MASTER DEGREE: Master's degree in Applied Telecommunications and Engineering Management (MASTEAM)

Thesis Title: Digital Signal Processing of Communication Signal With Python: DPD

MD Samsuzzaman

SUPERVISED BY

Dr. Gabriel Montoro López

Universitat Politècnica de Catalunya

Master's degree in Applied Telecommunications and Engineering Management (MASTEAM)

Overview

This paper demonstrates the opportunities that the new AIs provide for DPD (Digital Predistortion) in RF PA. The project is established on the AD9361 Testbench which was programmed and simulated using MATLAB. For execution in the testbed and evaluation of the performance of DPD, it contains 3 main codes. Our work included the development of a python code which was executed simultaneously with this MATLAB and the results are compared within both interfaces.

Digital Predistortion includes a part with memory and one without memory. The project we are working with has three steps, (i) Baseband IQ signal xBB generation (ii) TX: Delivery of xBB to a transmitter TX (true or simulated) (iii) RX: Base band signal yBB collection. After the delivery to the transmitter (one similar to the AD9361 one), the hardware processes the signal by, a)Upconversion to Radiofrequency, b) Amplification using Power Amplifier and c) Dispatching to an antenna. In the third step, the band signal yBB coincides with one that by the downconversion of the radiofrequency signal was received in AD9361.

The eminent part of this project was the python script that we designed to observe how the DPD processes within the changed environment. Rudimentarily, some complications arose but we could overcome them and the final result resembled the outcomes from MATLAB.

This project proposes the following aspects:

1. Sophisticated AI development for DPD.
2. Simpler algorithm design to test DPD in PA.
3. A full proof design of DPD to reduce PA energy consumption.
4. Participate in universal global warming reduction by increasing overall energy efficiency as stated in the preceding proposal.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1. INTRODUCTION

Radio frequency power amplifiers (RF – PAs) are leading to be a very crucial part of the newly emerging 5G technology. Inclination towards wireless applications is rising because of its mobility and data rates and the 5G technology has shown its efficacy in this sector. Therefore, RF – PAs are going to be a component of copious possibilities.

In this aspect, DPD is the most exemplary algorithm in taking care of PAPR problems in PA and there is no dearth of research works in this area of telecommunication systems. Ideally, this should be an inverse function of the one used in PA. Theoretically, DPD is a mathematical function which is implemented in the signal to reverse the deteriorating effect of losing information during nonlinear amplification through PA. This process is robust and iterative as it's coefficients can adapt according to input and time of the signal.

Variant processing steps lead to the formulation of DPD in PA thus taking us to AD9361 testbench with SDR. MATLAB is commonly used with SDR but we approached Python juxtaposing with the MATLAB processing to get a comparison. Basic concepts of both processes will be discussed below.

1.1. DPD Overview

Higher efficiency can be achieved for higher drive levels when DPD is implemented as it uses digital signal processing techniques to overcome nonlinear distortion in the Radio frequency power amplifiers [1]. It has become the most popular technique as it linearizes the nonlinear PA by standalone add-on digital block which eventually provides freedom to vendors from the complex and burdensome manufacturing process for complex analog/RF circuits.

Typically used baseband domain DPD systems are limited to a smaller range of signal bandwidth which can be solved by using an RF domain DPD system in the power amplifier of a transmitter (in which DPD system is most commonly used) to correct the nonlinear distortions. In the Rf domain, DPD performs after RF domain up-samples the signal and they can handle wider, variant bandwidths with more efficacy in the spectrum.

1.2. SDR Overview

Software defined Radio, a wireless communication system that is being used in a variety of existing and next generation technology operates at radio frequency. It is configurable using software and components like Amplifiers, [2] Modulator or demodulator modules, Filters (FIR filter concept etc. can be configured, whereas hardware components can use FPGA or DSP.

Transmitter:

In this part of the architecture, the digital baseband part coded in DSP renders I/Q data according to the needs per transmitter. DUC, digital LO and digital mixer are used to digitally convert them.

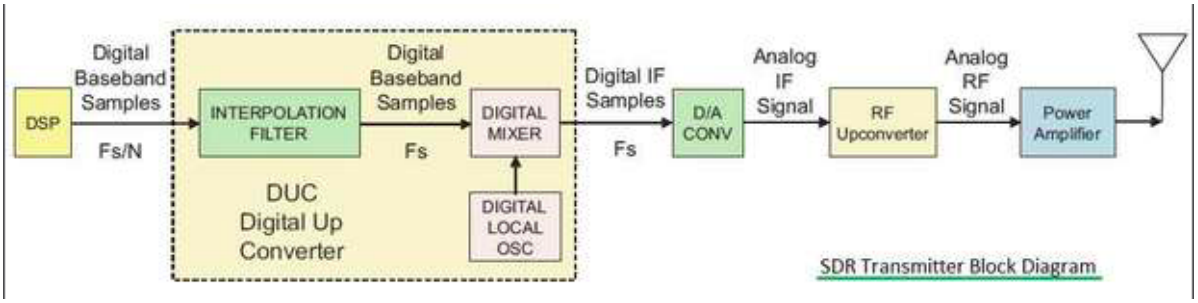

Figure- 1.1. Depicts transmitter parts of SDR architecture.

The digital IF is converted to analog IF which is again converted to analog RF using the RF converter and finally it is amplified before transmission over the air using the proper antenna.

Receiver:

In the analog receiver from the figure, RF signal is amplified and fed to the RF mixer to perform RF down conversion. By beating the input amplified RA signal with the local LO signal, this down conversion is carried out. The extracted and amplified IF signal is demodulated and as we can see in the figure is passed to the audio amplifier. In amplitude and frequency demodulators, envelope detectors and frequency discriminators are used respectively. The conversion of RF to IF which is a major function in this case is done in the RF mixer.

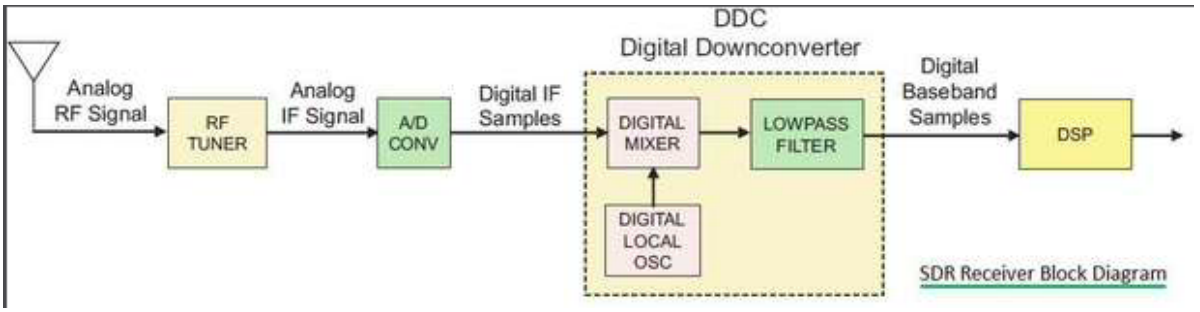

Figure- 1.2. Depicts receiver parts of SDR architecture.

For Am and PM broadcasting receivers, 455 KHz and 10.7MHz IF centre frequencies are used.

Typical algorithms like I/Q gain, phase imbalance, DC offset, frequency, time and channel impairment correction can be used for SDR. In order to correct the realtime baseband and RF related impairments found in the I/Q data, DSP or FPGA processing chain can be helpful with advanced algorithms.

Benefits:

- Cost efficient due to a digitalized system that enables less time consumption and easy testing during changes.
- Easy to work on and perform new experiments even when the system is in its operational mode.
- Upgrading to different versions is easier with the software version. Thus coping up with the new technologies and market requirements is easier.

CHAPTER 2. STATE OF THE ART

2.1. Evolution of DPD

For the past several years, designing wireless communication systems having capability to handle multi-standard or multi-band signals simultaneously have been researched on. One single PA processing multi-bands simultaneously significantly reduces component necessity and cost of RF subsystems. Multi-band signals can be simplified considering the nonlinear distortion requiring attention are the ones arising close to the band of interest and filtering others. Multidimensional DPD, that is, the multiple input single output model, is useful in many realistic cases to recover unwanted distortion where wideband DPD is not feasible. For instance, in PAs, while combining concurrent multi-band transmissions with dynamic load or dynamic supply modulation. In the same way, in multi-antenna systems also, bulky components are removed so that system costs and complexity does not occur as each transmit path contains its own PA and antenna element. These removal leads to nonlinear distortion due to the mix of antenna crosswalk and mismatch with the PA output, adding to the distortion already formed by the PAs. [7]

2.2. Advantages of DPD

DPD is complementary in reducing the cost per operation which comes from the shrinking of the transistor size, improving the efficiency of PAs in BS situations for about three decades. The required spectral mask cannot be compromised for Base Stations. In Spite of being high power consuming, DPDs performance is eminent [43] for example, with feedforward, 3%-5% power efficacy of a normal PA for WCDMA system can have 6%-8% whereas with DPD it is 8%-10%. Cost of DPD will go down significantly with advanced scaling and smarter application of memory polynomial based DPD through look-up-table which we can see in the DPD showed in [46] which takes in a meager of 40mW for predistortion of a 20 MHz LTE signal.

2.3. Disadvantages of DPD

While implementing DPD, other elements present in the part from the start should have the capacity to handle wide signal bandwidth. Along that, it is a generic rule that the bandwidth should be five times the signal bandwidth in the loop. Smaller dimension surely reduces the cost but continuous increase in the communicational signal bandwidth brings about other challenges to actuate the design and the amount of power consumed. The change in power overhead is tantamount to the change in clock frequency, thus the power overhead increases with the increasing bandwidths also giving rise to timing closure challenges while digital implementation. If we consider a regular zero-IF transmitter, it contains components in single chain, DACs in I and Q paths, IQ modulator, filters for reconstruction or anti

imaging or RF band pass. These parts must be able to sustain the wide bandwidth to reduce power overhead otherwise it can be ineffective for small cell BS PA. Therefore, power overhead reduction during all levels of design should be taken into careful consideration.

2.4. Challenges and Issues

In wideband systems, DPD needs to be run under band-limit situations. In this situation, the act of linearization will lose its quality, which might not occur if major nonlinear distortion could be captured correctly. So, a major challenge occurs to design the DPD module in such a way that it works efficiently under band-limit. Alongside wireless service providing continuous frequency, a wider frequency band can be aggregated from various smaller bands which might or might not have gaps between sub-bands.

On the other hand, the continuous rise of digital system bandwidth and sampling rate, more complex models in application with more coefficients, the whole system will be getting more and more complex. This will cause the necessity of more clock cycles to extract the coefficients and demand for ADC and DAC systems. Thus, power consumption and costly components will be on the rise.

Finally, in small sized devices consuming low power, implementing a DPD system is a challenge. It's typically designed for high or medium power amplifiers. DPD’s consumption of energy will reduce a whole transmitter's efficiency which is why it is not suited to low power amplifiers.

2.5. Energy efficiency, power consumption, and cost-saving

Recently increasing energy efficiency combining with a decrease in cost has been an important issue of research. It is observed recently that prior to amplification, using digital pre-processing for input signal, analog RF power amplifiers can provide optimum efficiency without causing any significant distortion thus leading to improved linearity and energy efficiency in the system.

CHAPTER 3. POWER AMPLIFIER

3.1 Power Amplifier Overview

Radio frequency power amplifiers are power devices used to convert DC supply into radio frequency power, amplifying the signal before transmitting to antenna. In the 1960s, these solid-state RF power devices were starting to be used. Before that power was generated using spark, arc, vacuum tube transmitters etc. These RF power devices have been used in a variety of types including MOSFET, HFET, HEMT, HBT in industrial cases and have been tempered to cause improvements leading to extended bandwidth and high output power levels [41].

