← LAB ◊ ↑ TOC ↑

Name, Vorname:

LAB 3: Filterentwurf mit pyfda

In diesem Versuch

Inhalt

0. Allgemeine Hinweise

0.1 Jupyter Notebook

- \<SHIFT>-\<RETURN> führt eine Codezelle aus und rendert eine Textzelle.
- In Markdown sind Leerzeilen wichtig zum Trennen von Abschnitten!
- Sie können LaTeX-Code zwischen $ ... \$ einschließen.
- Kontexthilfe zu Funktionen etc. bekommen Sie über \<SHIFT>-\<TAB>

Abbildungen in diesem Notebook wurden konvertiert mit https://www.base64-image.de/ und in den HTML-Code eingebettet.

0.2 Installation von pyfda

Für diesen Versuch benötigen Sie die Software **pyfda** (Python Filter Design and Analysis), die Sie per `pip install pyfda` installieren können (wenn Sie eine Python-Installation auf Ihrem Rechner haben), alternativ gibt es unter https://github.com/chipmuenk/pyfda Binaries für Windows und für Ubuntu 20.10 (funktioniert vermutlich auch mit anderen halbwegs aktuellen Distros).

0.3 Abgabe

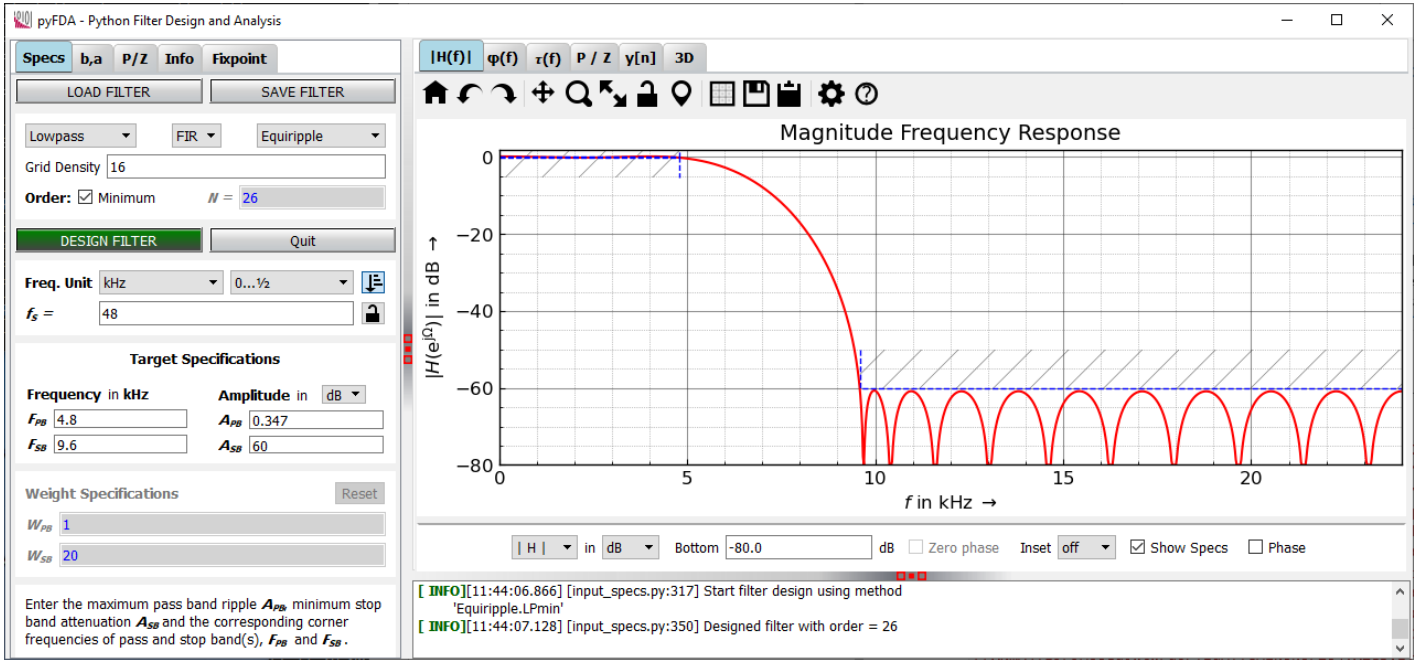
Nach dem Praktikumsversuch exportieren Sie das Notebook mit Textantworten, Codezellen und Plots als HTML (File -> Export Notebook As ... -> Export Notebook to HTML) und reichen es in Moodle ein.

In [ ]:

```
import os, sys
module_path = os.path.abspath(os.path.join('..')) # append directory one level up to import path
if module_path not in sys.path:  # ... if it hasn't been appended already
    sys.path.append(module_path)
import dsp_fpga_lib as dsp
dsp.versions() # print versions

%matplotlib inline
import matplotlib.pyplot as plt
size = {"figsize":(12,5)} # Plotgröße in Inch

import numpy as np
import scipy.signal as sig
import wave

from IPython.display import Audio, display
#------------------------------------------------------------------------
def wav2np(filename):
    """ Read the wav-file and convert it to a one or two-dimensional numpy array,
        depending on the number of channels.
        Properties of the WAV-file are stored as function attributes (evil)
    """
    wf = wave.open(filename,'rb')
    wav2np.N_CH = wf.getnchannels() # number of channels
    wav2np.W    = wf.getsampwidth() # wordlength per sample in bytes
    wav2np.N_FR = wf.getnframes()   # number of frames
    wav2np.f_S  = wf.getframerate() # sample (frame) rate

    print("{0} channels with {1} fnames of {2} bytes and f_S = {3} Hz ".format(wav2np.N_CH, wav2np.N_FR, wav2np.W, wav2np.f_S)
```

```
    print('{0} channels with {1} frames of {2} bytes and f_S = {3} Hz.'.format(wav2np.N_CH, wav2np.N_FR, wav2np.W, wav2np.f_S))

    if wav2np.W == 2:
        samples_in = np.frombuffer(wf.readframes(-1), dtype=np.int16) # read wav data as 16 bit integers, R and L samples are
    elif wav2np.W == 1:
        samples_in = np.frombuffer(wf.readframes(-1), dtype=np.int8) # read wav data as 8 bit integers, R and L samples are in
    else:
         raise TypeError("Unknown data format: {0} bytes".format(wav2np.W))

    samples = np.array([samples_in[idx::wav2np.N_CH] for idx in range(wav2np.N_CH)], dtype=np.int32) # deinterleave channels t
    return samples
```

1. Überblick über pyfda



Auf der linken Seite wählen und entwerfen Sie Filter unter den Tabs "Specs" (Filterentwurf), "b,a" (Eingabe von Filterkoeffizienten), "P/Z" (Eingabe von Polen und Nullstellen). Auf der rechten Seite schauen Sie sich Betrags- und Phasengang, Gruppenlaufzeit, P/N Diagramm und die transiente Antwort auf verschiedene Stimuli an.

