498}} \approx 2.8657$$

Aufgerundet ergibt dies $N = 3$, die Ordnung bleibt also glücklicherweise gleich und der Prototyp aus Gleichung 15 kann verwendet werden. Lediglich die Grenzfrequenz muss nun nach Gleichung 13 neu bestimmt werden. Das ergibt:

$$\Omega_c = \frac{0.6498}{\sqrt{(10^{\frac{1}{10}}-1)^{\frac{1}{2.8657}}}} \approx 0.1593 \tag{17}$$

Für die Anwendung der Bilineartransformation ist nach Gleichung 16 wieder eine Darstellung im Laplacebereich zweckmäßig. Deswegen setze ich abermals $\Omega = s$ in Gleichung 15 und erhalte den entnormierten Tiefpass im s-Bereich:

$$H(s) = \frac{1}{1 + 2\left(\frac{s}{\Omega_c}\right) + 2\left(\frac{s}{\Omega_c}\right)^2 + \left(\frac{s}{\Omega_c}\right)^3} \tag{18}$$

Nun setzt man in Gleichung 18 die Bilineartransformation aus Gleichung 16 ein und erhält (um unnötige Bruchstriche zu vermeiden stelle ich die inverse Übertragungsfunktion dar):

$$\begin{aligned}\frac{1}{H(z)} &= 1 + \frac{4}{T\Omega_c}\left(\frac{z-1}{z+1}\right) + \frac{8}{T^2\Omega_c^2}\left(\frac{z-1}{z+1}\right)^2 + \frac{8}{T^3\Omega_c^3}\left(\frac{z-1}{z+1}\right)^3 \\ &= \frac{(z+1)^3}{(z+1)^3} + \frac{\frac{4}{\Omega_c T}(z-1)(z+1)^2}{(z+1)^3} + \frac{\frac{8}{\Omega_c^2 T^2}(z-1)^2(z+1)}{(z+1)^3} + \frac{\frac{8}{\Omega_c^3 T^3}(z-1)^3}{(z+1)^3}\end{aligned}$$

Fasst man diese Brüche mit gemeinsamen Nenner zusammen, so folgt:

$$H(z) = \frac{P(z)}{Q(z)} = \frac{(z+1)^3}{(z+1)^3 + \frac{4}{\Omega_c T}(z-1)(z+1)^2 + \frac{8}{\Omega_c^2 T^2}(z-1)^2(z+1) + \frac{8}{\Omega_c^3 T^3}(z-1)^3}$$

Multipliziert man die Polynome nun aus, so erhält man für die Nullstellen:

$$P(z) = z^3 + 3z^2 + 3z + 1$$

Und für die Pole gilt:

$$\begin{aligned}Q(z) &= z^3 + 3z^3 + 3z + 1 \\ &\quad + \frac{4}{\Omega_c T}(z^3 + z^2 - z - 1) \\ &\quad + \frac{8}{\Omega_c^2 T^2}(z^3 - z^2 - z + 1) \\ &\quad + \frac{8}{\Omega_c^3 T^3}(z^3 - 3z^2 + 3z - 1) \\ &= \underbrace{\left(1 + \frac{4}{\Omega_c T} + \frac{8}{\Omega_c^2 T^2} + \frac{8}{\Omega_c^3 T^3}\right)}_{b_0} z^3 \\ &\quad + \underbrace{\left(3 + \frac{4}{\Omega_c T} - \frac{8}{\Omega_c^2 T^2} - \frac{24}{\Omega_c^3 T^3}\right)}_{b_2} z^2 \\ &\quad + \underbrace{\left(3 - \frac{4}{\Omega_c T} - \frac{8}{\Omega_c^2 T^2} + \frac{24}{\Omega_c^3 T^3}\right)}_{b_2} z \\ &\quad + \underbrace{\left(1 - \frac{4}{\Omega_c T} + \frac{8}{\Omega_c^2 T^2} - \frac{8}{\Omega_c^3 T^3}\right)}_{b_3}\end{aligned}$$