During the designing of power amplifiers, as they work to enhance the power to a higher level, output power level represents the performability of the PAs as well as the power gain. Also, efficiency is important which is directly related to costing. High linearity is expected from a PA as it lessens the distortion providing higher integrity.

But linearity and efficiency are inversely related. For example, Doherty, envelope tracking etc efficient PA designs work very poorly in consideration of linearity. Thus, high efficiency PAs need to be checked for exemplary linearity too.

3.2 Power Gain and Efficiency

Power efficiency is the percentage of power converted from the amount supplied. An amplifier is used to increase the intensity of the input signal but not all of it is transferred to the output signal [3]. The part which is converted can be indicated using the efficiency. Drain efficiency and power added efficiency: the two ways efficiency can be described [4]. Normally, in both cases the value reaches the culminating point at saturated power and is proportional to the power when decreasing. Linear problems can occur at high power regions.

Drain efficiency is the generalized one having the following equation -

$$\eta_d = \frac{P_{out}}{P_{DC}} \tag{3.1}$$

Here, P_{out} = power delivered to the load; P_{DC} = power taken from supply

Input power is not required for calculating drain efficiency.

On the other hand, power added efficiency can be described with the following

equation -

$$PAE = \frac{P_{out} - P_{in}}{P_{DC}} \tag{3.2}$$

Here, P_{in} = input power

It is evident that power added efficiency includes input power in calculation and subtracts the value as a power loss thus providing more accurate results. Again, the input power can be set as a function of power gain which gives out a reasoning of power amplifiers performance with respect to power gain but it can be negative if power gain is too low. The following equation describes this assertion where G is the power gain, which is basically the ratio of output to input power in a two-port device -

$$PAE = \frac{P_{out} - P_{out}/G}{P_{DC}} \tag{3.3}$$

Here G = P_{out} =P_{in}

Which is the power gain of the power amplifier. The power gain of a two-port device, for instance a power amplifier, is the ratio of the output power to the input power. It is usually represented in decibel as,

$$G_{dB} = 10\ log_{10}(\frac{P_{out}}{P_{in}}) \tag{3.4}$$

Similarly, the power can also be defined with an equation tantamount to the one with the power gain equation using decibel form -

$$P_{dBm} = 10\ log_{10}(\frac{P}{1mW}) \tag{3.5}$$

Typically, the efficiency of a PA, both drain efficiency and PAE, reaches its maximum value at saturated power and decreases rapidly as the power decreases. However, at high power regions, the PA faces a linear problem.

3.3 Power Amplifier Classification

Power amplifiers can be classified as class A, B, AB, C, D, E etc.

Class A amplifiers are most common having a simple structure and highest linearity. It has a typical configuration of an emitter circuit staying 'ON' all the time in order to keep electricity flowing throughout the cycle creating low distortion and high amplitude.

Class B amplifiers can be considered more useful than class A as it has no DC bias thus current does not flow when input signal is in very low range (near 0). [32] It is more efficient than class A as a lot of heat is dissipated because of the flowing current in class A. But Crossover Distortion is a known critical phenomenon that affects class B's performance.

A mix between class A and B results in the class AB. A voltage difference is formed among the base and emitter by positioning two diodes in between. It reduces or sometimes removes the crossover distortion problem of class B but the efficiency is reduced as well which remains in the region between class A and B.

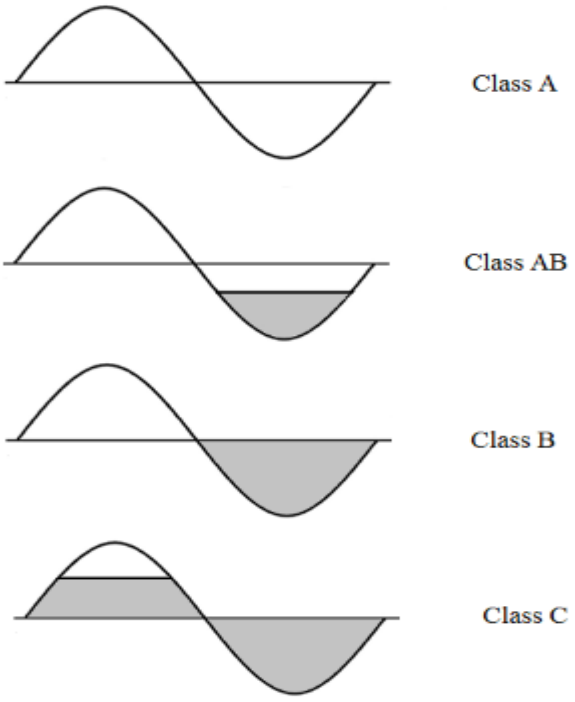

Figure 3.1 Comparison of output signals of different amplifier classes of operation.

Class C, D, E shows high nonlinearity. Both class C and D amplifiers provide almost 100% efficiency in some aspects, works out the lesser half (only active parts) of the input and creates strongly non-linear output. Class C has a negative voltage biased transistor and also shows crossover distortion as class B. In Class D amplifiers, minimum two transistors are used as switches to create an output PWM (pulse width modulation) signal which is then passed through the rst harmonic to the load.

For different amplifiers, signals are demonstrated in the figure 3.1 and linearity in table 3.1.

Efficiency%	Linearity	Class of Operation
50	Good	A
Between Class A and B	Between Class A and B	AB
78.5	Moderate	B
100	Poor	C
100	Poor	D

Table 3.1. Comparison of different class of operation in terms of efficiency and linearity

Though 100% accuracy can be inferred, it is not completely attainable as ideal devices are never made and saturation, junction capacitance and switching speed can cause power loss.

3.4 Behavioral Model of PA

RF-PA models can show memory effects along with its dynamic nonlinear behavior. Behavioral models are designed observing the input output behavior instead of the internal condition which makes it the best choice. There are multitudinous models like the comprehensive, full Volterra model, memory polynomial and its generalized version, Rational function model, Wiener model, Hammerstein model etc. Selection among them is complicated and can be done by comparing their computational complexity, accuracy, derivation techniques etc. Derivation techniques like least square, least mean square or recursive relative square are popular and need ample consideration before selection.

We selected a memory polynomial model as it is less complicated and significantly accurate and an LS method which requires a linear model in its parameters.

3.4.1. Memoryless Model

Memoryless model is named in such a way because it only depends on the ongoing input signal and other signals provided before have no effect. It has an acceptable level of accuracy with easier computational application that can be useful in many cases and thus has been used for a long time in simulating system level. There is one to one mapping among the input and output signals which provides a memoryless aspect which eventually creates less distortion.

Polynomial function model [18] is mostly used along with the Rapping model [20], Ghorbani model [19], Saleh model [8] for this model system and the Look-up Table (LUT) method to structure the model. Look-up table system is a table where necessary data is pre-stored.

3.4.2. Full Volterra Model

The Volterra model, [21,22] A merge of linear convolution and nonlinear power series builds up the Volterra model which offers a way to model a memory included nonlinear system. This model demonstrates the following expression-

$$y(n) = \sum_{p=1}^{P} \sum_{i_1=0}^{M} \cdots \sum_{i_p=0}^{M} h_p(i_1, \cdots, i_p) \prod_{j=1}^{p} x(n - i_j) \tag{3.6}$$

Here, x(n)= input signal, y(n)= output signal, hp (i_1-i_p) = coefficients of Volterra model (Volterra kernels), P= nonlinearity order of the model, M= memory depth.

A higher value of K and M might increase the accuracy limit of the model but it causes unwanted complexity during computation. Nevertheless, it is suited to the nonlinear behavior systems.

3.4.3. Memory Polynomial Model

A reduced version of the Volterra model where only the diagonal terms are included is the memory polynomial model which has been used for its memory effects. There are several types such as Orthogonal [26], Non-uniform [27,28], Envelope memory polynomial model etc. It corresponds to the following equation which is quite similar to the one stated earlier-

$$y(n) = \sum_{m=0}^{M} \sum_{p=0}^{P} a_{mp} x(n - m) |x(n - m)|^p \tag{3.7}$$

Here, a_{mp}= model coefficients

The polynomial order with only the even ones is 5 with a memory depth of 2. This resulted in a 26.00 dB NMSE for this model.

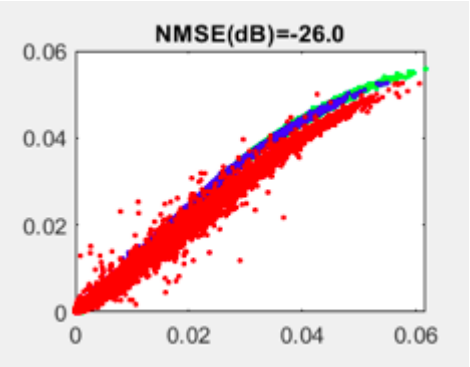

Figure 3.2 Measured and modeled AM-AM characteristics comparison for memory polynomial model. The red represents the measured data and the green is for the model.

3.4.4. Generalized Memory Polynomial Model

Generalized memory polynomial models include the terms excluded in the memory polynomial model from the Volterra model due to the increased range of signal bandwidth. It is an updated advanced model having better performance and less spectral growth than the polynomial model. Though it comes with a complex computational baggage.

It can be described as

$$\sum_{m=0}^{M}\sum_{p=0}^{P} a_{mp}x(n)|x(n-m)|^{p}, \tag{3.8}$$

A local delay of either or both positive and negative m samples are inserted between signal and exponential part and if both are included that are close to the current one, a generalized version will emerge.

$$\begin{aligned} y(n) = &\sum_{p=0}^{P_a}\sum_{l=0}^{L_a} a_{pl}x(n-l)|x(n-l)|^{p} \\ &+ \sum_{p=1}^{P_b}\sum_{l=0}^{L_b}\sum_{m=1}^{M_b} b_{plm}x(n-l)|x(n-l-m)|^{p} \\ &+ \sum_{p=1}^{P_c}\sum_{l=0}^{L_c}\sum_{m=1}^{M_c} c_{plm}x(n-l)|x(n-l+m)|^{p} \end{aligned} \tag{3.9}$$

Here, P_a, L_a; P_b, L_b; P_c, L_c are the polynomial order and memory depth for consecutively current, positive and negative portions and M_b, M_c are the current memory shifted from local memory.

For digital signal processing using MATLAB in the project, a required model is derived which can be stated as the following-

$$y_{BB\_static}(n) = f(g_{BB}(n)) = w_1\, g_{BB}(n) + w_2\, g_{BB}(n)\left|g_{BB}(n)\right|^1 + w_3\, g_{BB}(n)\left|g_{BB}(n)\right|^2 + \ldots$$

$$y_{BB\_dynamic}(n) = f(g_{BB}(n)) + f(g_{BB}(n-1)) + f(g_{BB}(n-2)) + \ldots$$

$$in\ matricial\ form \rightarrow \quad \bar{y}_{BB} = \bar{y}_{BB\_static} + \bar{y}_{BB\_dynamic} = \left[X_{static} \quad X_{dynamic}\right]\bar{w} \tag{3.10}$$

3.4.5. NMSE and ACPR

NMSE

Deviation between the calculated and inferred value can be estimated using Normalized Mean Square Error (NMSE) in the time domain. It can be expressed using the following decibel equation.

$$\mathrm{NMSE} = 10\ log_{10}\left(\frac{\sum_k |y_k - x_k|^2}{\sum_k |x_k|^2}\right) \tag{3.11}$$

X_k= experimental output instant of the DUT

Y_k= output instant from the model

Accuracy in this model will be inversely proportional to NMSE. Therefore, the lower the value, the more accurate results can be developed. It can also derive linear behavior if x_k, y_k is taken as input and output instants for instance –35 dB shows a moderate linearity.