1.1. Frequenzen und Amplituden

Frequenzen können in unterschiedlicher Form eingeben und angezeigt werden:

- Normalisiert auf die Samplingfrequenz f_S: $F = f/f_S$
- Normalisiert auf die Nyquistfrequenz $f_{Ny} = f_S/2$: $F = 2f/f_S$
- In absoluten Frequenzen, hier können Sie die Abtastfrequenz f_S eingeben. In der Einstellung "Lock" (verriegeltes Schloss) bleiben die absoluten Frequenzen gleich wenn die Abtastrate geändert wird. Bei "Unlocked" werden die absoluten Frequenzen so angepasst, dass die normalisierten Frequenzen gleich bleiben und damit auch Filterentwürfe, Signalverläufe etc.

Amplituden werden in V, W oder dB eingegeben und angezeigt.

1.2 Filterspezifikationen

Filterentwurfsalgorithmen benötigen unterschiedliche Eingabeparameter. Wenn "Order Minimum" angeklickt ist, versucht ein Algorithmus die minimale Ordnung zu den gegebenen Filterspezifikationen zu berechnen, halbwegs genau funktioniert das aber nur bei Equiripple (FIR) und den meisten IIR-Filtern. Bei pyfda können Sie nach dem Entwurf mit minimaler Ordung das Häkchen entfernen und dann selber Entwurfsparameter bearbeiten.

Nicht benötigte Parameter sind entweder deaktiviert oder ausgeblendet. Manche Parameter sind ausgegraut, aber bearbeitbar: Diese Parameter beeinflussen nicht den Filterentwurf, sie dienen aber z.B. dazu um das Toleranzschema anzuzeigen.

2. Filterentwurf

Ein Audiotrack von einer CD ($f_S = 44,1\,\mathrm{kHz}$) soll tiefpassgefiltert werden. Die Filterspezifikationen sind:

- Eckfrequenz des Passbands: $f_{PB} = 3,4\,\mathrm{kHz}$
- Eckfrequenz des Sperrbands: $f_{SB} = 4\,\mathrm{kHz}$
- Max. Ripple im Passband: $A_{PB} = 1\,\mathrm{dB}$
- Min. Dämpfung im Sperrband: $A_{SB} = 60\,\mathrm{dB}$

Es sollen die folgenden Filtertypen getestet werden:

- Fourier-Approximation / Least-Square (Windowed FIR mit Boxcar-Fenster)
- Windowed FIR mit Kaiser-Fenster
- Equiripple
- Butterworth
- Chebychev 1
- Chebychev 2
- Elliptisch (Cauer)

2.1 Filtervergleich

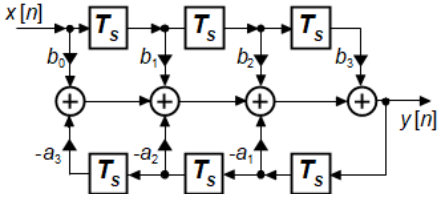
SIMULATION:

Ermitteln Sie die folgenden Filtereigenschaften und fassen Sie sie in einer Tabelle zusammen mit den Spalten:

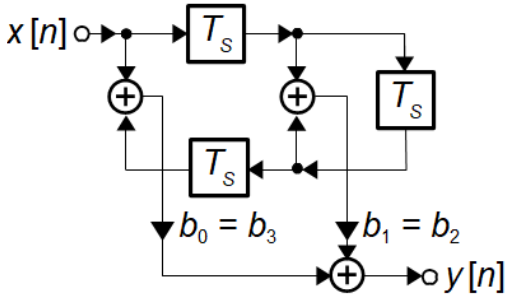
Filtertyp	IIR/FIR?	Ordnung *N*	Gruppenlaufzeit τ_g	Mult. *L* pro Sample	Max. Samplerate [GHz]	Vorteil	Nachteile
Fourier-Approximation / Least-Square(Windowed FIR mit Boxcar-Fenster)	FIR	164	1.84 ps	164	1.83	geringe Nichtlinearität	hohe Rippel / zu geringe Steilheit
Windowed FIR mit Kaiser-Fenster	FIR	268	30.2 ns	268	1.12	keine Rippel im Sperrbereich / hohe Däpfung im Sperrbereich	mittlerer Rippel im Sperrbereich
Equiripple	FIR	163	18.4 ps	163	1.8	geringe Nichtlinearität	hoher Rippel
Butterworth	IIR	PyFDA -> max. 20	1.46 ms	23	13.0	kein Rippel / geringe Nichttlinearität	geringe Steilheit
Chebychev 1	IIR	14	4.75 ms	17	17.6	mittlere Steilheit	hoher Rippel im Durchlassband / hohe Nichttlinearität
Chebychev 2	IIR	14	1.48 ms	17	17.6	mittlere Steilheit	hoher Rippel im Sperrbereich / hohe Nichttlinearität
Elliptisch (Cauer)	IIR	7	2.31 ms	10	30	geringe Ordung	hoher Rippel

- Welche der Filter sind **FIR**, welche **IIR**?
- Vergleichen Sie die benötigte **Ordnung *N***. Für nicht alle der Filter gibt es gut funktionierende Designalgorithmen, Sie müssen dann von Hand die Filterordnung und / oder Eckfrequenzen so lange anpassen, bis alle Spezifikationen erfüllt sind. Im Tab "Info" sehen Sie, ob Spezifikationen verletzt werden.
- Welche Ordnung benötigen Sie, wenn Sie die Abtastrate vor der Filterung auf die Hälfte reduzieren? (Tipp: Aktivieren Sie das "Schloss" neben der Samplingfrequenz, bevor Sie f_S ändern.) Es genügt, wenn Sie das für ein Filter ausprobieren. Was müssen Sie beachten wenn Sie die Samplerate reduzieren wollen?

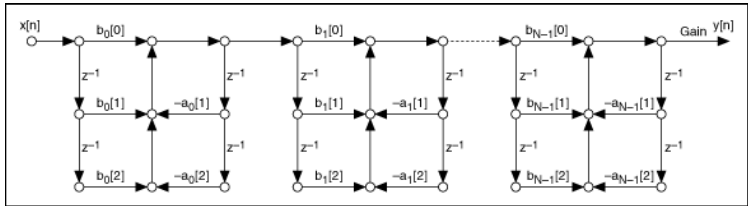
- Welche **Gruppenlaufzeit τ_g** haben die Filter bei $f = 2\,\mathrm{kHz}$? Schauen Sie sich im `y[n]` an, wie die Signalform eines 500 Hz Sägezahnsignals durch das Filter verformt wird.
- Wieviele **Multiplikationen *L* pro Sample** werden benötigt? Berücksichtigen Sie dabei die Filterstruktur (Optimierung durch Linearphasigkeit möglich?), IIR Filter sollen in einer kaskadierten SOS-Struktur realisiert werden - siehe folgende Abbildungen.
- Welche **maximale Samplerate** kann man auf einem DSP mit max. $300 \cdot 10^6$ Multiplikationen / s verarbeiten? Zusätzliche Informationen: How fast are DSPs?
- Fügen Sie den **Plot des Betragsgangs** mit Phasengang (Checkbox "Phase") ein.



