Anleitung und Hinweise zum Python Skript zur Messung und Vorverzerrung

Denys Bast

18.04.2018

**Inhaltsverzeichnis**

[1 Grundlegendes 1](#_Toc511808785)

[2 Lineare Vorverzerrung 2](#_Toc511808786)

[2.1 runme\_compute 2](#_Toc511808787)

[2.2 computeUin 2](#_Toc511808788)

[2.3 getH 2](#_Toc511808789)

[2.4 MLBS 3](#_Toc511808790)

[2.5 FFT 3](#_Toc511808791)

[3 Nichtlineare Vorverzerrung 4](#_Toc511808792)

[3.1 NL\_allesineinem 4](#_Toc511808793)

[3.2 computeUin 4](#_Toc511808794)

[3.3 U\_inp\_allg 4](#_Toc511808795)

[3.4 Param\_Signal 5](#_Toc511808796)

[3.5 K1 5](#_Toc511808797)

[3.6 NL\_vorverzerrung 5](#_Toc511808798)

[3.7 writeAWG 5](#_Toc511808799)

[3.8 writeDSO 5](#_Toc511808800)

[3.9 getH 5](#_Toc511808801)

[3.10 MLBS 5](#_Toc511808802)

[3.11 FFT 6](#_Toc511808803)

[4 Ausblick 7](#_Toc511808804)

# Grundlegendes

Das Tool dient dazu, ein Eingangssignal zu bestimmen, dass zu einer einzelsinusförmigen Spannung am Gap eines Kavitätensystems führt.

Das Tool besteht aus drei Teilen.

* Übertragungsfunktion bestimmen
* Lineare Vorverzerrung
* Nichtlineare Vorverzerrung

Die Funktionalität des vorliegenden Python Skript wurde in verschiedenen Instanzen entwickelt.

Zuerst wurde im Rahmen der Lehrveranstaltung Projektseminar Beschleunigertechnik ein Tool entwickelt, das dazu dienen soll die Übertragungsfunktion eines Kavitätensystems zu messen. Dazu kann für detailliertere Erklärungen die zugehörige Ausarbeitung konsultiert werden.

Auf der Basis von Matlab-Skripten von Jens Harzheim wurde die Funktionalität dann so erweitert, dass eine lineare Vorverzerrung vorgenommen werden kann.

Die Entwicklung der Funktionalität der nichtlinearen Vorverzerrung ist zu diesem Zeitpunkt noch nicht abgeschlossen bzw. noch nicht getestet.

Es ist immer darauf zu achten, dass das Tool nur in einem sehr begrenzten Umfang getestet wurde. Dies beschränkt sich im Wesentlichen auf 80MHz Übertragungsfunktion und 900kHz Kavitätensystem.

Die Implementierung ist in Python 3.6 über die Python-Distribution Anaconda erfolgt. Für die Ausführung wurde die in Anaconda mitgelieferte IDE Spyder verwendet (Tipps beziehen sich auf dessen Oberfläche). Auf dem Gruppenlaptop der HF Group liegt die nötige Installation vor.

# Lineare Vorverzerrung

## runme\_compute

Diese Datei muss ausgeführt werden, um die lineare Vorverzerrung durchzuführen. Dazu werden folgende Schritte von der Methode (automatisch) ausgeführt:

* Übertragungsfunktion mit getH aus Projektseminar berechnen
* Eingangssignal berechnen, dass mit der gemessenen Übertragungsfunktion zu einem Einzelsinus am Gap führt (computeUin)
* Berechnetes Signal auf den AWG geben

## computeUin

Hier wird das notwendige Eingangssignal mit Hilfe der Übertragungsfunktion berechnet. Dazu wird die bekannte Fourier Reihe des Einzelsinus mit der Übertragungsfunktion „zurückgerechnet“.

* Zeile 47-52: Die Phase der gemessenen Übertragungsfunktion durch das Tool weist normalerweise Phasensprünge auf. Diese werden hier korrigiert. Dies nicht zu machen hat in der Vergangenheit zu Fehlern geführt!

for ind in range (0,(len(Phase)-1)):

if PhaseVGL[ind]\*PhaseVGL[ind+1]<0:

if PhaseVGL[ind]>np.pi/2 and PhaseVGL[ind+1]<-np.pi/2:

Phase[ind+1:]=Phase[ind+1:]+2\*np.pi

elif PhaseVGL[ind]<-np.pi/2 and PhaseVGL[ind+1]>np.pi/2:

Phase[ind+1:]=Phase[ind+1:]-2\*np.pi

## getH

Hauptsächlich aus dem Projektseminar übernommen. Änderungen sind:

* Zeile 65-73: die Adresse des AWG die zum Ansteuern benötigt wird, wird mit dem String “USB” gesucht. Andere angeschlossene USB Geräte können hier zu Problemen führen!