ACPR

To observe the distribution of power among the in-band and adjacent channels, the power ratio between them is calculated which is known as the Adjacent Channel Power Ratio (ACPR). The error can be quantified in a variety of frequency domain ranges. It is given as-

$$ACPR = 10\ log_{10}\left(\frac{\int_{f_1}^{f_2}|E(f)|^2\ df + \int_{f_3}^{f_4}|E(f)|^2\ df}{\int_{f_2}^{f_3}|E_{desired}(f)|^2\ df}\right) \quad (3.12)$$

Here, $|E(f)|^2$ = power at one frequency; range f_1-f_2 = lower adjacent channel; range f_3-f_4 = upper adjacent channel; range f_2-f_3 = in-band channel

The following figure shows the ACPR output signal in the frequency range of –14MHz to 14MHz.

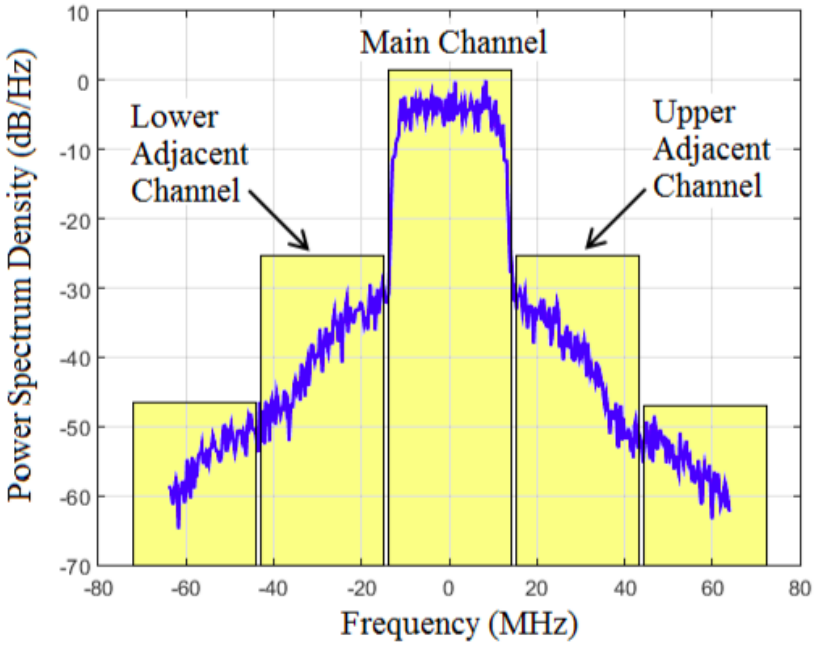

Figure 3.3 ACPR figure of a sample PA output

CHAPTER 4. DPD

DPD has gained preference over other techniques because of its simplicity and better performance. It is markedly different from techniques such as feedforward or feedback as linearization is gained by positioning a nonlinear block in front of the PA in DPD. This block is named as a predistorter and it shows inverse behavior of that of the Power Amplifier

4.1. Baseband DPD

The preceding study of distortion is aimed at designing architectures capable of reducing them. DPD is mostly assigned in this segment as it shows promising amenities including accuracy, singularity, digital signal processing techniques etc. The whole system is in just one block and it omits the necessity of re designing.

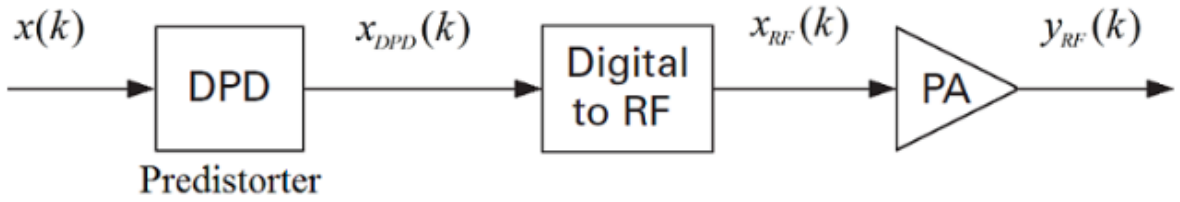

Figure 4.1 Digital transmitter with digital predistortion

To implement DPD techniques, FPGA has advantages in digital signal processing such as high speed and reliability, exible performance and parallelism computation. LUT technique shows efficacy in predistortion as well. Although FPGA is preferred but the aforementioned processing is not limited to FPGA.

This chapter discussed the basics of digital predistortion with comparison between both direct and indirect learning architectures for DPD. It successfully improves the linearity to the point it is required but also comes with baggage. It causes bandwidth expansion and energy consumption issues.

4.2. Basic Principle

The basic theoretical principle has been discussed earlier about DPD where we already stated its necessity for linearization and removal of distortions. A nonlinear component named as a predistorter is placed before a power amplifier which works just the opposite way in terms of phase and magnitude so that it can counter the nonlinear behavior and implements linearity.

It can also be viewed considering the intermodulation distortion since PA produces them. Thus, if the amplitude is equal with a 180 degree change in phase, it will

cancel out the distortion. Figure 4.2 shows the anti-phase with downward arrow.

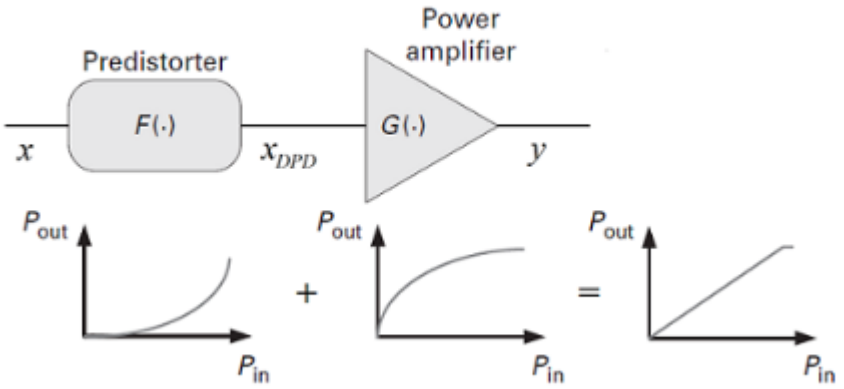

Figure 4.2 Basic principle of digital predistortion

Mathematically, terms of DPD are discussed below,

Pre Distorted signal $x_{DPD} = F(x)$; which can also be denoted as $x_{DPD} = G_{DPD}(x).x$

Here, x = input signal; $G_{DPD}(x)$ = nonlinear gain of predistorter at x or can also be said to be the slope of transfer function among x_{DPD} and x.

The output signal can be expressed as follows,

$y = G(x_{DPD}) = G(F(x))$ or,

$y = G_{PA}(x_{DPD}).\ x_{DPD}$

Here, $G_{PA}(x_{DPD})$ = nonlinear gain of PA or can also be said to be the slope of transfer function among x_{DPD} and y.

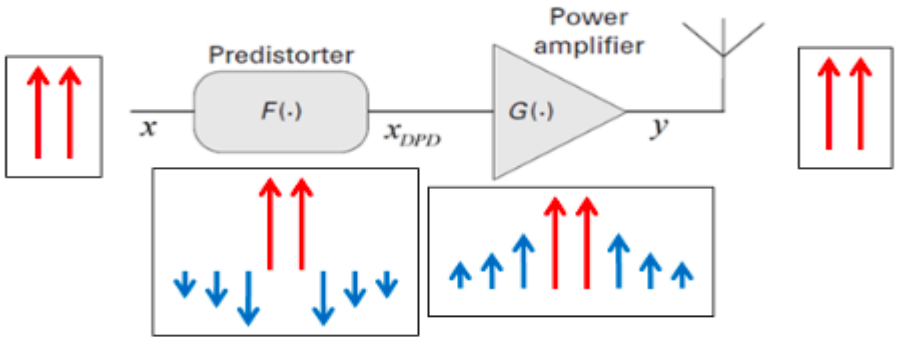

Figure. 4.3 Frequency domain interpretation of digital predistortion

PA and predistorter shows opposite behavior in nonlinearity, and taking the output signal y being normalized by the power gain of PA, it can be inferred that,

$G_{DPD} = 1/ G_{PA}$

Therefore, the gain for the entire system can be expressed as the following equation-

$G = dy/dx = dy/dx_{DPD} \cdot dx_{DPD}/dx = G_{PA}. G_{DPD} = 1$

It can be observed that the gain is a normalized constant value 1 which indicates the linearity of the system.

A practical applied example is demonstrated below in figure 4.4 and 4.5 where the real data measured from PA is valued and the AM/AM and AM/PM characteristics are shown for a PA (blue), predistorter (green) and also the complete system including both of them (red).

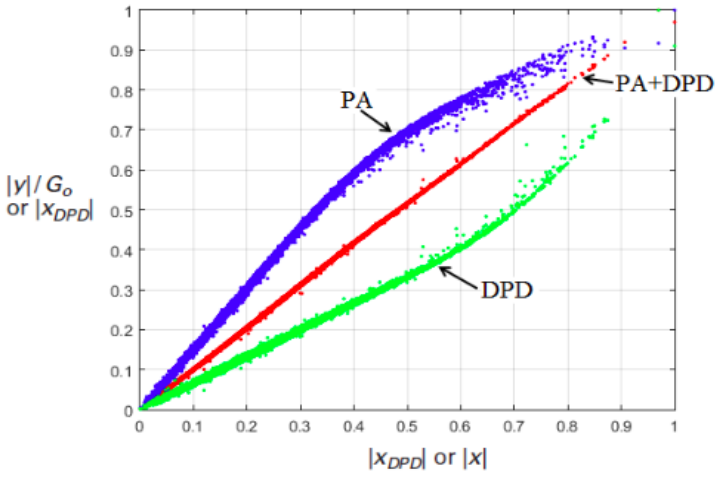

Figure. 4.4 Transfer functions for a sample PA with memory (blue), predistorter suitable to obtain linear response (green), PA and predistorter(red)

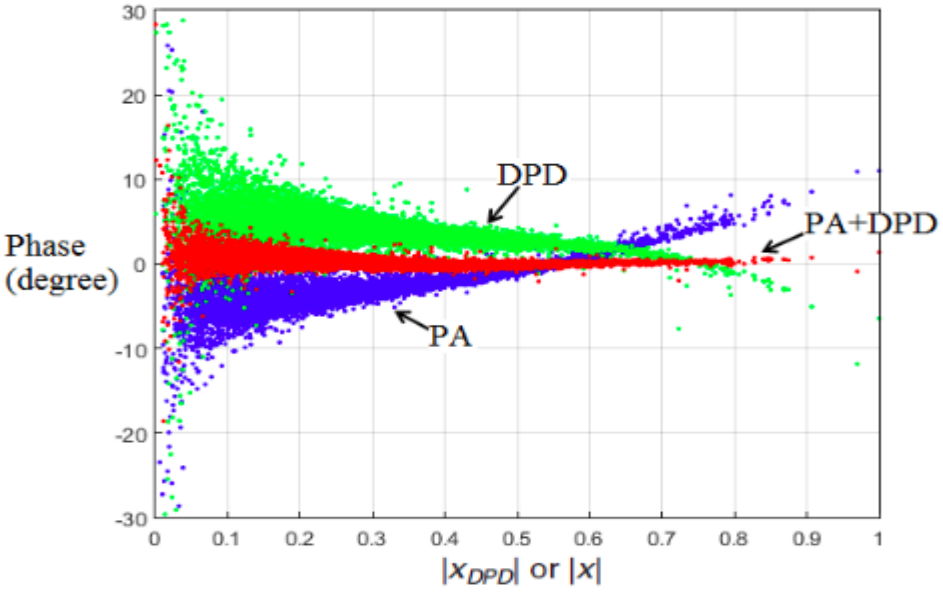

Figure. 4.5 Phase curves for a sample PA with memory (blue), predistorter suitable to obtain linear response (green), PA and predistorter(red)

In fig 4.4, the vertical and horizontal axis illustrates consecutively the magnitude for the predistorter and PA (normalized) output signal and the input to the predistorter and the PA.

