# PDM nach PCM Wandlung – Ergebnisse kurz zusammengefasst.

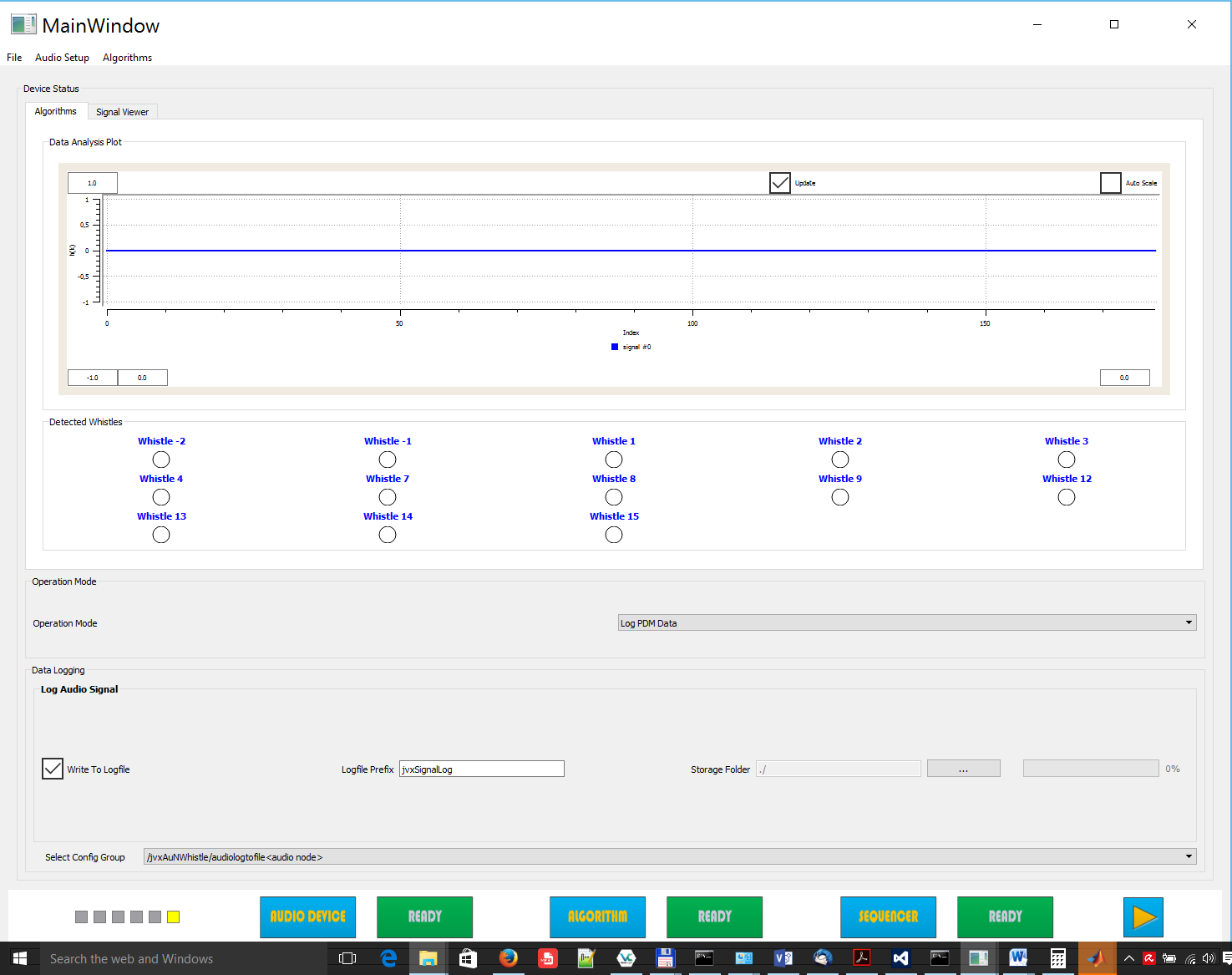
Die Wandlung der PDM Signale nach PCM läuft auf dem BBB ab und soll den FPGA ersetzen. Prinzipiell wird für sowas ein CIC Filter eingesetzt. Dieses ist aber ein eher schlechtes Umtastungsfilter, da es einen hochfrequenten Roll-Off und keine konstante Aliasing-Unterdrückung hat. Ich könnte gut damit leben, WENN nicht auch die Implementierung auf einem DSP schwierig wäre, da die PDM Werte (+-1) in 16 Bit Variablen abgelegt werden und für die Filterung Bit für Bit verarbeitet werden müssten. Diese Bit-Schiebe und -Extraktions-Aufgabe kostet zuviel Rechenkapazität auf dem DSP.

Die Alternative stellt ein FIR Design dar, bei dem die gesammelten PDM Bits in Bytes über Lookup-Tables gefiltert werden. Aus Effizienzgründen ist das Filter in Polyphasenstruktur umgesetzt und nutzt aus, dass die Impulsantwort symmetrisch ist. Es stellt sich die Frage, welcher Ansatz besser ist.