Setzt man das Polpolynom und das Nullstellenpolynom wieder zusammen, ergibt sich als Übertragungsfunktion die Form:

$$H(z) = \frac{z^3 + 3z^2 + 3z + 1}{a_1 z^3 + a_2 z^2 + a_3 z + a_4} \tag{19}$$

$$= \frac{1 + 3z^{-1} + 3z^{-2} + z^{-3}}{a_0 + a_1 z^{-1} + a_2 z^{-3} + a_3 z^{-3}} \tag{20}$$

mit den Koeffizienten ($T = 1$, $\Omega_c = 0.1593$ (siehe Gleichung 17)):

$$\begin{aligned} \mathbf{b} &= (1, 3, 3, 1) \\ \mathbf{a} &= 10^3 \cdot (2.3211, -6.2261, 5.6015, -1.6884) \end{aligned} \tag{21}$$

In MATLAB wurden die Koeffizienten wie folgt implementiert:

```
B = [1, 3, 3, 1];
A = [ 1 + 4/(Wc * T) + 8/(Wc^2 * T^2) +  8/(Wc^3 * T^3) , ...
      3 + 4/(Wc * T) - 8/(Wc^2 * T^2) - 24/(Wc^3 * T^3) , ...
      3 - 4/(Wc * T) - 8/(Wc^2 * T^2) + 24/(Wc^3 * T^3) , ...
      1 - 4/(Wc * T) + 8/(Wc^2 * T^2) -  8/(Wc^3 * T^3) ];
```

Und mit folgendem Code wurde die Übertragungsfunktion für $\omega = \{0, \ldots, \pi\}$ bestimmt (da das Spektrum ohnehin symmetrisch ist, habe ich es nur bis π bestimmt):

```
w = linspace(0, pi, 1000);
f = w*(Fs/2)/pi;
z = exp(j*w);
H = (B * [z.^0 ; z.^-1 ; z.^-2 ; z.^-3]) ./ ...
(A * [z.^0 ; z.^-1 ; z.^-2 ; z.^-3]);
```

Der Betragsfrequenzgang des fertigen zeitdiskreten Filters ist in Abbildung 10 zu sehen. Abermals ist anhand der Marker zu sehen dass die Spezifikationen eingehalten wurden:

- Bei 1 Hz gibt es eine Dämpfung von 0,05 dB, maximal wären 1 dB erlaubt
- Bei 5 Hz gibt es eine Dämpfung von über 36 dB, hier wäre ein Minimalwert von 35 dB erlaubt

Diese Spezifikationen lassen sich übrigens verbessern, indem man niedrigere Werte für die maximal zulässige Dämpfung im Durchlassbereich bzw. höhere Werte für die minimal zulässige Dämpfung im Sperrbereich wählt. Allerdings erhöht sich damit auch die Filterordnung und dadurch die Komplexität des Filters (und dadurch die Gruppenlaufzeit).

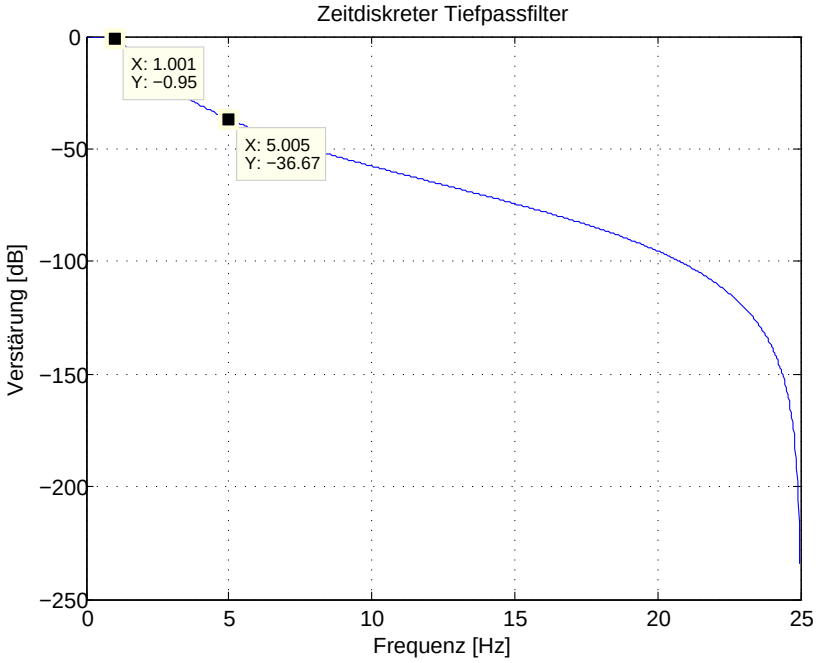


Abbildung 10: Übertragungsfunktion des zeitdiskreten Filters

2.5 Test des Filters

Um den Tiefpassfilter zu testen, habe ich das Signal aus der Angabe (Gleichung 7) in MATLAB erstellt und filtere es mit meinem Filter:

```
t = 0:(1/Fs):10-1/Fs;
x = 5*sin(2*pi*F1*t) + 2*sin(2*pi*F2*t);
y = filter(B, A, x);
```