It can be evaluated from the figure that the magnitude is not straight but causes a compressed power gain at higher regions which on the contrary is compensated by the gain expansion caused by the predistorter leading to a straight line which in this case shows linear behavior.

4.3. DPD Architecture

Digital predistortion technique introduces a reverse characteristic of the PA by deriving an inverse model using an inverse function with input and output of PA. There are direct and indirect architectures for DPD.

4.3.1. Direct Learning Architecture

The direct learning architecture, as shown in figure 4.6 works directly using feedback error e(n) with adaptation algorithm to adjust the parameters of a predistorter model where e(n) demonstrates the gap between the time aligned input and output signal x(n) and z(n). Among the plethora of adaptive systems proposed [11-13], as a model reference adaptive system the Direct learning method by braithwaite [1] is discussed in the following sections.

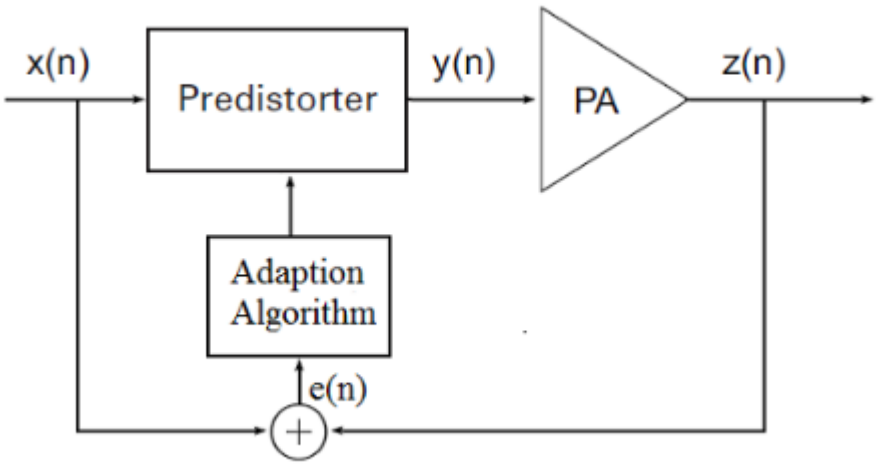

Figure 4.6 Block diagram of the direct learning architecture

which are time aligned. Several adaptation algorithms [11-13] have been proposed for direct learning architecture. The Direct Learning method proposed by Braithwaite [32] as a Model Reference Adaptive System (MRAS) is analyzed below [47].

The error e[n] and the estimated error $\hat{e}[n]$ are defined as,

$$e[n] = z[n]G_o^{-1} - x[n], \tag{4.1}$$

$$\hat{e}[n] = F_u \Delta w_i, \tag{4.2}$$

respectively, while G_o represents the linear gain of the PA, F_u the basis waveforms matrix constructed with the original input signal u[n] and Δw_i is the coefficients vector computed at the i^{th} iteration. Then, the cost function to be minimize is the square of the difference between the real error and the estimated one

$$J[n] = |e[n] - \hat{e}[n]|^2 \tag{4.3}$$

Knowing the PA output, then it is possible to estimate the least-squares solution to calculate the estimated error and the coefficients needed for the adaptive process:

$$\Delta w_i = (F_u^H F_u)^{-1} F_u^H e, \tag{4.4}$$

$$w_{i+1} = w_i + \mu \Delta w_i, \tag{4.5}$$

The weight factor μ is a value between 0 and 1, usually reduced at each iteration as it converges. Therefore, the additive distortion

$$dw[n] = F_u w_i, \tag{4.6}$$

and the new PA input signal will be

$$y[n] = x[n] - dw[n] \tag{4.7}$$

It must be considered in first iteration,

$$w_1 = \Delta w_0 \text{ and } y[n] = x[n] \quad (4.8)$$

4.3.2. Indirect Learning Architecture

The indirect learning architecture is yet another method for the extraction of model parameters. It uses a post predistorter which has a lookalike nonlinear transfer function of a post-inverse estimation block. This estimation block minimizes the error signal e(n) and generates the Pd parameters which is then copied to the predistorter.

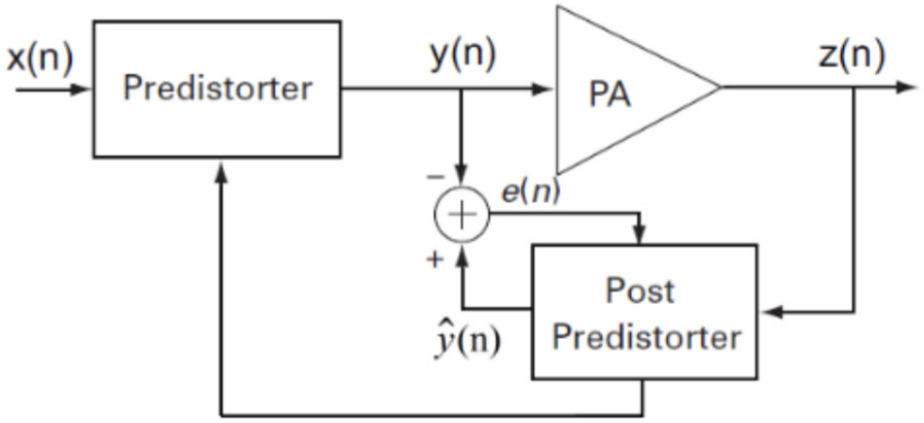

Figure 4.7 Block diagram of the indirect learning architecture

As shown in figure 4.7, the input and output to Pd is stated as x(n) and y(n); z(n) as the normalized output from PA and ÿ(n) is the output of the post predistorter block. The e(n) can be derived as follows- e(n) = y(n) - ÿ(n)

Also as the power amplifier is linear, x(n) = z(n) which also concludes y(n) = ÿ(n)

CHAPTER 5. IMPLEMENTATION of DPD in MATLAB and PYTHON

5.1. AD9361 Testbench

The AD9361, formulated to be applied in 3G and 4G base station applications is an ideal agile transceiver for a broad range application for its high performability, programmability, wideband capacity and integrated radio frequency. It comes in a 10X10 mm package, 144-ball chip scale package ball grid array. An RF front end combined with a malleable mixed-signal baseband section, configurable digital interface to the processor to simplify the design-in and integrated frequency synthesizers makes up the complete device.

This testbench has externally manageable, flexible and manual gain modes. To digitize the signals, there are two high dynamic range convertors (ADCs) for each channel are used where the signals I and Q received are passed through the filters (configurable decimation and 128-tap finite impulse response) which produces an output signal of 12-bit in the required rate.

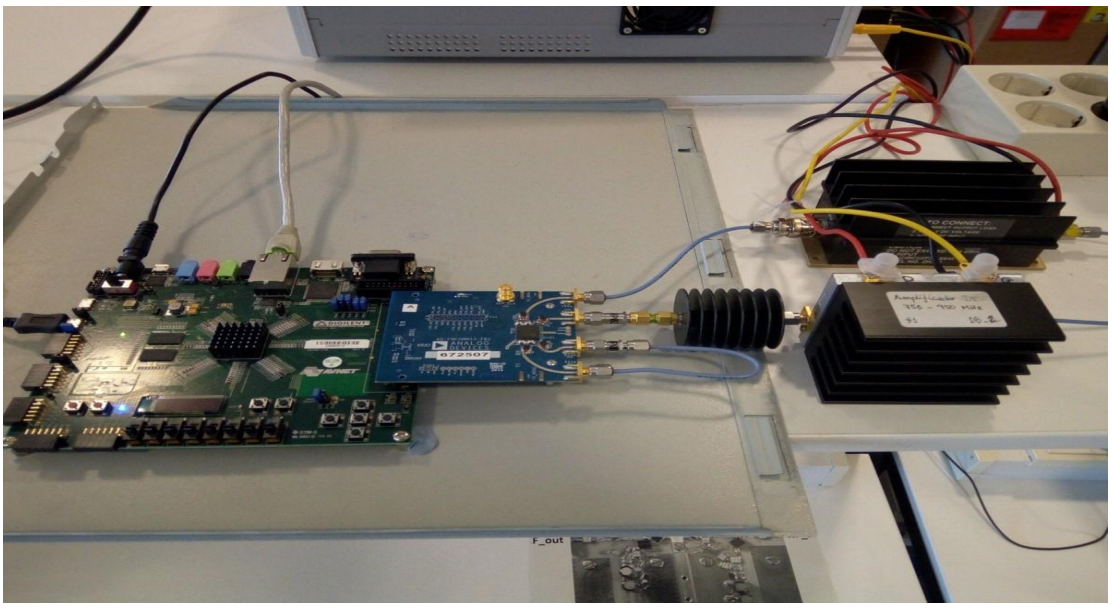

Figure 5.1 AD9361 testbench.

The receiver (70MHz – 6GHz) and transmitter (47MHz – 6GHz) LO operate in a range which covers all bands- licensed or unlicensed and less than 200KHz to 56MHz channel bandwidths are also supported. There are direct conversion

receivers working independently that integrate linearity and state of the art noise figure. These receiver subsystems incorporate automatic gain control, dc offset correction, digital filtering, quadrature correction which eventually eliminates the necessity of devices having these functions.

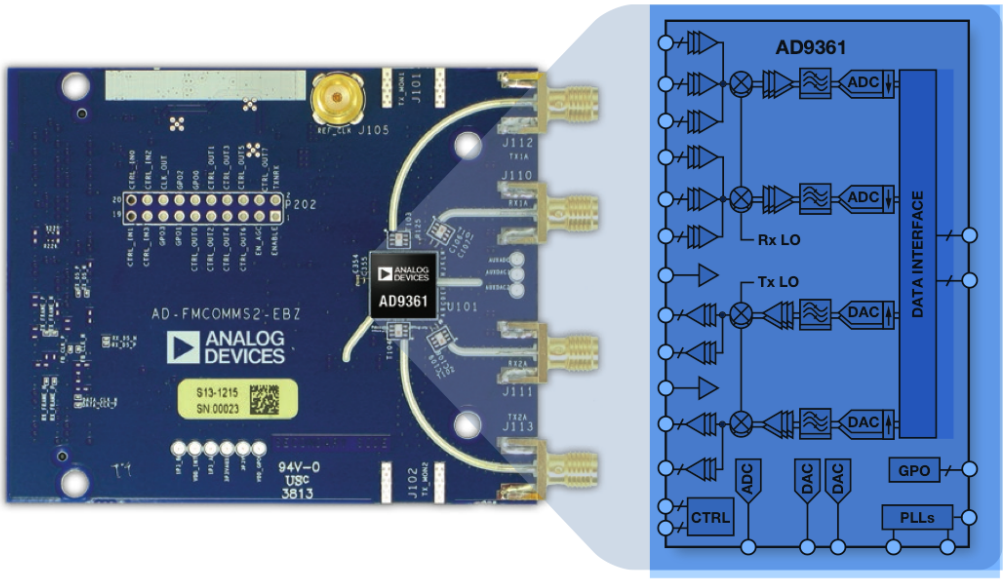

Figure 5.2 AD-FMCOMMS2-EBZ board

High modulation accuracy and ultralow noise can be achieved using a direct conversion architecture in the transmitters. Error vector magnitude of about < -40dB offers good margin for selecting external PA. The onboard transmit power monitor provides higher accuracy in TX power measurements using it as a power detector whereas low power fractional-N frequency synthesis for all receiving and transmitting channels can be deduced with the fully integrated phase-locked loops. Channel separation for the frequency division duplex is included along other integrated VC and loop filters.