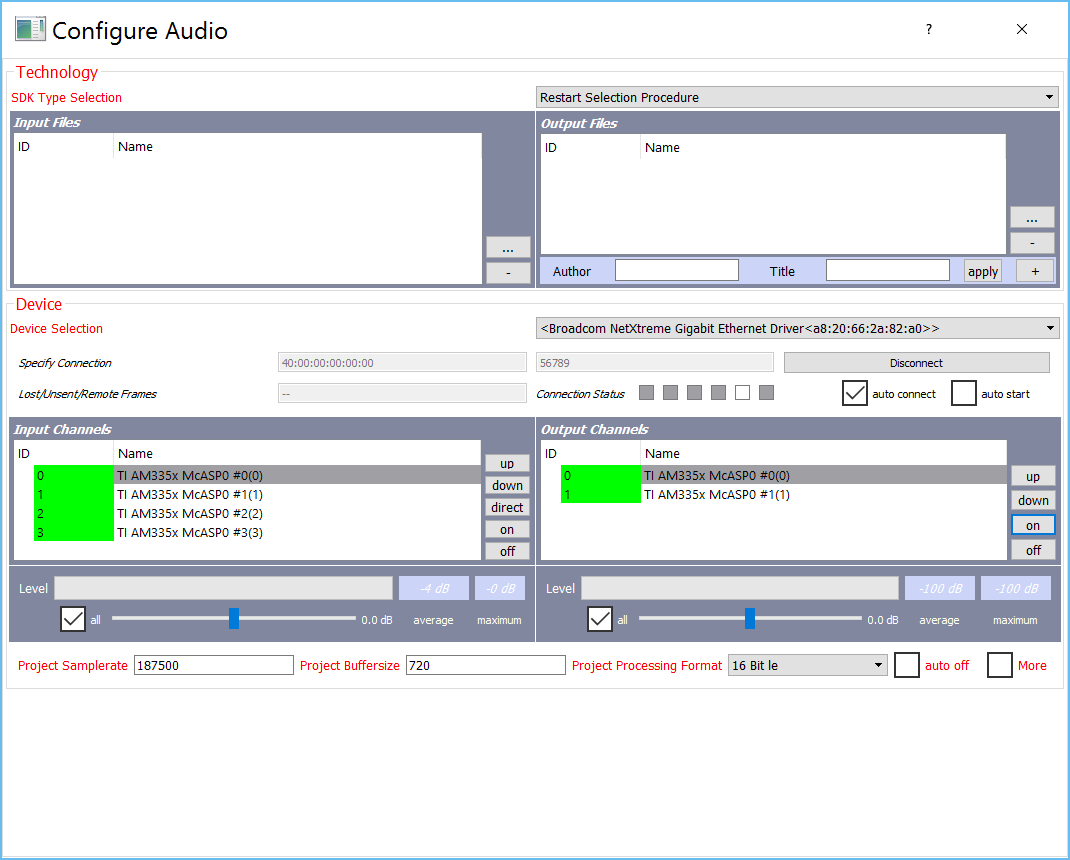
Beide Varianten wurden in Matlab als Referenz implementiert und evaluiert. Hier sind die Ergebnisse kurz zusammengefasst.

## Vorgehen

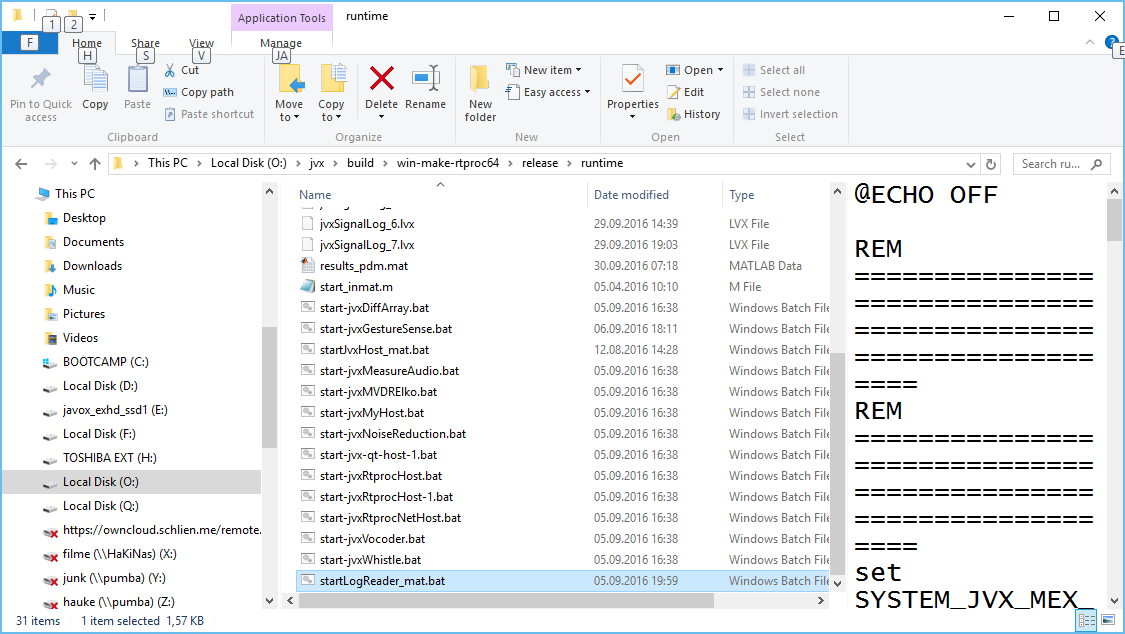
Die PDM Daten werden in den PC über den BBB mittels Raw-Socket Verbindung eingegeben. Der BBB kann die Daten bereits mittels CIC Filter in PCM Daten wandeln und dann als 16 BIT PCM übertragen. Über die Konfiguration des BBB kann aber auch erreicht werden, dass die PDM Daten direkt an den PC versendet werden. In diesem Fall stehen in den auf dem PC eingehenden 16 BIT Werten die Bits des PDM Signals in der Reihenfolge „LSB-First“ zur Verfügung. In der „Whistle“ Anwendung, die mit dem Raw-Socket BBB umgehen kann, steht ein Mechanismus zur Verfügung, die anfallenden Daten in einem Logfile zu speichern. Hierzu muss die Anwendung gestartet und dann korrekt eingestellt werden:



Wenn man nun das Programm verbindet und dann startet, wird eine Logdatei geschrieben, die z.B. jvxSignalLog\_4.jvx heißt. Diese beinhaltet alle 6 Kanäle plus einen siebten Referenzkanal, der ein aus dem PDM Signal des ersten eingehenden Kanal erzeugtes PCM Signal beinhaltet. Nachdem das Logfile geschrieben wurde, kann die Aufnahmeanwendung direkt beendet werden. Zu beachten ist, dass keine Dropouts bei der Aufnahme auftreten. Dies kann kontrolliert werden, indem im Device Kontroll-UI der Wert der verlorenen Frames kontrolliert wird:



Das weitere Vorgehen geschieht in Matlab. Dazu wird Matlab mit dem Script startLogReader\_mat gestartet:



Daten werden aus dem Logfile wie folgt gelesen:

content]=jvxDataLogReader.jvx\_read\_file('');

for(ix=1:4)

sig(ix,:)=reshape(content{1}.data{ix}.data',[1, prod(size(content{1}.data{ix}.data))]);

end

Nun stehen die 4 aufgenommenen Kanäle in Variable sig zur Verfügung. Drei Varianten der PDM-nach-PCM-Wandlung habe ich durch folgenden Aufruf getestet:

sig\_cic = examples.jvx\_pdm\_run(sig(1,:), 187500, false);

sig\_fir\_32 = examples.jvx\_pdm\_run(sig(1,:), 187500, true, 32);

sig\_fir\_4 = examples.jvx\_pdm\_run(sig(1,:), 187500, true, 4);

Das erste ist das CIC Filter, die zweite Variante ist ein FIR Filter mit 32 Taps im unterabgetasteten Bereich (187.5 kHz, 64 Read+Add pro Sample), und die dritte Variante ist so wie die zweite, jedoch mit einem extrem kurzen Filter (4 Taps im unterabgetasteten Bereich, somit 8 Read & Add pro Sample).

Wenn alle durch sind, kann man die Daten auf Festplatte speichern:

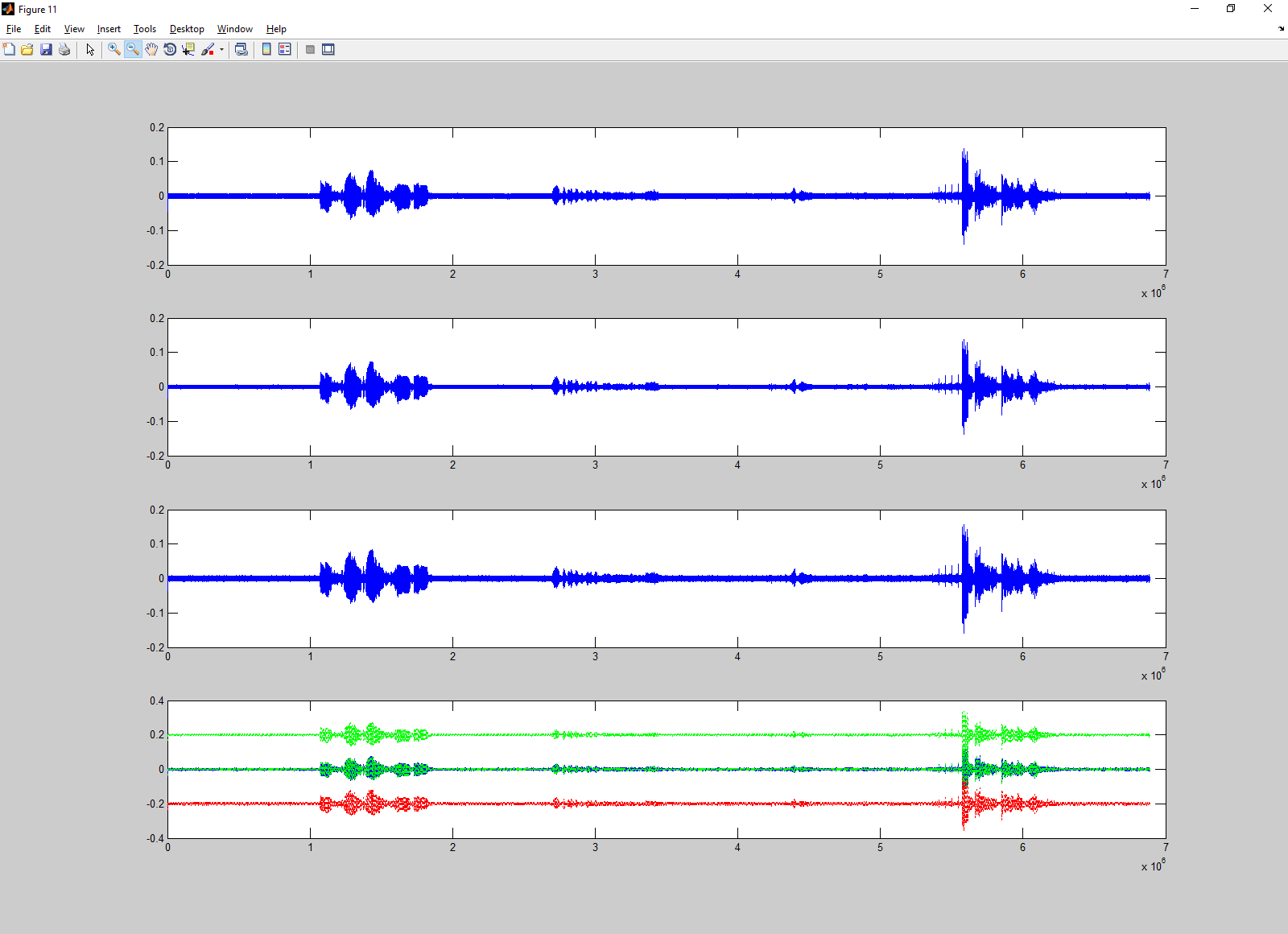
results.sig\_cic = sig\_cic; results.sig\_fir\_32 = sig\_fir\_32; results.sig\_fir\_4 =sig\_fir\_4;

save('results\_pdm.mat\*, 'results');