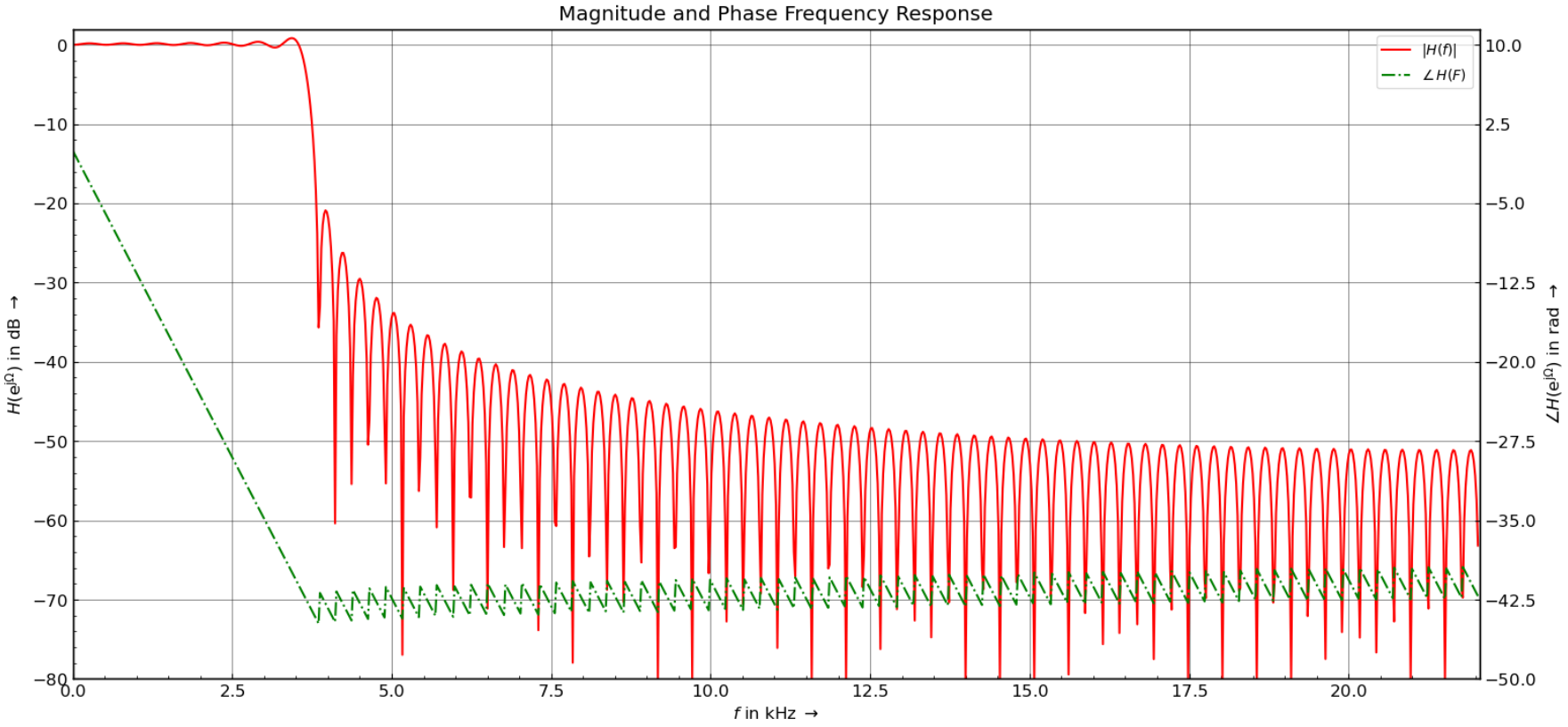
Direktform



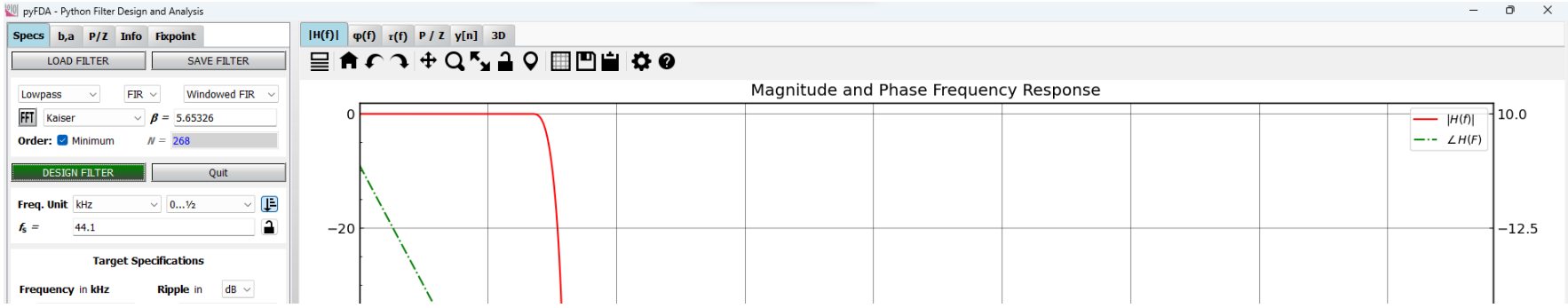
Lin. phasige Form

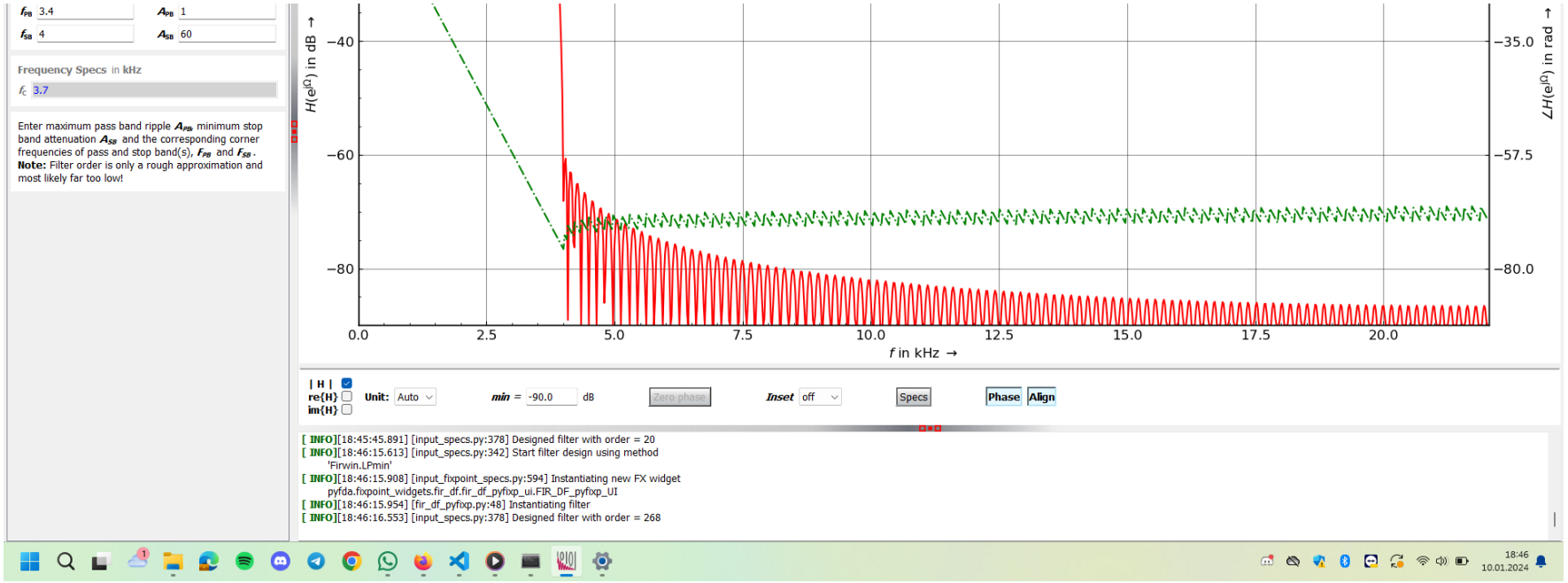