The AD9361 core can be powered straight from the 1.3V regulator while controlling the IC with a standard 4-wire serial port and input/output control pins which come with comprehensive power-down modes to reduce power consumed.

5.2. Software description of Matlab

We had a computer with MATLAB software where we installed “Communications Systems Toolbox” and “Signal Processing Toolbox” to create M-QAM modulation. We also installed “Instrument Control Toolbox” to use AD9361 remote hardware which required internet connection. All the Matlab scripts were stored in “Matlab_DEMO_IQ_AD9361” in the UPC LAB. Now, opening and running the “MAIN_IQ.m” in Matlab will show us the several plots created beforehand. Various

IQ signals will be created and analyzed in the time domain (constellation) and spectral one. Several signals like tone modulation, random IQ and M-QAM can be generated using Matlab

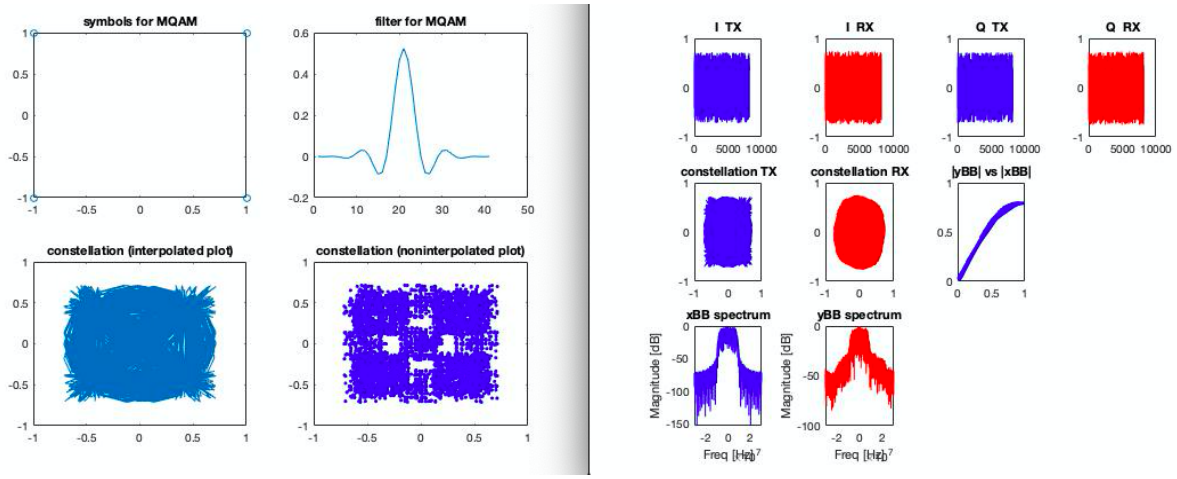

Figure 5.3 Constellation of MQAM, TX and RX

The software basically works in the following way-

- Generation of baseband IQ signal xBB (the general IQ is stored in xBB)
- Sending xBB to a transmitter (transmitter code is done in the Matlab function 'dofn_TXRX.m") TX which will process if it's a hardware transmitter (such as AD9361)
- Receiving a baseband yBB signal corresponding to a down-conversion of RF signal received in AD9361

The signal processing in second step can be done for,

➔ Up-converting to radio frequency
➔ Amplification by PA
➔ Delivering to antenna

Transmitting and Receiving action can be done in two particular ways one being Simulated TXRX 'PA_SIM01"

if we Open the file "MAIN_IQ.m" and take a look inside the code. This Matlab script is able to generate several types of IQ signals: tone modulation, random IQ, and M-QAM. The generated IQ signal is stored in the variable xBB. The generated baseband IQ xBB can be transmitted to the input of a communications transmitter (the transmitter code is done in the Matlab function "dofn_TXRX.m"). The received IQ signal, also in baseband, is stored in the complex array yBB. Basically, the software capabilities are the following: Generating a baseband IQ signal xBB.

Tx: Sending xBB to a transmitter TX (true or simulated). In the case of sending xBB to a hardware transmitter (as the AD9361 one), the hardware will process the IQ xBB signal for: i) upconverting to RF (radiofrequency), ii) amplify it by means of a PA

(Power Amplifier) and iii) sending to an antenna.

RX: Receiving a base band signal yBB, that corresponds to the downconversion of the RF signal received in the AD9361. There are two options for transmitting xBB and receiving yBB signals. One is with Simulated TXRX 'PA_SIM01', another one is with

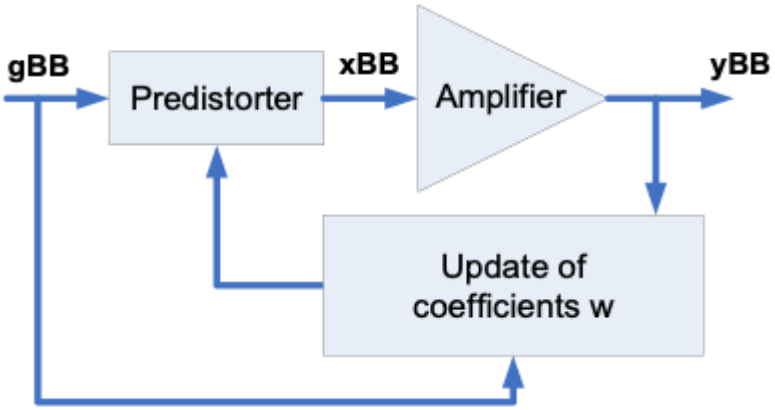

Figure 5.4 Predistortion scheme.

5.3. Implementation of Static DPD in Matlab

In order to apply the static DPD in Matlab, the Main.IQ file has to be run followed by the script “do_DPD.m” which will carry out five iterations of predistortion consisting the following parameters- gain desired, gain desired back off, eBB, gBB (input), yBB (output), dw, μ.

The mathematical expression is shown below,

$$y_{BB\_static}(n) = f(g_{BB}(n)) = w_1\, g_{BB}(n) + w_2\, g_{BB}(n)\left|g_{BB}(n)\right|^1 + w_3\, g_{BB}(n)\left|g_{BB}(n)\right|^2 + \ldots \quad (5.1)$$

Converting in Matlab, we get the following equation,

V = [ones(PARAM.L.LBB,1), agBB, agBB.^2, agBB.^3, agBB.^4, agBB.^5]; (5.2)

The output after each iteration is demonstrated below where the blue, cyan and red lines represent iteration 1, iteration 2-5 (of DPD) and PA. It can be observed that the linearity increased with every iteration with the evolution of DPD. In figure 5.6 it indicates that fifth iteration led to linear |yBB| and |gBB|, with six DPD coefficients.

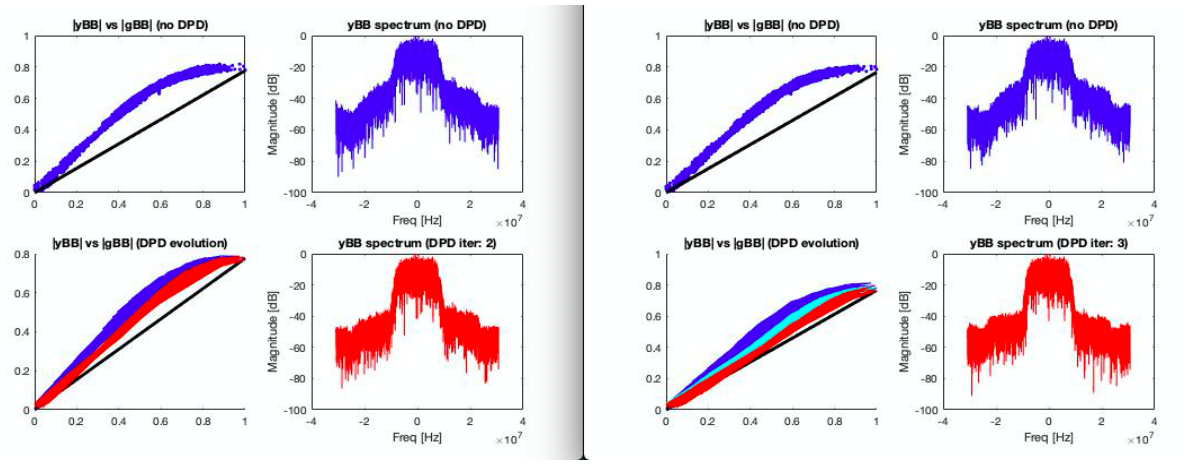

Figure 5.5 absolute value of yBB Vs gBB and spectrum of yBB with and without DPD for iteration 2 and 3.

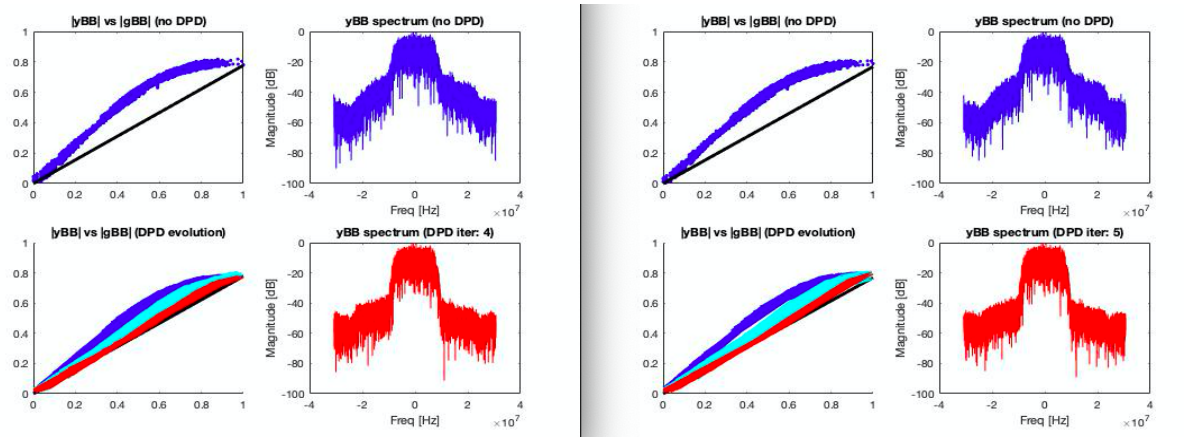

Figure 5.6 absolute value of yBB Vs gBB and spectrum of yBB with and without DPD for iteration 4 and 5.