Zum Anhören geht man wie folgt vor:

p\_cic = audioplayer(resample(sig\_cic,1,4)\*10, 48000); play(p\_cic)

Dabei wird das Signal zunächst in den hörbaren Bereich runtergesampled und dann bei 48 kHz abgespielt. Verstärkungsfaktor 10 dient dazu, das Hintergrundrauschen besser hören zu können. Eine Darstellung der drei Signale zeigt, dass die erreichten Ergebnisse in etwas ähnlich sind:



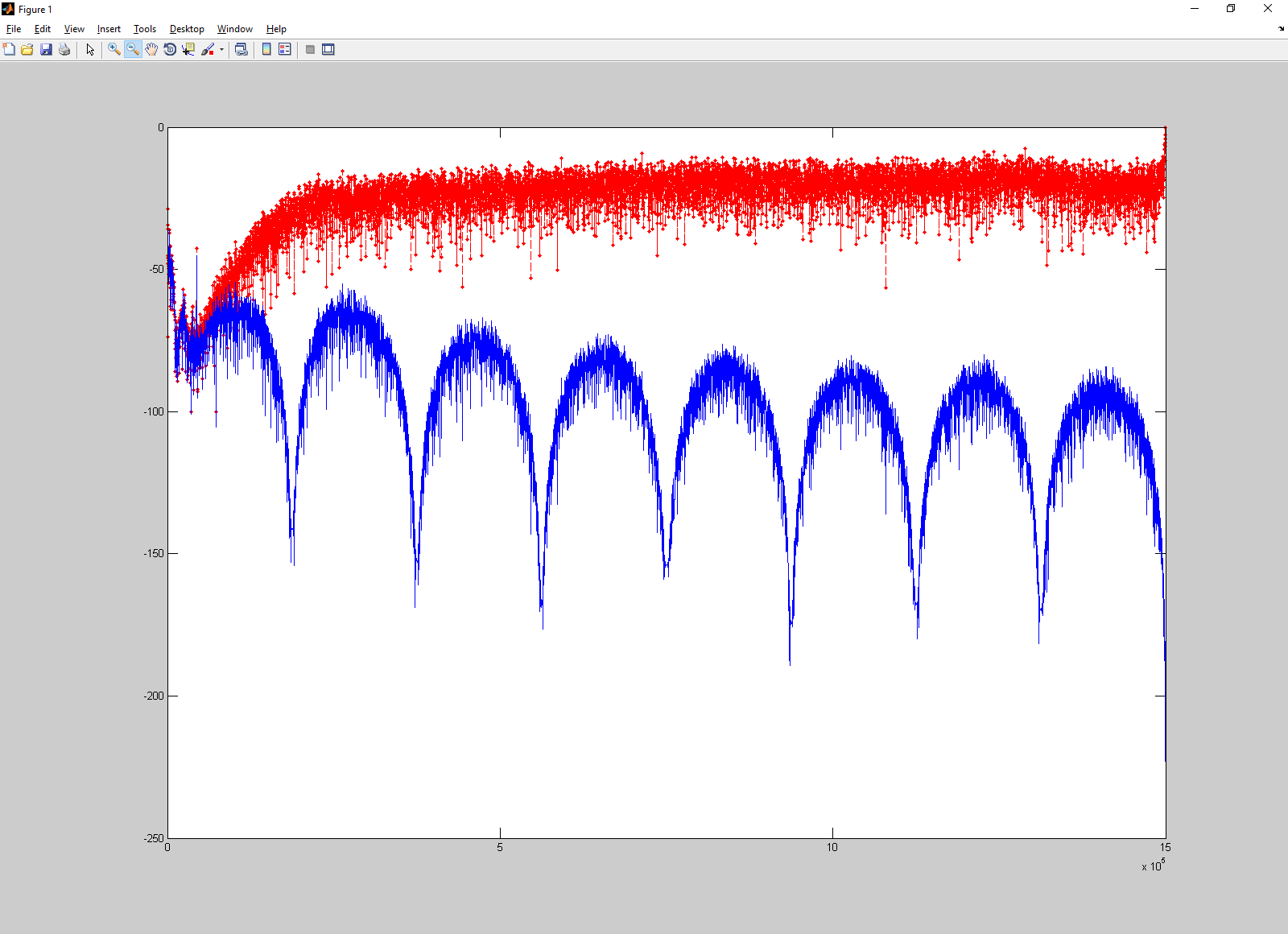
