Inverse Fourier Transformation

04.06.2018, Sebastian Schmidt (sebastian.seb.schmidt@fau.de)

github.com/schmidtseb/RaspberryPiUebung

Modifikation am Ausleseprogramm

Im Audio-Ausleseprogramm wurde der Betrag der FFT zurückgegeben, der für das Plotten des Spektrums wichtig ist. Um eine Rücktransformation vorzunehmen, muss jedoch der tatsächliche Wert zurückgegeben werden. Dies ist in der folgenden Funktion korrigiert:

```
def calculateFFT(time, amplitude):
    # Mittlere Zeitdifferenz zwischen zwei Datenpunkten
    tMean = np.mean(np.diff(time))

    # Erzeuge neue Zeiten
    timeNew = np.arange(min(time), max(time), tMean)

    # Interpolation
    amplitudeNew = np.interp(timeNew, time, amplitude)

    # Calculate FFT
    freq = np.fft.rfftfreq(timeNew.size, tMean)
    freqAmplitude = np.fft.rfft(amplitudeNew)

    return freq, freqAmplitude
```

Hierbei wird die Information der Frequenz als komplexe Zahl ausgegeben, wobei der Realteil die Amplitude und der Imaginärteil die Phase repräsentiert. Beim Abspeichern muss dies wie folgt berücksichtigt werden:

```
    # Erzeugung und Speicherung der Freuqenzdaten
    freq, freqAmplitude = calculateFFT(timeList, amplitudeList)
    frequencyData = np.asarray( [freq, abs(freqAmplitude),
np.real(freqAmplitude), np.imag(freqAmplitude)] ).T
    np.savetxt('frequencyData.dat', frequencyData, header='Frequency
(Hz)\tAmplitude (a.u.)\tRealteil\tImaginaerteil')
```

Inverse FFT

Die gespeicherten Daten werden über folgende Zeilen wieder geladen

```
f, A, Ar, Ai = np.genfromtxt('frequencyData.dat').T
t = 1. / (f[1] - f[0])
amplitude = np.fft.irfft( Ar + 1j*Ai )
time = np.linspace(0, t, len(amplitude))
```

Über `Ar + 1j*Ai` wird aus Real- und Imaginärteil wieder eine komplexe Zahl gemacht. Die Funktion `irfft()` führt letztendlich die inverse Fourier Transformation aus, die am Ende wieder die Amplitude des Signals zurückliefert.

Frequenzfilter

Frequenzen können ganz einfach nach Ausführen der FFT herausgefiltert werden. Wählt man einen zu filternden Bereich `fmin <; f <; fmax`, so kann dieser über

```
Ar[np.logical_and(f > fmin, f < fmax)] = 0
Ai[np.logical_and(f > fmin, f < fmax)] = 0
```

im Frequenzspektrum auf null gesetzt werden. Danach wird, wie vorher beschrieben, eine Rücktransformation durchgeführt.