The NMSE values are given in the following table,

Iteration	NMSE Value in dB
1	-21.649490
2	-25.574182
3	-27.079678
4	-26.247182
5	-27.163957

Table 5.1: NMSE evolution after 5 iterations of Static DPD

5.4. Implementation of Dynamic DPD in Matlab

The code in this case needs to be tempered into a dynamic version. The script needs to be changed into “do_DPD_memory” so that memory effects are implied. Therefore, we have to build an X matrix with consideration of some static terms and some others relating to delays (tantamount to nonlinear FIR filter). “Circshift” function is used to create these delayed terms. Finally, the new code for dynamic DPD looks like the following equation,

```
Xstatic=gBB.*[ones(PARAM.L.LBB,1), agBB, agBB.^2, agBB.^3, agBB.^4, agBB.^5];
Xdelay1=circshift(Xstatic,[1 , 0]);
Xdelay2=circshift(Xstatic,[2 , 0]);
X=[Xstatic , Xdelay1 , Xdelay2];
```

(5.3)

The iterations are done in the similar way five times as the static DPD and outputs are observed and illustrated below,

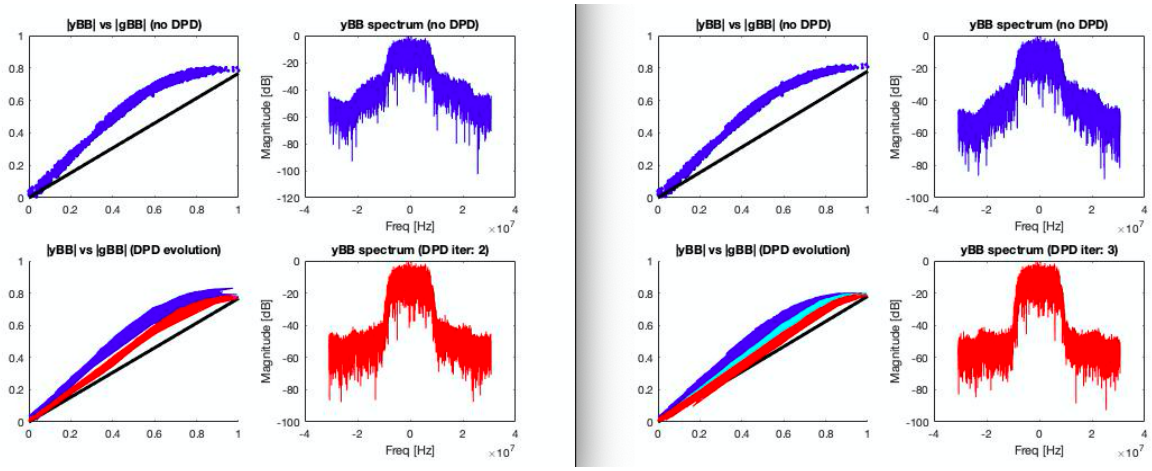

Figure 5.7: absolute value of yBB Vs gBB and spectrum of yBB with and without DPD for iteration 2 and 3.

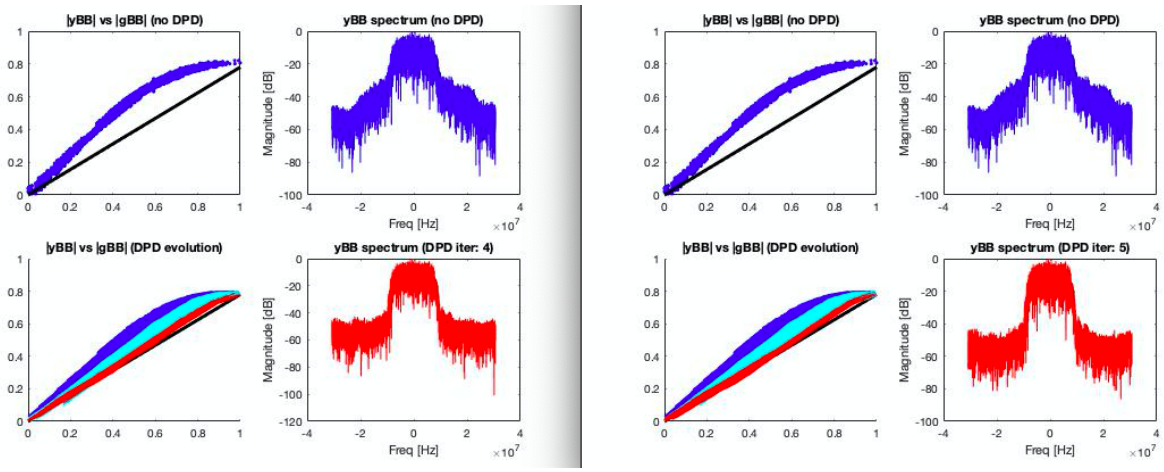

Figure 5.8: absolute value of yBB Vs gBB and spectrum of yBB with and without DPD for iteration 2 and 3.

The graphs show the same parameters with the corresponding color codes as the static DPD figures and outcomes are also similar regarding the linearity. Though in this case, there are 18 DPD coefficients. The NMSE values are shown below,

Iteration	NMSE Value in dB
1	-21.153586
2	-29.148059
3	-30.102704
4	-30.180556
5	-30.865795

Table 5.2: NMSE evolution after 5 iterations of Dynamic DPD

5.5. Implementation of Static DPD in Python

The python path needs to be configured in Matlab. Three paths for python have been created- own path, Module path and Python path. After creating a folder named as “Python_scripts”, the Matlab scripts are converted into python scripts and then called from Matlab. We can see the following form the figure.

```
[own_path, ~, ~] = fileparts(mfilename('fullpath'));
module_path = fullfile(own_path, './python_scripts/');
python_path = py.sys.path;
fprintf("Python_path: %s", python_path);
if count(python_path, module_path) == 0
    insert(python_path, int32(0), module_path);
end
```

Figure: 5.9 script of Python path to call from Matlab

The calculation part is given in figure 5.10

```
w = mat_contents['w']
gain_desired = mat_contents['gain_desired'].flat[0]
gain_peak_backoff = mat_contents['gain_peak_backoff'].flat[0]
yBB = mat_contents['yBB']
gBB = mat_contents['gBB']

mu = 0.4
yBB = np.array(yBB)
gBB = np.array(gBB)

mul = np.multiply(gain_desired, gBB)
sub = np.subtract(yBB, mul)
eBB = np.divide(sub, gain_desired)
agBB = np.absolute(gBB)

X = np.multiply(
        gBB,
        np.concatenate((
            np.ones((8192, 1)),
            np.power(agBB,1),
            np.power(agBB,2),
            np.power(agBB,3),
            np.power(agBB,4),
            np.power(agBB,5)
        ), axis=1)
    )

dw = np.linalg.lstsq(X, eBB, rcond=-1)[0]
w = np.add(w, np.dot(mu, dw))
```

Figure: 5.10: A part of python script for static DPD calculation

Scipy and Numpy – these two libraries are used to convert the matrix algorithm and complex computation in python. Both are free sources, scipy is used for scientific and technical computing whereas numpy can add support for multitudinous and multidimensional matrices.

Taking the value of w, the gain desired, gain desired peak back off, gBB, yBB, μ etc are calculated. As data types are different, matrix calculations are diverse. Calculation process is the same as Matlab but python is not as efficient in complex calculations. This is why complex numbers need to be changed. Matlab values are used to calculate in python and the output is then again used by Matlab. Now the problem that raised is that it is not feasible to directly call python from Matlab, so a different approach was necessary. A data file named ".mat" like "mat2py.mat" and "py2mat.mat" are created to dump necessary data, store the value of some functions as dw, w, μ, eBB, gBB(input), yBB(output) etc. This will enable python to read the value and redo the calculation.

```
%%%%% update
%{
save('mat2py.mat')
py.dpd.do_calculation();
py_w = load('./py2mat.mat','w');
py_xBB = load('./py2mat.mat','xBB');
w = py_w.w;
xBB = py_xBB.xBB;
delete('mat2py.mat');
delete('py2mat.mat');
%}
```

Figure: 5.11 Dump file for exchanging data between Matlab and Python

The full results after calculation are returned in the ".mat" file named "py2mat.mat". For every iteration, matlab dumps the data in "mat2py.mat" and python returns it in "py2mat.mat". Results of iteration 2-5 of static DPD in Python are shown in the

following figure along with NMSE evolution in the table. Number of coefficients is 6.

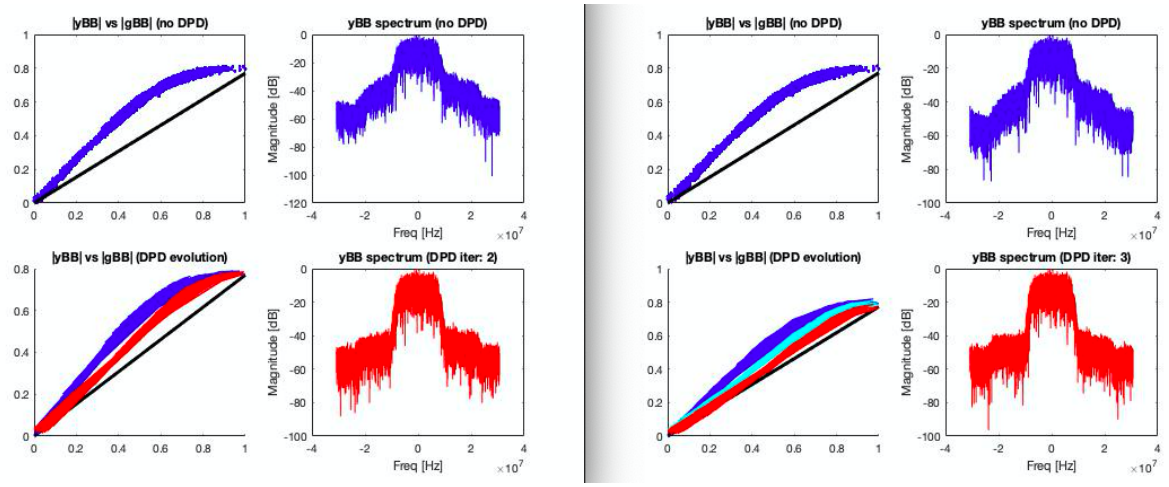

Figure 5.12: absolute value of yBB Vs gBB and spectrum of yBB with and without DPD for iteration 2 and 3 in Python. (Static DPD in Python)

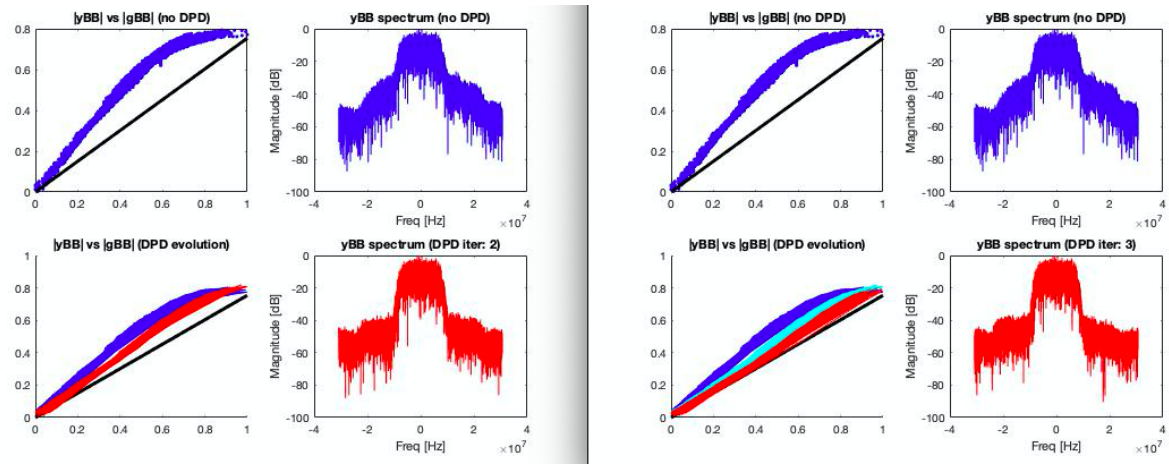

Figure: 5.13 absolute value of yBB Vs gBB and spectrum of yBB with and without DPD for iteration 4 and 5. (Static DPD in Python)

Number of coefficients are 6. The NMSE evolution after 5 Iterations are,

Iteration	NMSE Value in dB
1	-20.898713
2	-25.732465
3	-27.466506
4	-26.443209
5	-26.749821

Table 5.3: NMSE evolution after 5 iterations of Static DPD in python

5.6. Implementation of Dynamic DPD in Python

Implementation of Static and Dynamic DPD processes is mostly similar; just additional lines in the static DPD scripts are required as stated in the following parts.