Kaskadierte Second Order Sections

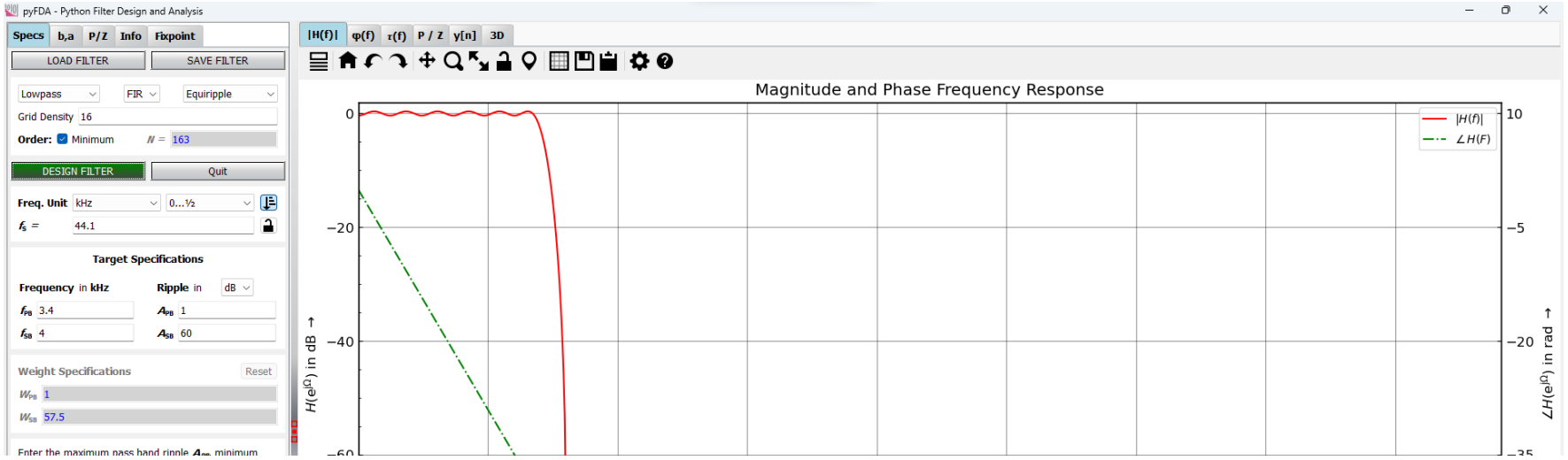


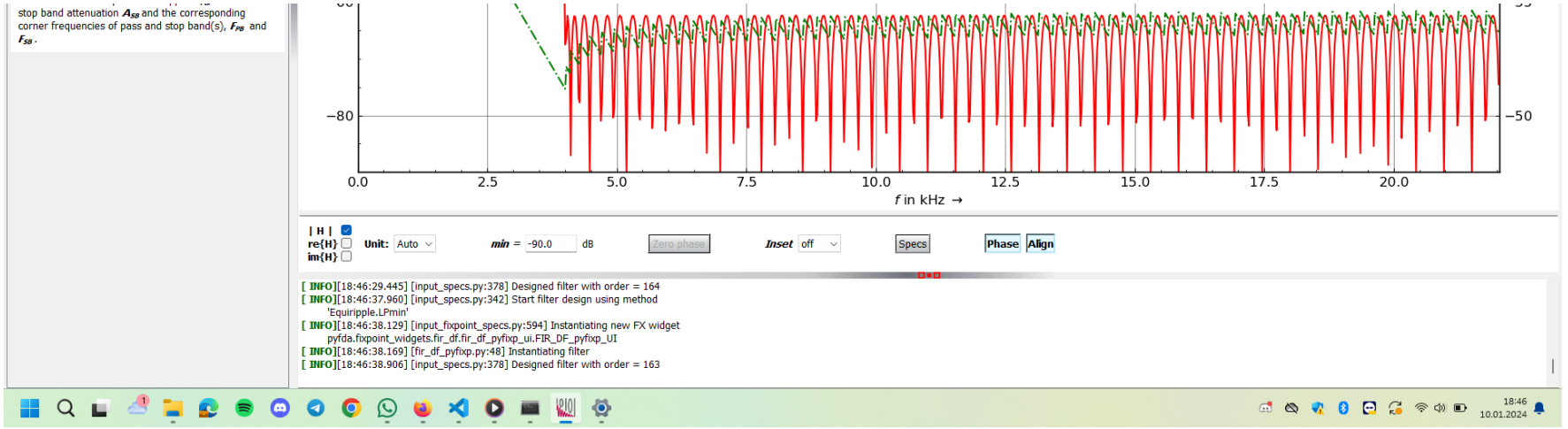
Windowed FIR mit Kaiser-Fenster:



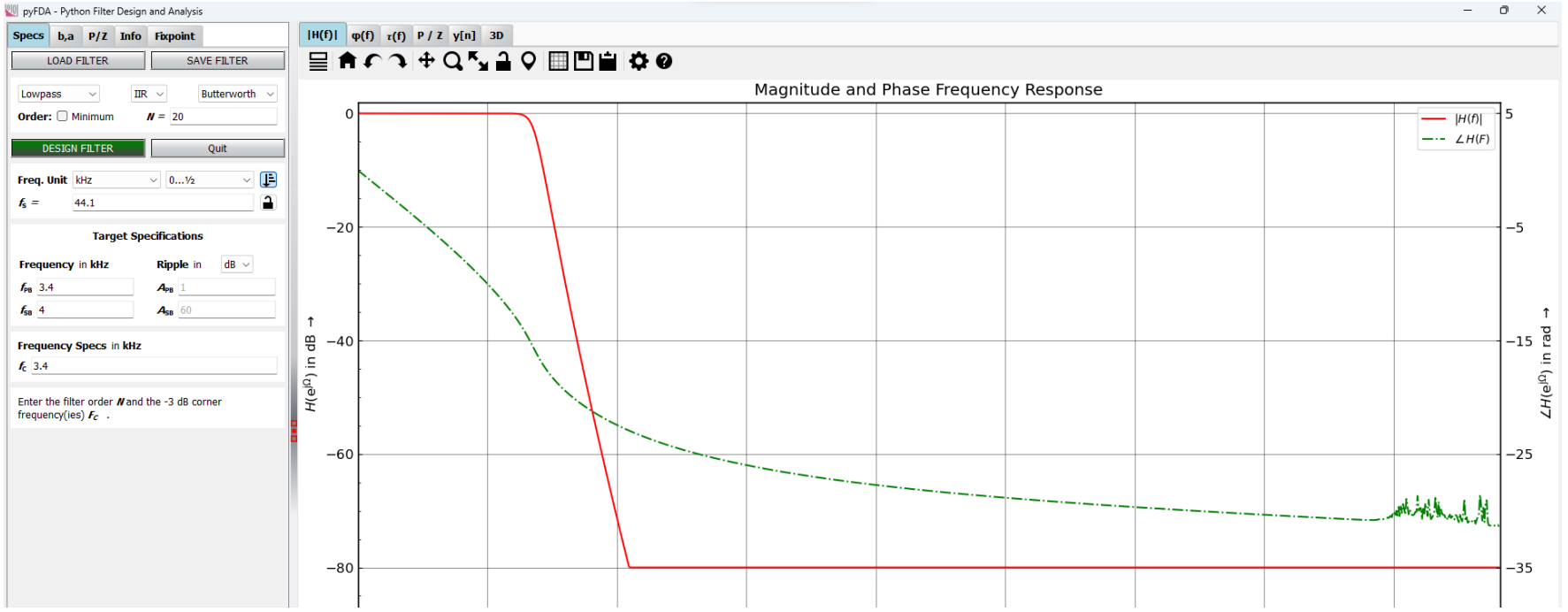

Frequency Specs in kHz
Enter maximum pass band ripple minimum stop band attenuation and the corresponding corner frequencies of pass and stop band(s), and .
Note: Filter order is only a rough approximation and most likely far too low!
f in kHz →
Unit: Auto
min = -90.0 dB
Zero phase
Inset off
Specs
Phase
Align
[INFO][18:45:45.891] [input_specs.py:378] Designed filter with order = 20
[INFO][18:46:15.613] [input_specs.py:342] Start filter design using method
'Firwin.LPmin'
[INFO][18:46:15.908] [input_fixpoint_specs.py:594] Instantiating new FX widget
pyfda.fixpoint_widgets.fir_df.fir_df_pyfixp_ui.FIR_DF_pyfixp_UI
[INFO][18:46:15.954] [fir_df_pyfixp.py:48] Instantiating filter
[INFO][18:46:16.553] [input_specs.py:378] Designed filter with order = 268
18:46
10.01.2024

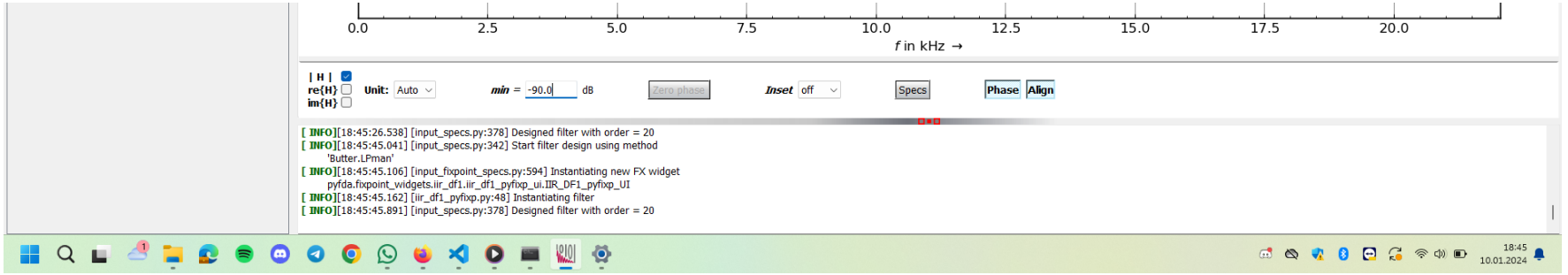
Equiripple:


pyFDA - Python Filter Design and Analysis
Specs
b,a
P/Z
Info
Fixpoint
LOAD FILTER
SAVE FILTER
Lowpass
FIR
Equiripple
Grid Density 16
Order: Minimum N = 163
DESIGN FILTER
Quit
Freq. Unit kHz
fs = 44.1
Target Specifications
Frequency in kHz
Ripple in dB
Weight Specifications
Reset
Enter the maximum pass band ripple minimum
|H(f)|
φ(f)
τ(f)
P / Z
y[n]
3D
Magnitude and Phase Frequency Response
|H(f)|
∠H(F)