Das Ergebnis ist in Abbildung 11 und 12 im Zeit- sowie im Frequenzbereich zu sehen: Die Störfrequenz wird sehr gut herausgefiltert. Die Länge des Zeitvektors wurde so gewählt dass kein Leakage auftreten kann. Beim Ausgangssignal ist jedoch ein Leakageeffekt zu erkennen (siehe Abbildung 12). Dieser lässt sich dadurch erklären, dass sich durch die Filterung das Eingangssignal sprunghaft ändert. Dadurch entstehen parasitäre Frequenzen bei der Filterung.

2.6 Übertragungsfunktion und Differenzengleichung

Die Übertragungsfunktion kann direkt aus Gleichung 20 abgelesen werden. Setzt man noch die Koeffizienten aus Gleichung 21, so erhält man:

$$\boxed{H(z) = \frac{1 + 3z^{-1} + 3z^{-2} + z^{-3}}{2321.1 - 6226.1z^{-1} + 5601.5z^{-3} - 1688.4z^{-3}}}$$

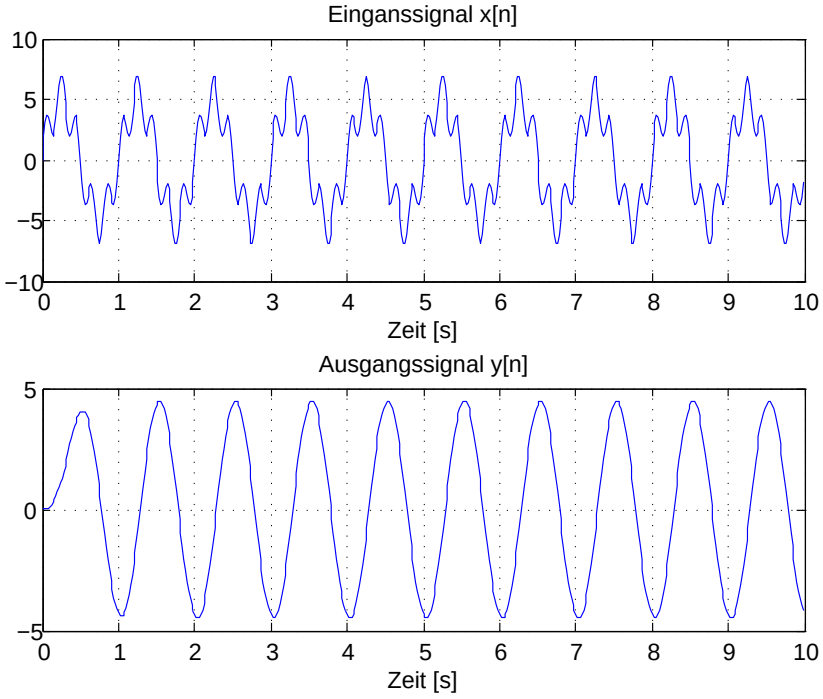


Abbildung 11: Test des Filters im Zeitbereich

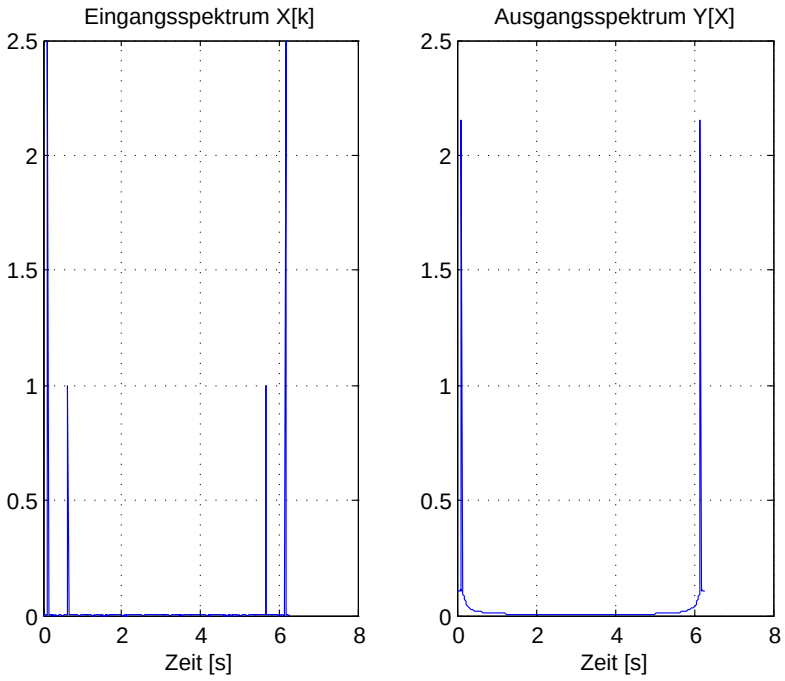


Abbildung 12: Test des Filters im Frequenzbereich

Um die Differenzengleichung zu bestimmen, setzt man in der Übertragungsfunktion $H(z) = Y(z)/X(z)$:

$$\begin{aligned} Y(z)(a_0 + a_1 z^{-1} + a_2 z^{-2} + a_3 z^{-3}) &= X(z)(b_0 + b_1 z^{-1} + b_2 z^{-2} + b_3 z^{-3}) \\ a_0 Y(z) + a_1 Y(z) z^{-1} + a_2 Y(z) z^{-2} + a_3 Y(z) z^{-3} &= b_0 X(z) \\ + b_1 X(z) z^{-1} + b_2 X(z) z^{-2} + b_3 X(z) z^{-3} & \end{aligned}$$