FIR Filter Länge 4

FIR Filter Länge 32

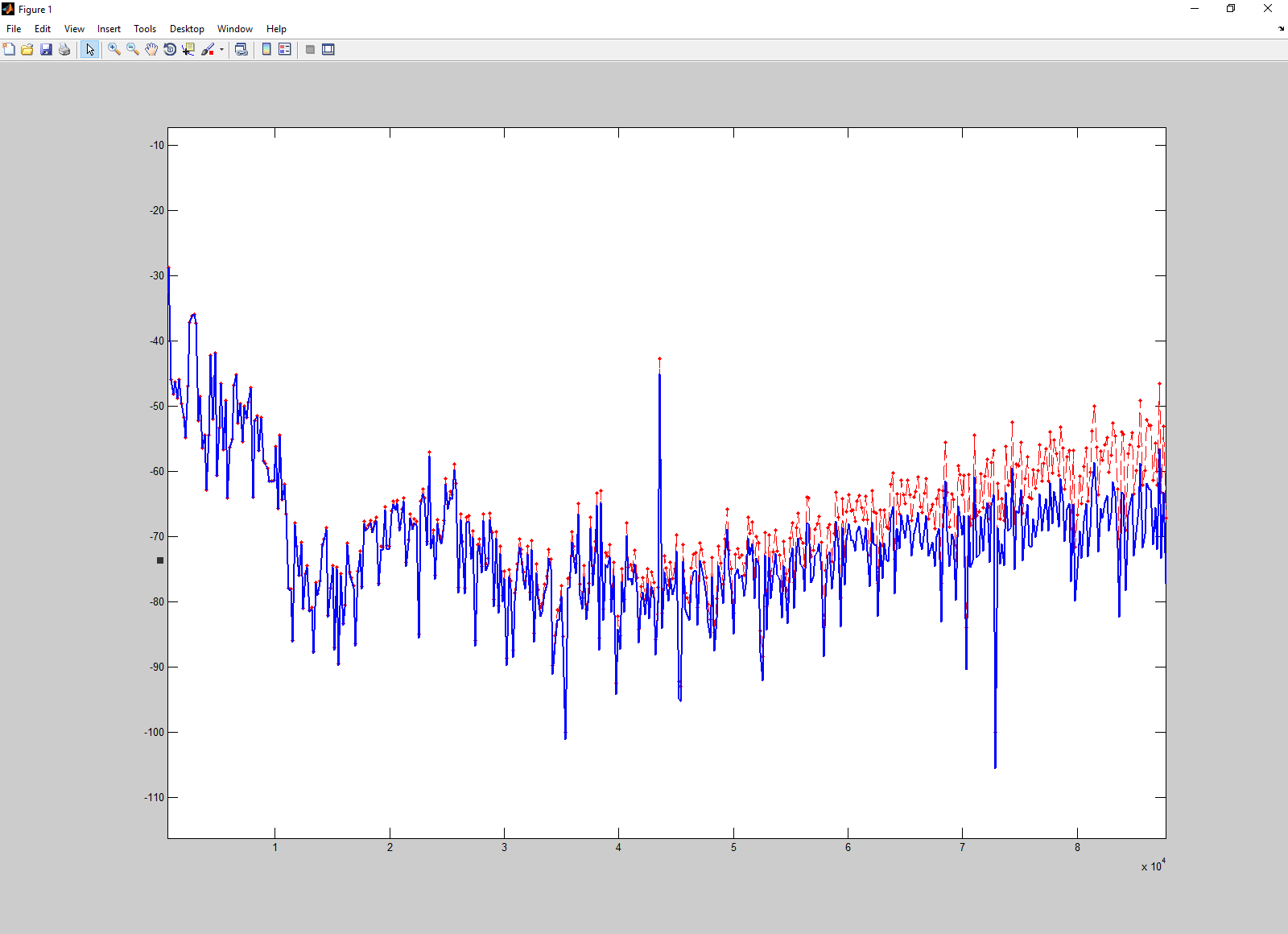
CIC Filter Ordnung 3

Von der reinen Anschauung liegt der Ansatz FIR 32 vorne, da das Rauschen minimal aussieht. Im Detail kann man die Wirkung der Signalverarbeitung gut im Frequenzbereich einschätzen. Wichtiger Punkt ist das Spektrum des PDM Signals und daraus abgeleitet das Spektrum des überabgetasteten Signals nach Tiefpassfilterung.