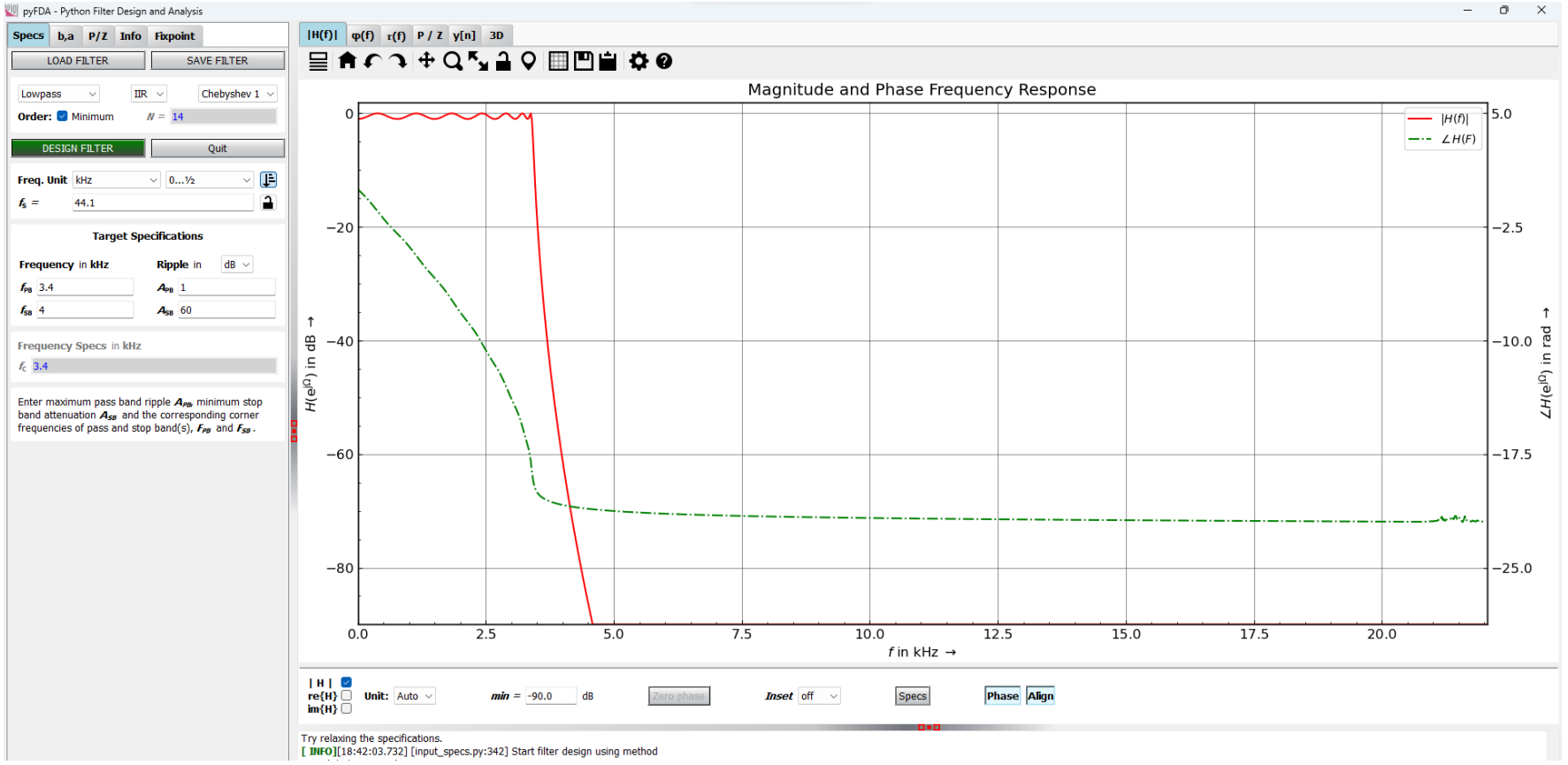


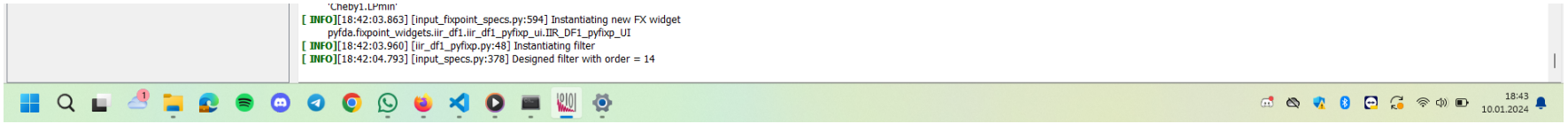
Butterworth:



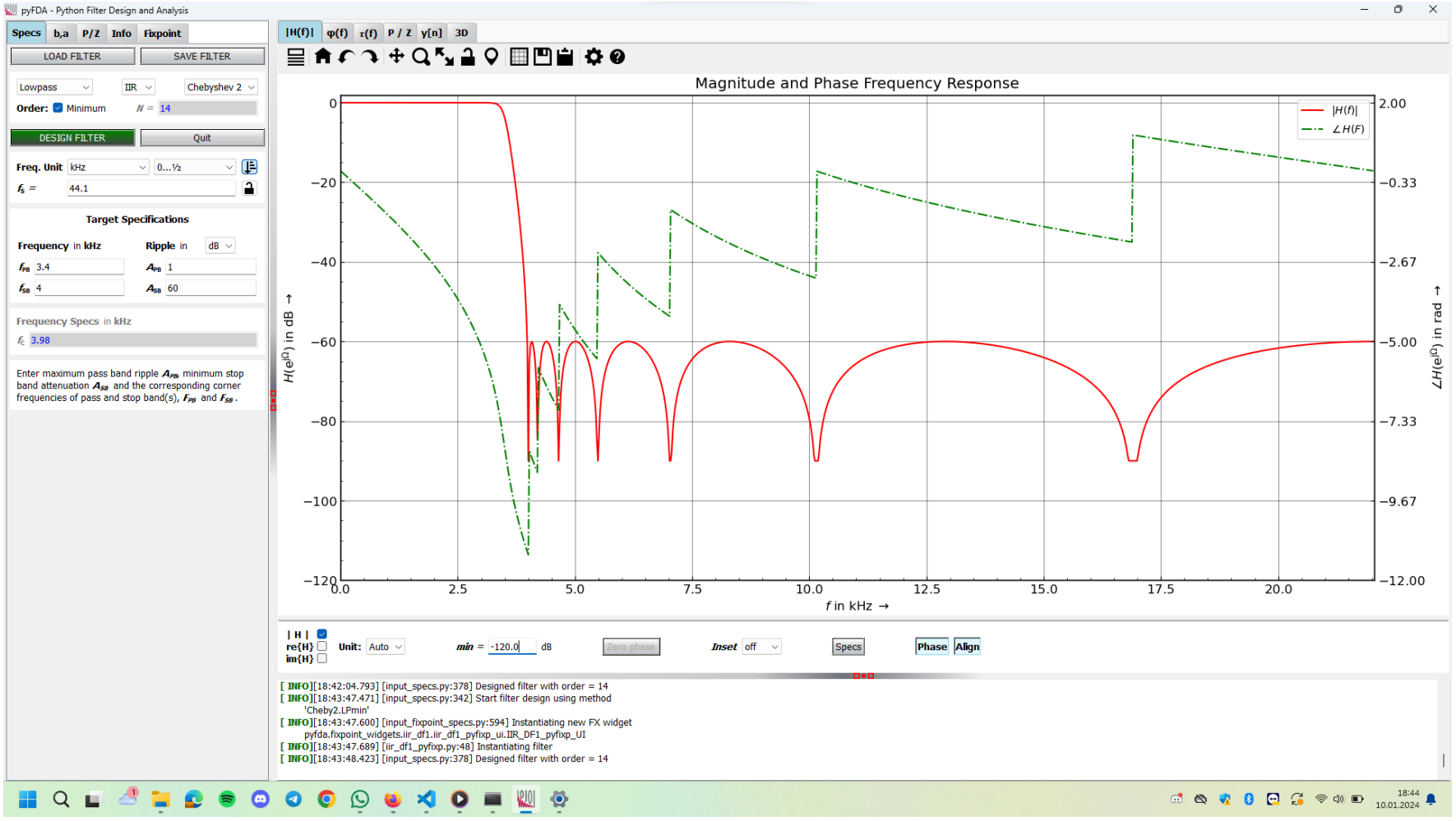

0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
f in kHz →
| H |
re{H}
im{H}
Unit: Auto
min = -90.0 dB
Zero phase
Inset off
Specs
Phase
Align
[INFO][18:45:26.538] [input_specs.py:378] Designed filter with order = 20
[INFO][18:45:45.041] [input_specs.py:342] Start filter design using method
'Butter.LPman'
[INFO][18:45:45.106] [input_fixpoint_specs.py:594] Instantiating new FX widget
pyfda.fixpoint_widgets.iir_df1.iir_df1_pyfixp_ui.IIR_DF1_pyfixp_UI
[INFO][18:45:45.162] [iir_df1_pyfixp.py:48] Instantiating filter
[INFO][18:45:45.891] [input_specs.py:378] Designed filter with order = 20
18:45
10.01.2024