*rm = visa.ResourceManager()*

*rs = rm.list\_resources()*

*for i in range (0,len(rs)):*

*pruf=rs[i]*

*test=pruf.find("USB")*

*if test != -1:*

*index=i*

*awg\_id = rs[index]*

*AWG = rm.open\_resource(awg\_id)*

* Zeile 162-168: Das Scaling des Oszis wird noch nicht automatisch ermittelt. Je nach Verstärker oder Übersetzungsverhältnis im Gapspannungsteiler muss hier die Auflösung angepasst werde,

DSO.write("HORIZONTAL:RECOrdlength " + str(recordLength)) #1e5

DSO.write("CH1:SCAle " + str(awg\_volt/6)) #Sets the vertical scale

DSO.write("MATH1:SCAle " + str(awg\_volt/6)) #Sets the vertical scale

DSO.write("CH2:SCAle 20.0E-3") #Sets the vertical scale

DSO.write("CH3:SCAle 50.0E-3") #Sets the vertical scale

DSO.write("CH4:SCAle 50.0E-3") #Sets the vertical scale

DSO.write("MATH3:SCAle 200.0E-3") #Sets the vertical scale

## MLBS

Aus dem Projektseminar übernommen. Dient zur Berechnung des Rauschsignals für die Messung der Übertragungsfunktion.

## FFT

Aus dem Projektseminar übernommen. Dient zur Berechnung der FFT die in getH benötigt wird.

# Nichtlineare Vorverzerrung

Die nichtlineare Vorverzerrung funktioniert noch nicht, bzw. ist noch nicht fertig entwickelt. Hier wird dokumentiert, wie der aktuelle Stand ist.

## NL\_allesineinem

Diese Datei muss ausgeführt werden, um die Nichtlineare Vorverzerrung durchzuführen. Hier ist markiert bis wohin die Methode als funktionstüchtig gilt. Man kann sie teilweise ausführen, indem man Code markiert unf F9 drückt (wie Matlab). Dabei ist darauf zu achten, dass alle import Befehle richtig gemacht wurden. Ggf. muss vorher einmal ein Syntax Fehler eingebaut werden, dann Play drücken (grüner Pfeil in der oberen Leiste bei Spyder) und dann den gewünschten Code mit markieren und F9 ausführen. Dies ist ein workaround, da wenn man in einer importierten Methode etwas ändert und diese ohne Play zu drücken wieder importiert (nur mit F9) es manchmal die alte Methode weiter verwendet.

## computeUin

Diese Methode berechnet die lineare Vorverzerrung. Die Korrektur der Phase wird nicht mehr in dieser Methode vorgenommen, sondern direkt in getH, sodass die übergebene Phase der Übertragungsfunktion direkt für die weitere Berechnung genutzt werden kann. Außerdem wird das berechnete Eingangssignal als csv gespeichert (UinL.csv). Der Ordnername entspricht der Uhrzeit und dem Datum zu dem die Methode ausgeführt wurde (wie im Projektseminar). Achtung! Es wird ein neuer Ordner erstellt! Die Datei liegt nicht im gleichen Ordner wie die Übertragungsfunktion.

## U\_inp\_allg

Diese Methode berechnet ein Eingangssignal zu gegebener Übertragungsfunktion und gegebenem Ausgangssignal. Im Testfall liefert die Matlab Methode und die Python Methode das gleiche Ergebnis. Dennoch ist dieses Skript noch nicht über alle Zweifel erhaben. Das Signal sieht sehr unterschiedlich aus, obwohl nur moderate Nichtlinearität erwartet wird.

## Param\_Signal

In dieser Methode wird aus linear vorverzerrtem Eingangssignal und mit der Übertragungsfunktion zurückgerechnetem Gapsignal eine Kennlinie berechnet. Bisherige Tests beschränken sich auf eine Kennlinie vom Grad 3. Dazu werden die beiden Signale mittels Kreuzkorrelation passend übereinander gelegt, dann eine Matrix erzeugt, die mittels least square Methode (Zeile 56: np.linalg.lstsq) Koeffizienten (a,b,c) für die Kennlinie liefert (ax+bx^2+cx^3).

## K1

Diese Methode wird in „Param\_Signal“ aufgerufen, um die Kennlinie grafisch darzustellen und als eine Art look-up table zu speichern.

## NL\_vorverzerrung

Diese Methode führt die eigentliche Vorverzerrung durch. Das erstellte Signal und die verwendete Kennlinie werden analog zu linearen Vorverzerrung als csv gespeichert (UinNL.csv, Kennlinie.csv).

## writeAWG

Diese Methode schreibt Signale in das AWG. Hier ist wieder darauf zu achten, dass andere usb Geräte neben dem AWG zu Problemen führen können.

## writeDSO

Diese Methode misst Ein- und Ausgangssignal mit Hilfe des Oszilloskops. Hier ist die Skalierung des Oszilloskops ebenfalls nicht automatisiert. Änderungen im Messaufbau müssen ggf. berücksichtigt werden (siehe getH lineare Vorverzerrung).

## getH

Diese Methode entspricht der bei der linearen Vorverzerrung mit der Erweiterung, dass die Phase der gemessenen Übertragungsfunktion automatisch korrigiert wird, sodass sie keine Sprünge mehr aufweist.

## MLBS

Aus dem Projektseminar übernommen. Dient zur Berechnung des Rauschsignals für die Messung der Übertragungsfunktion.

## FFT

Aus dem Projektseminar übernommen. Dient zur Berechnung der FFT die in getH benötigt wird.

# Ausblick

Folgende Aufgaben sind noch offen:

* Nichtlineare Vorverzerrung (Hinweise, wo Fehler vermutet werden, sind im Code kommentiert)
* Automatische Einstellung des Scaling beim Oszi (MEASURE Funktion)
* AWG durch neue Signalgeneratoren ersetzen (Tabor)
* Andere Testcases (Frequenzen des Kavitätensystems, Schieberegisterlängen..)