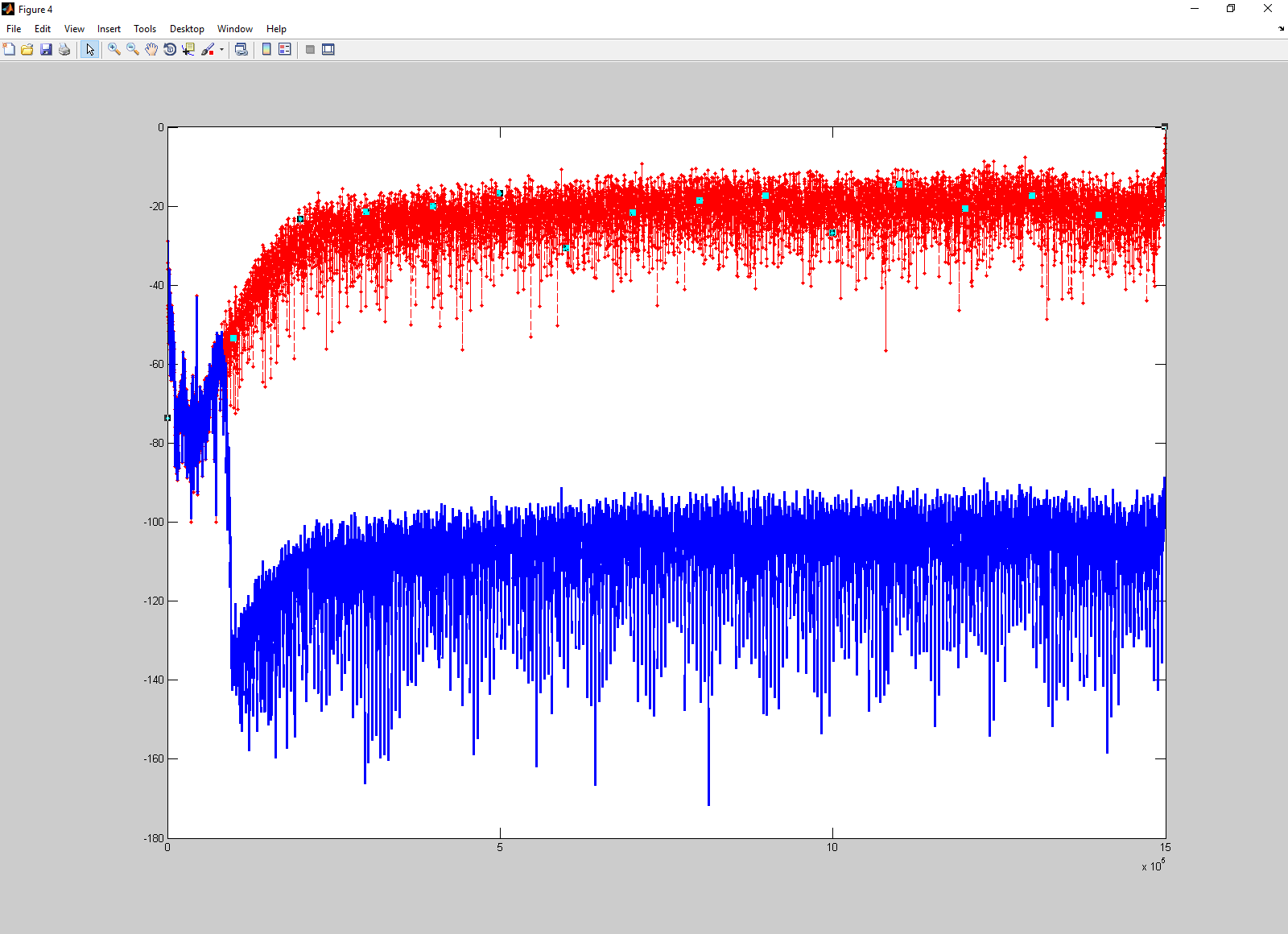
Das folgende Diagramm zeigt die Auswertung des CIC Filters:



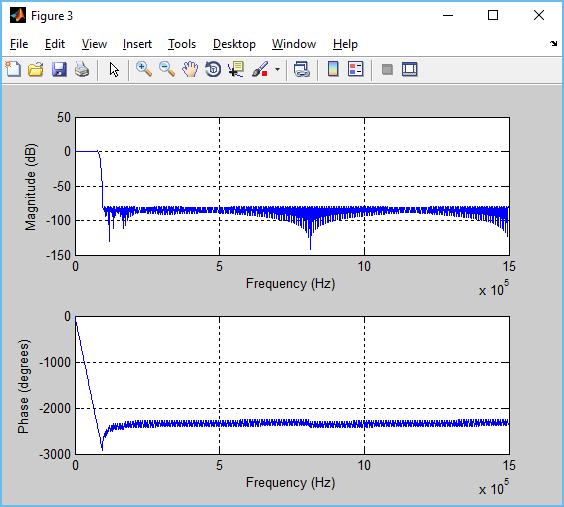
Man erkennt in rot das Spektrum des Eingangssignals sowie in blau das Spektrum des Signals nach CIC Tiefpassfilterung – ohne kritische Unterabtastung. Der Nutzbereich, der unser Ultraschallsignal beschreibt, ist durch das grüne Quadrat hervorgehoben. Wenn wir reinzoomen, sehen wir im Detail:

Der Roll-Off beginnt bei 20 khz, eine Sperrdämpfung ist bei der Nypquistfrequenz nach Unterabtastung (187500/2 = 93750 Hz) noch nicht wirklich dolle. Im hörbaren Bereich ist aber alles ok.