Chebychev 1:

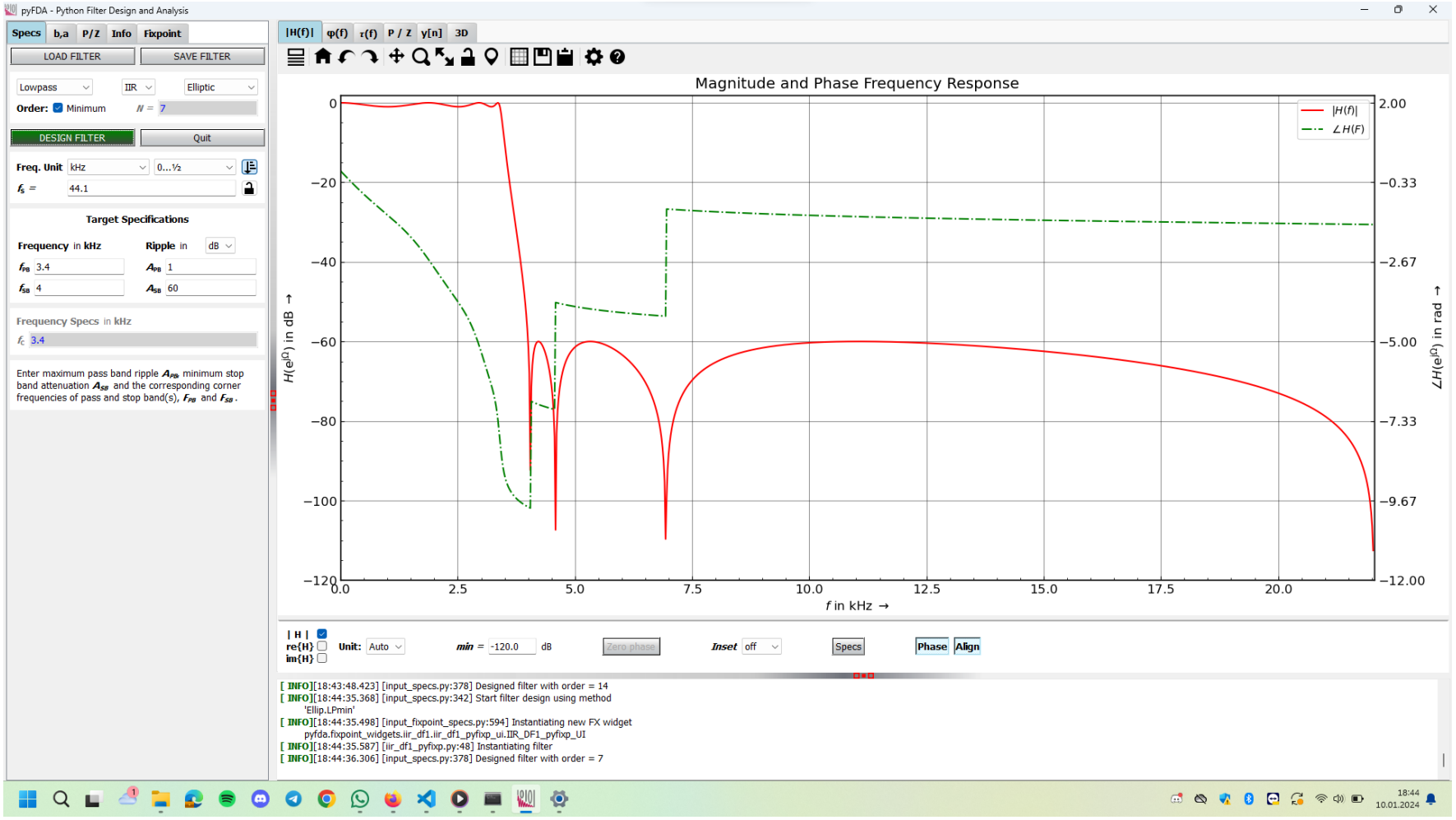

pyFDA - Python Filter Design and Analysis
Specs
b,a
P/Z
Info
Fixpoint
LOAD FILTER
SAVE FILTER
Lowpass
IIR
Chebyshev 1
Order: Minimum
N = 14
DESIGN FILTER
Quit
Freq. Unit kHz
0...½
f_S = 44.1
Target Specifications
Frequency in kHz
Ripple in dB
f_PB 3.4
A_PB 1
f_SB 4
A_SB 60
Frequency Specs in kHz
f_C 3.4
Enter maximum pass band ripple A_PB, minimum stop band attenuation A_SB and the corresponding corner frequencies of pass and stop band(s), F_PB and F_SB.
|H(f)|
φ(f)
τ(f)
P / Z
y[n]
3D
Magnitude and Phase Frequency Response
|H(f)|
∠H(F)
0
−20
−40
−60
−80
5.0
−2.5
−10.0
−17.5
−25.0
H(e^jΩ) in dB →
∠H(e^jΩ) in rad →
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
f in kHz →
| H |
re{H}
im{H}
Unit: Auto
min = -90.0 dB
Zero phase
Inset off
Specs
Phase
Align
Try relaxing the specifications.
[INFO][18:42:03.732] [input_specs.py:342] Start filter design using method


'Cheby1.LPmin'
[INFO][18:42:03.863] [input_fixpoint_specs.py:594] Instantiating new FX widget
pyfda.fixpoint_widgets.iir_df1.iir_df1_pyfixp_ui.IIR_DF1_pyfixp_UI
[INFO][18:42:03.960] [iir_df1_pyfixp.py:48] Instantiating filter
[INFO][18:42:04.793] [input_specs.py:378] Designed filter with order = 14
18:43
10.01.2024