und unterwirft diese der inversen z-Transformation:

$$a_0 y[n] + a_1 y[n-1] + a_2 y[n-2] + a_3 y[n-3] = b_0 x[n] + b_1 x[n-1] + b_2 x[n-2] + b_3 x[n-3]$$

Nun wird diese Gleichung noch nach $y[n]$ aufgelöst:

$$y[n] = \frac{b_0}{a_0} x[n] + \frac{b_1}{a_0} x[n-1] + \frac{b_2}{a_0} x[n-2] + \frac{b_3}{a_0} x[n-3] - \frac{a_1}{a_0} y[n-1] - \frac{a_2}{a_0} y[n-2] - \frac{a_3}{a_0} y[n-3]$$

Werden noch die Koeffizienten aus Gleichung 21 eingesetzt, so erhält man:

$$\boxed{\begin{aligned} y[n] = 0.0004 \cdot x[n] + b_1 \cdot 0.0013 + 0.0013 \cdot x[n-2] + 0.0004 \cdot x[n-3] \\ +2.6824 \cdot y[n-1] - 2.4133 \cdot y[n-2] + 0.7274 \cdot y[n-3] \end{aligned}} \tag{22}$$

Wird der Filterausgang mit der Differenzengleichung direkt realisiert (Direktform 1), so kann man die Anzahl der benötigten Rechenschritte direkt aus Gleichung 22 ablesen:

- 6 Additionen
- 7 Multiplikationen mit einer vordefinierten Konstante

2.7 Der Signalflussgraph

Der Signalflussgraph in Direktform 1 kann direkt aus der Differenzengleichung (Gleichung 22) abgelesen werden. Er ist in Abbildung 13 abgebildet. Ist ist allerdings möglich, den Filter in anderen Strukturen zu realisieren (Direktform 2, Kaskadenform, ...).

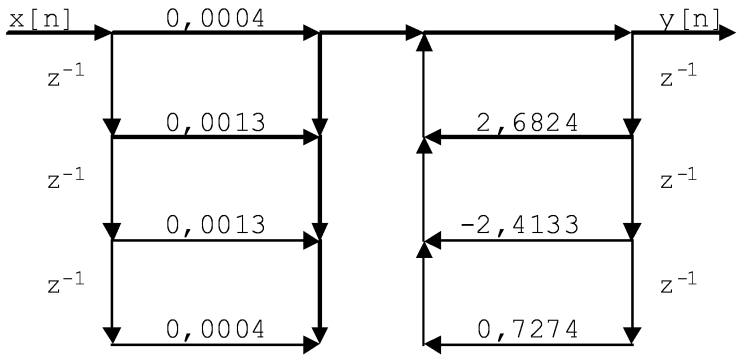


Abbildung 13: Signalflussgraph des Filters