During calculation, “np.multiply”, “np.array”, “np.concatenate”, “np.linalg.lstsq”, “np.divide”, “np.subtract” are given for respectively multiplication, matrix, aggregating matrix, linear algorithm, division and subtraction. After calculation, the data is stored in “mem_py2mat.mat” where Matlab reads the output results of python form and plots the graph accordingly.

```
Xdelay1 = np.roll(Xstatic, 1);
Xdelay2 = np.roll(Xstatic, 2);
X = np.concatenate((Xstatic, Xdelay1, Xdelay2), axis=1);

dw = np.linalg.lstsq(X, eBB, rcond=-1)[0]
w = np.add(w, np.dot(mu, dw))

xBB_correction = np.dot(X, w)
xBB = np.subtract(np.dot(gain_peak_backoff, gBB), xBB_correction)

sio.savemat('./mem_py2mat.mat', {'w': w, 'xBB': xBB})
```

Figure 5.14 Added extra lines for Dynamic DPD

Finally, after iterating five times, the output of dynamic DPD can be seen as below,

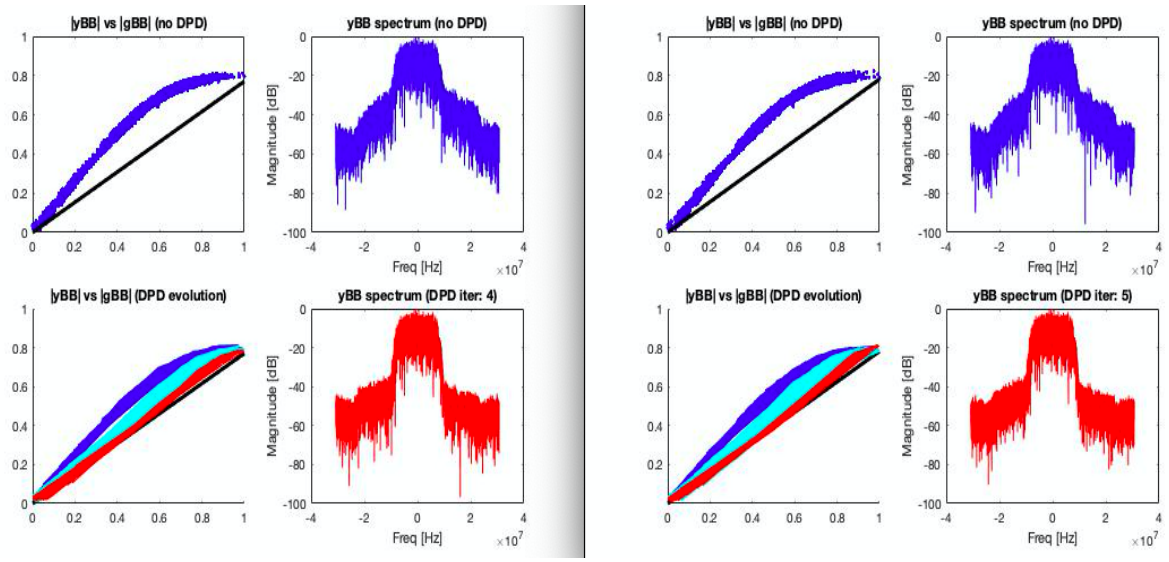

Figure: 5.15 absolute value of yBB Vs gBB and spectrum of yBB with and without DPD for iteration 2 and 3. (Dynamic DPD in Python)

It is evident from the figures that the linearity increased with the iteration number for the effect of DVD evolution. If NMSE values are compared, for python we got –28.541522 after 5th iteration whereas for Matlab it was –30. 865795. The NMSE evolution for the iterations are given in the following table. Number of co-efficient are 18.

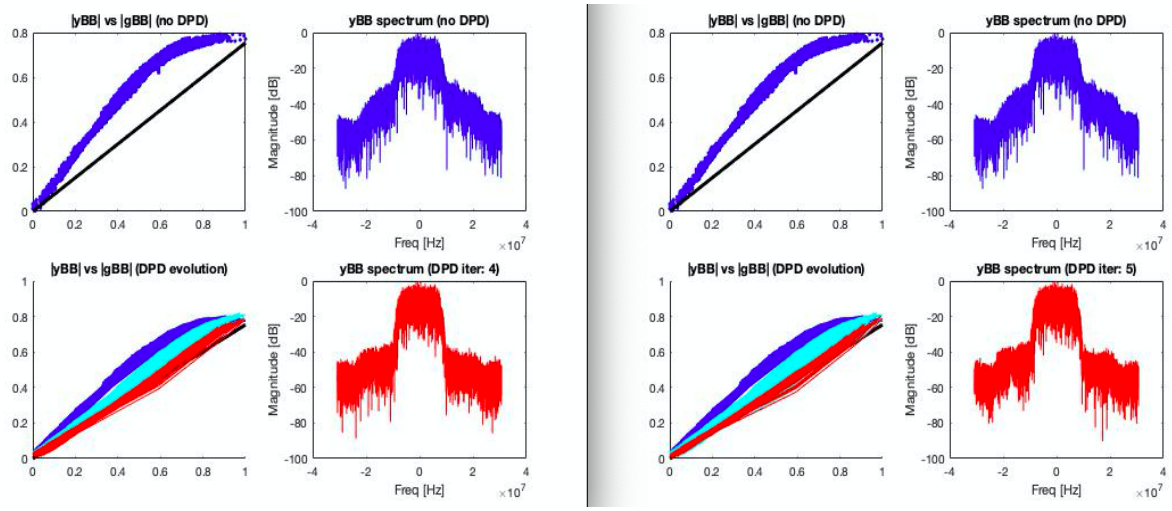

Figure: 5.16 absolute value of yBB Vs gBB and spectrum of yBB with and without DPD for iteration 4 and 5. (Dynamic DPD in Python)

The NMSE evolution after 5 Iterations are,

Iteration	NMSE Value in dB
1	-20.681254
2	-25.724949
3	-28.093849
4	-27.265225
5	-28.541511

Table 5.4: NMSE evolution after 5 iterations of Dynamic DPD in python

5.7 Numpy and Scipy in Python

linalg.lstsq:

Return the least-squares solution to a linear matrix equation.

Rcond float,

np.linalg.lstsq(X, eBB, rcond=-1)[0]

Cut-off ratio for small singular values of dw. For the purposes of rank determination, singular values are treated as zero if they are smaller than rcond times the largest singular value of dw.

Changed in version 1.14.0: If not set, a Future Warning is given. The previous default of -1 will use the machine precision as rcond parameter, the new default will use the machine precision times max(X, eBB). To silence the warning and use the new default, use rcond=None, to keep using the old behavior, use rcond=-1.

Raises

If computation does not converge.

If eBB is a matrix, then all array results are returned as matrices.

np.concatenate:

To determine the rank, if singular values are smaller than rcond times the greatest value for X, they are treated as zero.

The modifications in version 1.14.0 : A future warning is provided unless set. Instead of using the machine precision as rcond parameter, the new default will use mas(M, N). Warning silencing and establishing the new default can be done using rcond = None, otherwise rcond = -1 gives the old behaviour.

Unless computation converges.

eBB being a matrix, the array results will provide matrices

The main component of NumPy is the homogenous multidimensional array that includes all similar elements indexed by a tuple of positive integers organized in a table. In NumPy, the dimensions are introduced as Axis.

The array class for NumPy is stated as ndarray or alias array. This is divergent from the typical python library class array which only administers one directional array with less functionality.

The following parameters are included in the NumPy function.

1. Any object exposing array interface whose _array_ approaches give back any array or nested sequence.

2. To upcast the array, this parameter is used which characterizes the parameters desired for the element of the array. Unless data type is defined, this processes the data type as the minimum type necessary to hold the object in the sequence.

3. Providing that the copy is set to true, it is copied. If not, it will be copied when it is required for other types as dtype, order etc or while an object is a nested sequence.

4. The memory layout of the array is specified by this parameter. When F is enumerated, the newly created array will follow Fortran order, if not, it will follow C order. If the object is an array, it holds the following order.

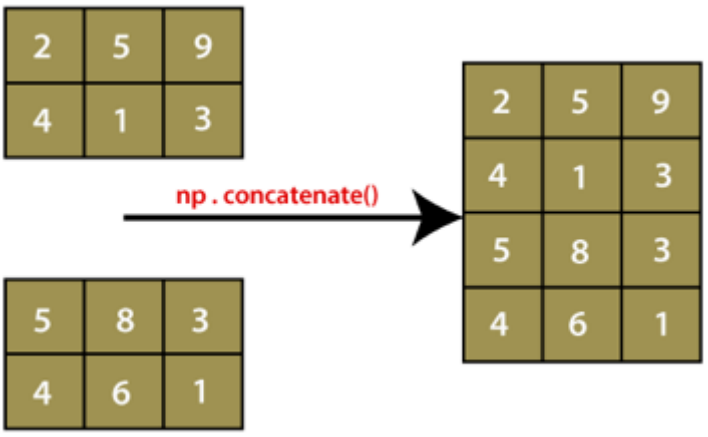

Figure: 5.17. An example of array joining through np.concatenate()

The concatenate () function, an essential one from the NumPy package, for combining arrays together is used to merge similar shaped arrays with specified axis. The following facts are to be followed in this case:

a. Rather than being a traditional database, NumPy concatenate () is like stacking NumPy arrays.

b. The concatenation process can be performed both vertically and horizontally.

The function is written as np.concatenate or can be written also numpy.concatenate depending on the way it's imported as np or numpy.

Parameter:-

1. (Xstatic, Xdelay1, Xdelay2…) This states the sequence of the arrays having similar shape but variation in dimension with respect to the axis.

2. Axis : int(optional) This parameter designates the axis along which the joining of the array is done and this gives a default value of 0.

Apart from the np.concatenate, np.linalg.lstsq we also used some regular functions for calculations. They are np.multiply, np.subtract, np.divide and np.absolute.

We tried to replace np.concatenate with Np.dstack, No.vstack, No.hstack and np.stack. But they weren't able to provide the desired results. So, we struck with the function np.concatenate.

5.8. DPD Comparison in Matlab and Python

After completion of five iterations in both Matlab and Python, the observed results show similarity in the graphs and NMSE(static DPD). Hence, it can be concluded that the efficiency of Python is tantamount to that of Matlab and thus, python can be used alongside Matlab for future performance evaluation of DPD. On another note, the values for dynamic DPD NMSE values, slight differences can be observed in Matlab and Python as stated in the preceding section (section 5.6).

Iteration (Matlab)	NMSE (dB)	Iteration (Python)	NMSE (dB)
1	-21.649490	1	-20.898713
2	-25.574182	2	-25.732465
3	-27.079678	3	-27.466506
4	-26.247182	4	-26.443209
5	-27.163957	5	-26.749821

Table 5.5 Comparison of NMSE evolution after 5 iterations of Static DPD in Matlab and Python

The outcome of NMSE value after 5 iterations of static DPD in both Matlab and Python are almost Identical. Which is also a proof that Python can provide a Matlab-like performance for DPD.

But for the case of dynamic DPD there is a little difference in NMSE value between Matlab and Python. After the 5th iteration the NMSE value for Dynamic DPD in python is -28.541511 and in Matlab is -30.865795.

Iteration (Matlab)	NMSE (dB)	Iteration (Python)	NMSE (dB)
1	-21.153586	1	-20.681254
2	-29.148059	2	-25.724949
3	-30.102704	3	-28.093849
4	-30.180556	4	-27.265225
5	-30.865795	5	-28.541511

Table 5.6 Comparison of NMSE evolution after 5 iterations of Dynamic DPD in Matlab and Python

CONCLUSION

Our prime objective was to match the performance of MATLAB for DPD using DSP in Python which we were able to achieve, with better outcome for the case of static DPD. From operational principles to behavioral modelling, almost all aspects of Digital predistortion techniques were discussed and we can conclude that with this new process, the amplification in transmitters will be economical with low cost amplifiers and provide higher returns in the wireless system industries.