Chebychev 2:


pyFDA - Python Filter Design and Analysis
Specs
b,a
P/Z
Info
Fixpoint
LOAD FILTER
SAVE FILTER
Lowpass
IIR
Chebyshev 2
Order: Minimum
N = 14
DESIGN FILTER
Quit
Freq. Unit kHz 0...½
f_s = 44.1
Target Specifications
Frequency in kHz
Ripple in dB
f_{PB} 3.4
A_{PB} 1
f_{SB} 4
A_{SB} 60
Frequency Specs in kHz
f_C 3.98
Enter maximum pass band ripple A_{PB}, minimum stop band attenuation A_{SB} and the corresponding corner frequencies of pass and stop band(s), F_{PB} and F_{SB}.
|H(f)|
φ(f)
τ(f)
P / Z
y[n]
3D
Magnitude and Phase Frequency Response
|H(f)|
∠H(F)
$H(e^{j\Omega})$ in dB →
$\angle H(e^{j\Omega})$ in rad →
f in kHz →
0
−20
−40
−60
−80
−100
−120
2.00
−0.33
−2.67
−5.00
−7.33
−9.67
−12.00
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
| H |
re{H}
im{H}
Unit: Auto
min = -120.0 dB
Zero phase
Inset off
Specs
Phase
Align
[INFO][18:42:04.793] [input_specs.py:378] Designed filter with order = 14
[INFO][18:43:47.471] [input_specs.py:342] Start filter design using method
'Cheby2.LPmin'
[INFO][18:43:47.600] [input_fixpoint_specs.py:594] Instantiating new FX widget
pyfda.fixpoint_widgets.iir_df1.iir_df1_pyfixp_ui.IIR_DF1_pyfixp_UI
[INFO][18:43:47.689] [iir_df1_pyfixp.py:48] Instantiating filter
[INFO][18:43:48.423] [input_specs.py:378] Designed filter with order = 14
18:44
10.01.2024

Elliptisch:



3. Filterexport und -simulation

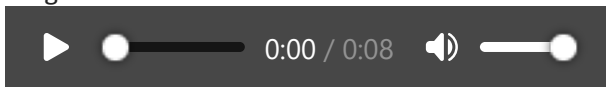
Im folgenden sollen Sie Audiofiles mit dem Equiripple und dem elliptischen Filter filtern. Dazu können Sie aus pyfda Filter in verschiedenen Formaten exportieren:

formaten exportieren.

In [ ]:
```
x_w = wav2np("../medien/87778__marcgascon7__vocals.wav")

#Equiripple
b_FIR = [0.0006123033469102477,0.0005231312265310316,0.000561083771645731,0.0004151809876883935 5,2.827711686114232e-05,-0.0006
a_FIR = [1.]


#Elliptisch
b_IIR = [0.001710952004172366,-0.005755838859144107,0.008363025285148405,-0.004112746893907044,-0.004112746893907043,0.0083630
a_IIR = [1.0,-6.10402966946488,16.40268768530511,-25.100822235115835,23.595177173520497,-13.615049072423911,4.4641836431418325


#Signal Filtern
y_FIR = sig.lfilter(b_FIR, a_FIR, x_w)
y_IIR = sig.lfilter(b_IIR, a_IIR, x_w)

print("Orginal")
display(Audio(data=x_w, rate=wav2np.f_S))


print("------------------------------------------------------------")
print("FIR: Equiripple")
print(y_FIR.dtype)
display(Audio(data=y_FIR, rate=wav2np.f_S))


print("------------------------------------------------------------")
print("IIR: Elliptisch")
print(y_IIR.dtype)
display(Audio(data=y_IIR, rate=wav2np.f_S))
```

Hören Sie einen Unterschied zwischen dem Orginal und den beiden unterschiedlichen Filtern? Woran liegt das?

```
2 channels with 349952 frames of 2 bytes and f_S = 44100 Hz.
Orginal
```

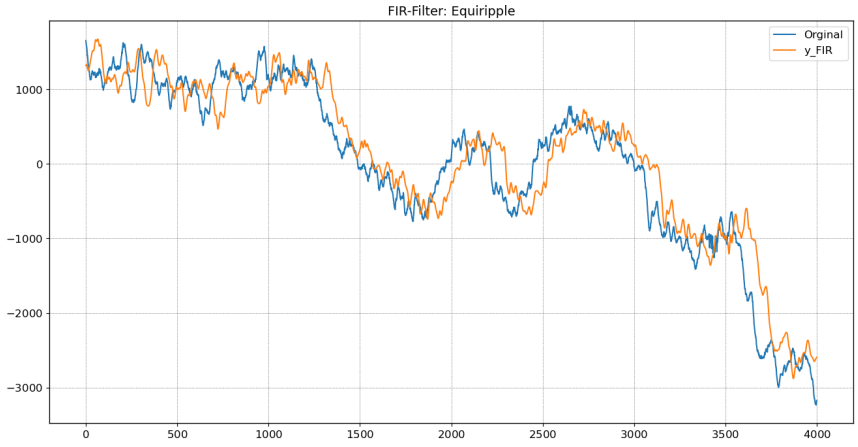


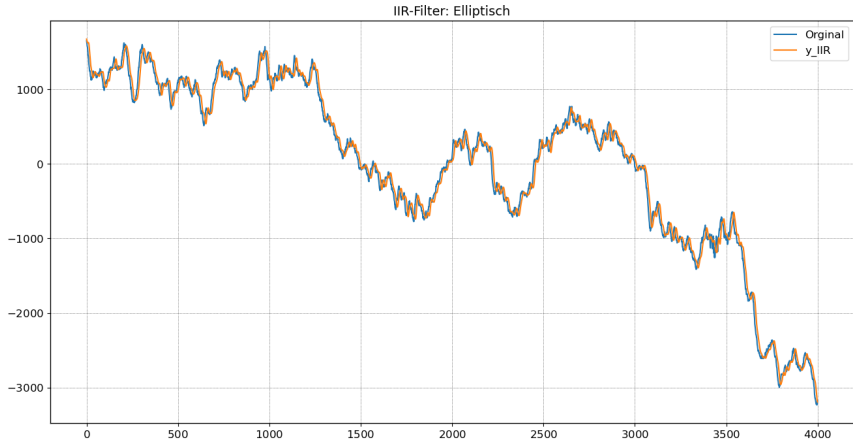

```
------------------------------------------------------------
FIR: Equiripple
float64
```

```
------------------------------------------------------------
IIR: Elliptisch
float64
```

In [ ]:
```
#fig, ax = plt.subplots(**size)
fig, ax = plt.subplots(nrows=1, ncols=2, figsize=(35,8))
l = 4000
start = 20000


ax[0].set_title("FIR-Filter: Equiripple")
ax[0].plot(np.arange(l), x_w[0][start:start+l], label="Orginal")
ax[0].plot(np.arange(l), y_FIR[0][start:start+l], label="y_FIR")
ax[0].legend()

ax[1].set_title("IIR-Filter: Elliptisch")
ax[1].plot(np.arange(l), x_w[0][start:start+l], label="Orginal")
ax[1].plot(np.arange(l), y_IIR[0][start:start+l], label="y_IIR")
ax[1].legend()
```

Out[ ]:
```
<matplotlib.legend.Legend at 0x1b3ef61a380>
```





Copyright