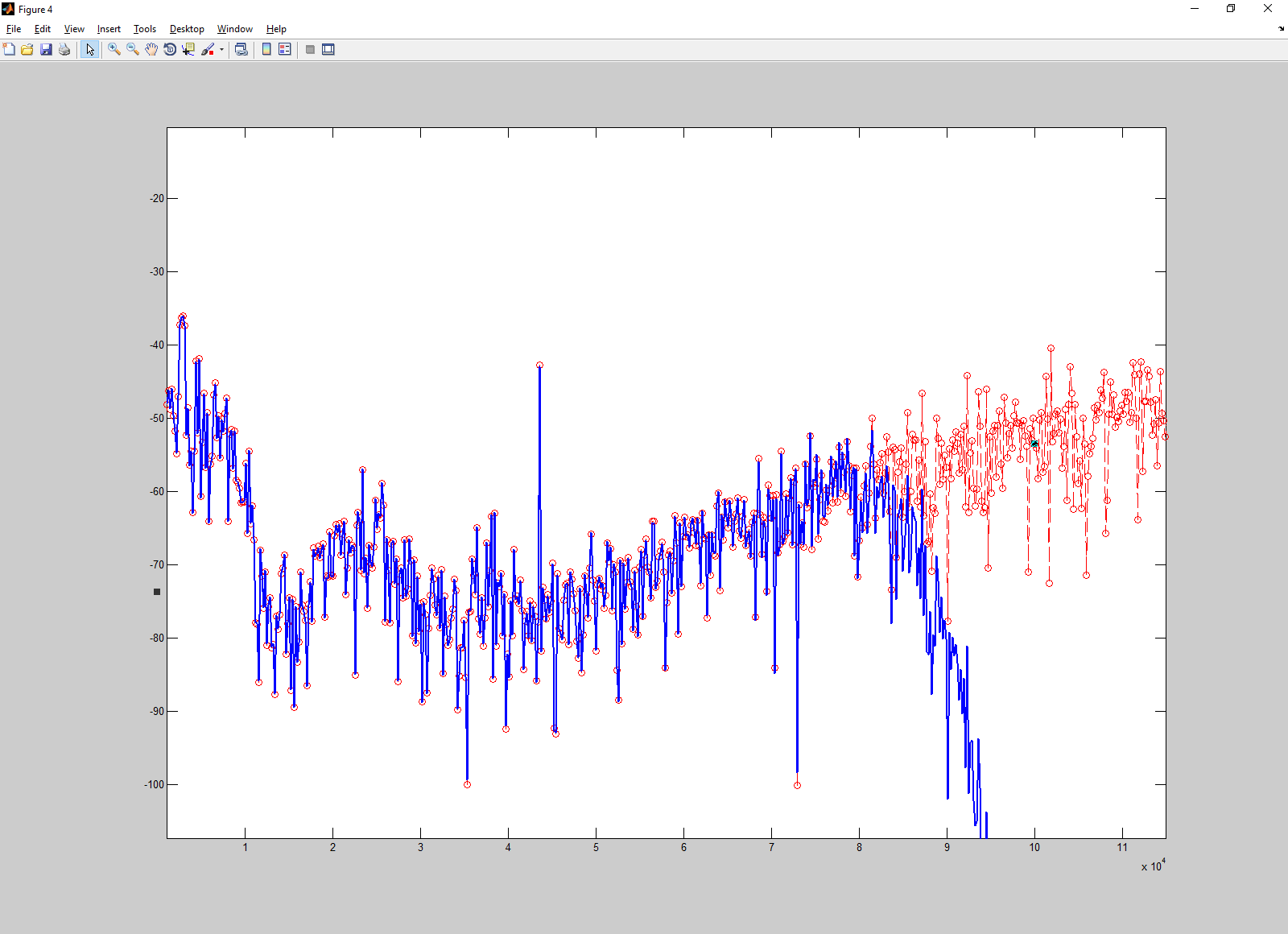
Bei langen FIR Filter sieht es wie folgt aus:



Die Dämpfung liegt an dem sehr stark dämpfenden Filter,

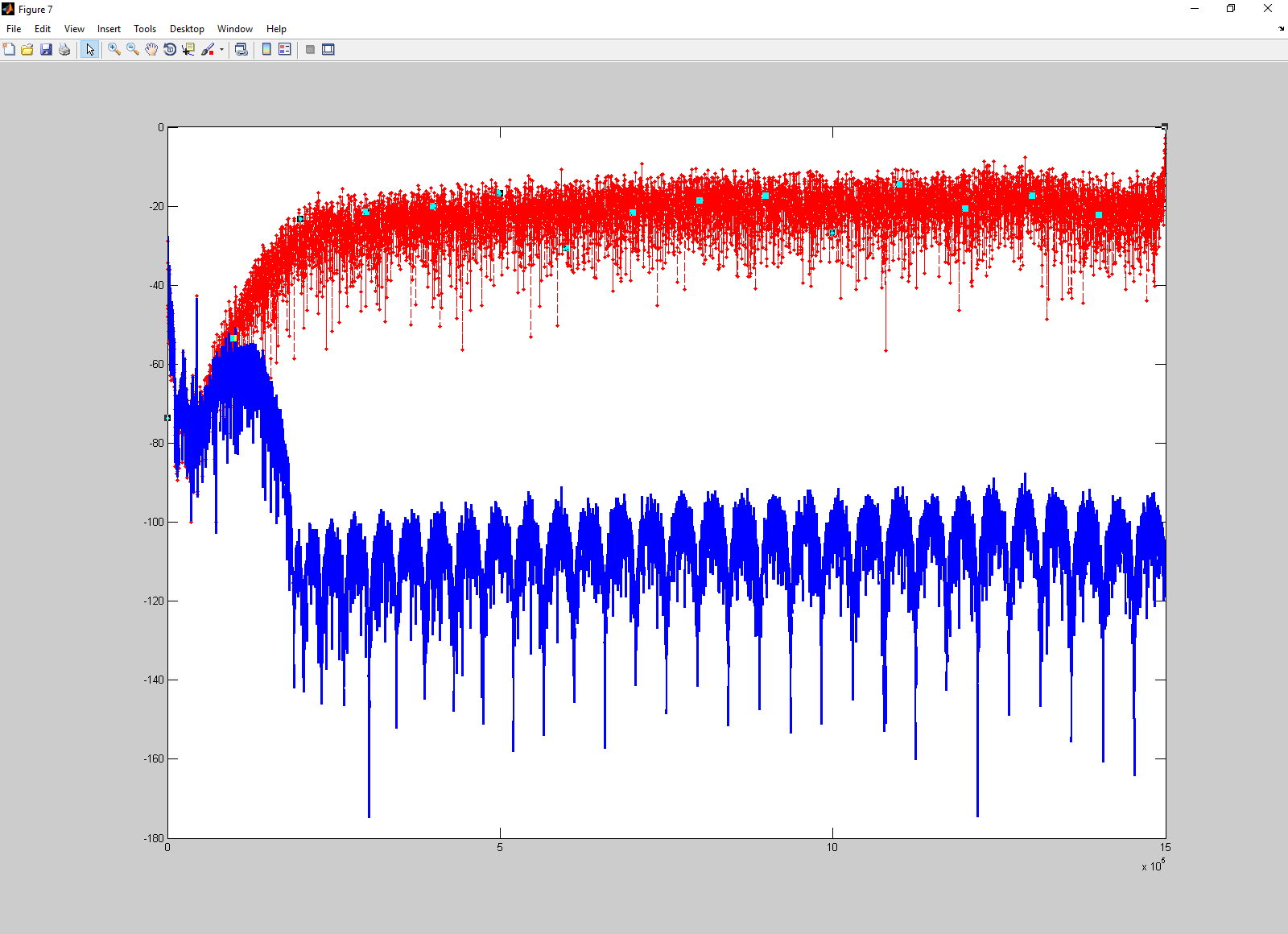


Entsprechend sind keine Aliasing-Bestandteile im Signal vor kritischer Unterabtastung gegeben. Im Detail sieht der interessante Bereich wie folgt aus:

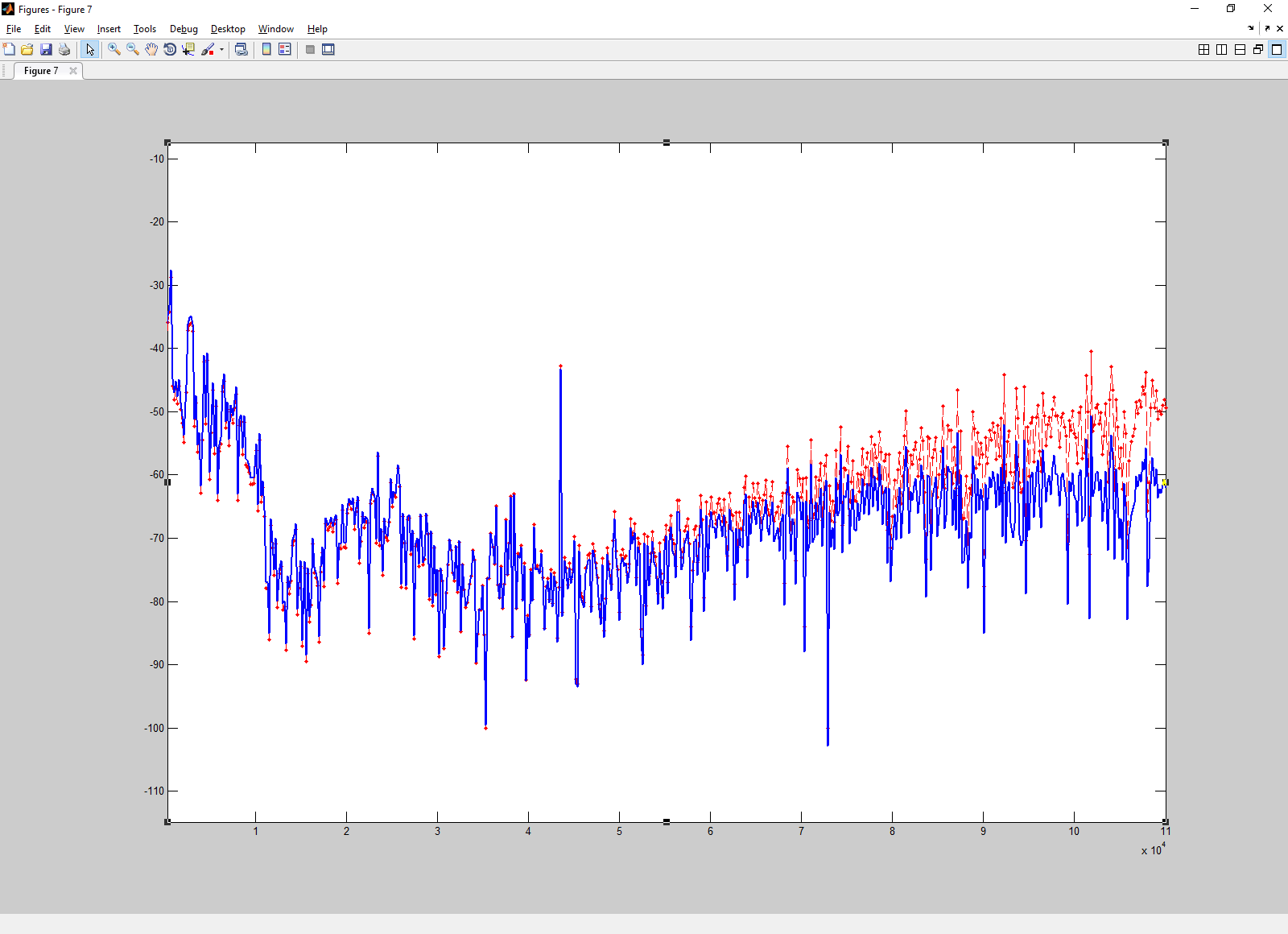


So gut wie kein Aliasing, Roll-Off erst ab ca. 80 kHz.

Beim stark verkürzten FIR Filter sieht es wie folgt aus:



Klar ist die Dämpfung weniger stark und auch ein Roll-Off vorhanden. Im Detail sieht das wie folgt aus:



Roll-Off beginnt bei ca. 40 kHz, Aliasing ist zu erwarten. Es sieht insgesamt ähnlich wie beim CIC Filter aus, jedoch mit etwas besseren Eigenschaften. Dieser Ansatz hat eine Komplexität von ca. 8 Read & Add pro Sample bei 187.500 Hz Abtastrate. Je nach Verfügbarkeit des Speichers sollte das gut machbar sein.