DPD has been widely accepted as one of the fundamental units in modern and future wideband wireless systems. This technique also mitigates the linearity problem in PAs operating in the saturation region which leads to more flexibility in designing RF structure architecture. This opens up the opportunity to provide better user experience and low power waste ensuring green communications in wireless access.

5.9. Suggested work

Software Defined radio (SDR) techniques enable end-user devices such as mobile phones the ability to change radio protocols in real time. An optimal SDR system requires a wideband analog front-end with a highly efficient switch-mode wideband RF PA. But due to their squarewave-like transmission mechanism, switch-mode PAs suffer from very strong nonlinear distortion, which can be an open issue for DPD research in terms of new behavioral model development and corresponding parameter-extraction schemes.

Most of the DPD solutions are proposed for medium-to-high PAs, which are normally utilized in middle-to-large-size base stations. For small base stations, mobile handsets, the forthcoming small-cells based wireless networks, low-power RF amplifiers still suffer from lower efficiency because of the use of power back-off to control the inherent nonlinear distortion. Those small-size devices and associated equipment may use different power levels under different situations. Therefore, another open issue is how to perform low-power, real-time DPD that can significantly enhance the power efficiency and prolong the battery life of these small wireless systems.

ACRONYMS

AI	Application Interface
ACPR	Adjacent Channel Power Ratio
ADC	Analog to Digital Converter
DAC	Digital to Analog
DSP	Digital Signal Processing
DUT	Device Under Test
DPD	Digital Predistortion
ETSI	European Telecommunications Standards Institute
FPGA	Field Programmable Gate Array
IMD	Intermodulation Distortion
LMS	Least Mean Square
LUT	Look Up Table
LS	Least Square
NMSE	Normalized Mean Square Error
NP	Numpy
OFDM	Orthogonal Frequency Division Multiplexing
PA	Power Amplifier
PAE	Power Added Efficiency
PAPR	Peak to Average Power Ratio
PD	Pre Distorted
QAM	Quadrature Amplitude Modulation
RFPA	Radio Frequency Power Amplifier
RLS	Recursive Least Square
RTL	Register Transfer Level

BIBLIOGRAPHY

[1] P. B. Kennington, High Linearity RF Amplifier Design. Norwood, MA: Artech House, 2000

[2]https://www.rfwireless-world.com/Articles/SDR-Software-Defined-Radio-basics.htm

[3] S. C. Cripps, RF Power Amplifiers for Wireless Communications. Norwood, MA: Artech House, 1999.

[4] R. W. Erickson, and D. Maksimovic, Fundamentals of Power Electronics, 2nd ed. Norwell, MA: Kliwer, 2000.

[5] H. Krauss, C. Bostian, and F. Raab, Solid State Radio Engineering. New York: Wiley, 1980, pp. 432-467.

[6] Fa-Long Luo, Digital Front-End in Wireless Communications and Broadcasting: Circuits and Signal Processing. Cambridge: Cambridge University Press, 2011. 6

[7] K. Hausmair, P. N. Landin, U. Gustavsson, C. Fager and T. Eriksson, "Digital Predistortion for Multi-Antenna Transmitters Affected by Antenna Crosstalk," IEEE Trans. Microw. Theory Tech., vol. 66, no. 3, pp. 1524-1535, March 2018.using envelope injection technique," IEEE Radio and Wireless Conference, Aug. 2001, p. 257 - 260.

[8] A. A. M. Saleh, "Frequency independent and frequency dependent nonlinear model of TWT amplifiers," IEEE Trans. Commun., vol. COM-29, pp. 1715- 1720, Nov. 1981.

[9] J. K. Cavers, "Amplifier linearisation using a digital predistorter with fast adaptation and low memory requirements," IEEE Trans. Veh. Tech., vol. 39, no. 4, pp. 374-382, Nov. 1990.

[10] Y. Nagata, "Linear amplification techniques for digital mobile communications," Proc. IEEE Veh. Tech. Conf. (VTC '89), San Francisco, pp. 159-164, May 1-3, 1989.

[11] Y. Y. Woo, J. Kim, J. Yi, S. Hong, I. Kim, J. Moon, and B. Kim, "Adaptive digital feedback predistortion technique for linearizing power amplifiers," IEEE Trans. Microw. Theory Tech., vol. 55, no. 5, pp. 932-940, May 2007.

[12] M. K. Nezami, "Fundamentals of power amplifier linearization using digital 119 Bibliography pre-distortion," High Frequency Electronics, pp. 54-59, Sep. 2004.

[13] D. Zhou and V. DeBrunner, "Novel adaptive nonlinear predistorter based on the direct learning algorithm," IEEE Trans. Signal Process., vol. 55, no. 1, pp. 120-133, Jan. 2007.

[14] R. Marsalek, P. Jardin, and G. Baudoin, "From post-distortion to predistortion for power amplifiers linearization," IEEE Commun. Lett., vol. 7, no. 7, pp. 308-310, Jul. 2003.

[15] K. J. Cho, W. J. Kim, J. H. Kim, and S. P. Stapleton, "Linearity Optimization of a High Power Doherty Amplifier Based on Post-Distortion Compensation," IEEE Commun. Lett., vol. 15, no. 11, pp. 748-750, Nov. 2005.

[16] C. Eun and E. J. Powers, "A new Volterra predistorter based on the indirect learning architecture," IEEE Trans. Signal Process., vol. 45, no. 1, pp. 223-227, Jan. 1997.

[17] L. Ding, G. T. Zhou, D. R. Morgan, Z. Ma, J. S. Kenney, J. Kim, and C. R. Giardina, “Memory polynomial predistorter based on the indirect learning architecture,” in IEEE GLOBECOM, Nov. 2002, vol. 1, pp. 967-971.

[18] P. J. Lunsford, II, G. W. Rhyne, and M. B. Steer, “Frequency domain bivariate generalized power series analysis of nonlinear analog circuits,” IEEE Trans. Microw. Theory Tech., vol. 38, no. 6, pp. 815-818, 1990.

[19] A. Ghorbani, and M. Sheikhan, “The effect of solid state power amplifiers (SSPAs) nonlinearities on MPSK and M-QAM signal transmission,” Proceedings Sixth International Conference on Digital Processing of Signals in Communications, Loughborough, UK, September 1991, pp. 193-197, 1991.

[20] C. Rapp, “Effects of HPA-nonlinearity on a 4-DPSK/OFDM signal for a digital sound broadcasting system,” Proceedings Second European Conference on Satellite Communications, Liege, Belgium, pp. 179-184, Oct. 1991.

[21] M. Schetzen, The Volterra and Wiener Theories of Nonlinear Systems. Malabar, FL: Reprint Krieger, 2006.

[22] S. Benedetto, E. Biglieri, and R. Daffara, “Modeling and performance evaluation of nonlinear satellite links - a Volterra series approach,” IEEE Trans. Aero. Electronic Syst., vol. 15, no. 4, pp. 494-506, April 1979.

[23] J. Kim and K. Konstantinou, “Digital predistortion of wideband signals based on power amplifier models with memory,” IET Electron. Lett., vol. 37, no. 23, pp. 1417-1418, Nov. 2001.

[24] L. Ding, G. T. Zhou, D. R. Morgan, Z. Ma, J. S. Kenney, J. Kim, and C. R. Giardina, “A robust digital baseband predistorter constructed using memory polynomials,” IEEE Trans. Commun., vol. 52, no. 1, pp. 159-165, Jan. 2004.

[25] R. N. Braithwaite, “Wide bandwidth adaptive digital predistortion of power amplifiers using reduced order memory correction,” in IEEE MTT-S Int. Microwave Symp. Dig., June 2008, pp. 1517-1520.

[26] R. Raich, H. Qian, and G. T. Zhou, “Orthogonal polynomials for power amplifier modeling and predistorter design,” IEEE Trans. Veh. Technol., vol. 53, pp. 1468-1479, Sept. 2004.

[27] O. Hammi, A. M. Kedir, and F. M. Ghannouchi, “Non Uniform memory polynomial behavioral model for wireless transmitters and power amplifiers,” Proceedings of the 2012 IEEE Asia Pacific Microwave Conference (APMC), Kaohsiung, Taiwan, pp. 836-838, Dec. 2012.

[28] N. Messaoudi, M. C. Fares, S. Boumaiza, and J. Wood, “Complexity reduced odd-order memory polynomial predistorter for 400-watt multi-carrier Doherty amplifier linearization,” Digest 2008 IEEE MTT-S International Microwave Symposium (IMS), Atlanta, GA, pp. 419-422, June 2008.

[29] O. Hammi, F. M. Ghannouchi, and B. Vassilakis, “A compact envelope-memory polynomial for RF transmitters modeling with application to baseband and RF-digital predistortion,” IEEE Microw. Wireless Compon. Lett., vol. 18, n. 5, pp. 359-361, 2008.

[30] D. R. Morgan, Z. Ma, J. Kim, M. G. Zierdt, and J. Pastalan, “A generalized memory polynomial model for digital predistortion of RF power amplifiers,” IEEE Trans. Signal Processing, vol. 54, no. 10, pp. 3852-3860, 2006.

[31] J.H.K. Vuolevi and T. Rahkonen, Distortion in RF Power Amplifiers, Artech House, 2002.

[32] Pedro Miguel Brinco De Sousa. Digital predistortion of wideband satellite communication signals with reduced observational bandwidth and reduced model order complexity. Master's thesis, Universitat Politècnica de Catalunya, 2014. 6, 14, 16,

[33]https://es.mathworks.com/matlabcentral/answers/289433-memory-polynomial-model-for-power-amplifier

[34]http://azadproject.ir/wp-content/uploads/2016/06/2006-A-Generalized-Memory-Polynomial-Model-for-Digital-Predistortion-of-RF-Power-Amplifiers.pdf

[35]https://www.radioeng.cz/fulltexts/2018/18_03_0909_0916.pdf

[36]https://es.mathworks.com/help/comm/ref/comm.dpdcoefficientestimator-system-object.html

[37]https://es.mathworks.com/help/simrf/examples/power-amplifier-characterization-with-dpd-for-reduced-signal-distortion.html

[38]https://upcommons.upc.edu/bitstream/handle/2117/80430/TMTT-2014-09-1115_Main.pdf

[39]https://pastel.archives-ouvertes.fr/tel-01997230/document

[40]https://www.researchgate.net/profile/Oualid_Hammi/publication/283328097_A_Novel_Weighted_Memory_Polynomial_for_Behavioral_Modeling_and_Digital_Predistortion_of_Nonlinear_Wireless_Transmitters/links/5638f5c708aed5314d22179e/A-Novel-Weighted-Memory-Polynomial-for-Behavioral-Modeling-and-Digital-Predistortion-of-Nonlinear-Wireless-Transmitters.pdf?origin=publication_detail

[41]https://core.ac.uk/download/pdf/76530542.pdf

[42]https://www.kth.se/polopoly_fs/1.591453.1550157530!/readme.pdf

[43]https://www.sciencedirect.com/science/article/pii/S0167926017300032

[44]https://www.chalmers.se/en/centres/ghz/publications/Documents/Lic_thesis_18-%20Digital_predistrotion_for_the_linearization_of_power_amplifiers.pdf

[45]https://www.eit.lth.se/sprapport.php?uid=954

[46]https://www.litepoint.com/wp-content/uploads/2018/12/Adaptive-DPD-Tech-Specs-050917.pdf

[47]https://upcommons.upc.edu/handle/2117/106582

[48]https://www.javatpoint.com/numpy-concatenate