Vol 56, No 2—April 2022

Digital Predistortion for RF Communications: From Equations to Implementation

Claire Masterson, Systems Engineer

Abstract

This article covers the mathematical fundamentals of digital predistortion (DPD) and how it is implemented in a transceiver's microprocessor and hardware. It addresses why DPD is needed in modern communication systems and explores how the mathematical model captures real-world signal distortion.

Introduction

DPD is an acronym that will be familiar to many RF (radio frequency) engineers, signal processing enthusiasts, and embedded software developers. DPD is ubiquitous in our cellular communications systems, enabling power amplifiers (PAs) to efficiently deliver maximum power to an antenna. As 5G drives up the antenna count in base stations and our spectrum becomes ever more congested, DPD has emerged as a key technology to allow for the development of efficient, cost-effective, and specification compliant cellular systems.

Many of us have a unique understanding of DPD based on our own perspectives, be that from the purely mathematical viewpoint or the more constrained implementation on a microprocessor. Perhaps you're an engineer evaluating the performance of DPD in your RF base station product or an algorithm developer who is curious as to how mathematical modeling techniques are implemented in real world-systems. This article aims to broaden your knowledge and empower you to fully grasp the topic from all angles.

What Is DPD and Why Is It Used?

When an RF signal is outputted from a base station radio (see Figure 1), it needs to be amplified before it is transmitted through the antenna. An RF PA is used to do this. In an ideal world, the PA takes an input signal and outputs a higher power signal that is proportional to its input. It also does this in the most power efficient way possible so that most of the DC power supply provided to the amplifier is converted into signal output power.

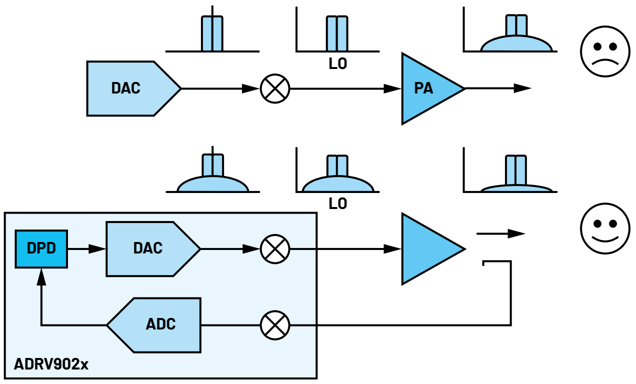


Figure 1. Block diagram of a simplified radio structure with and without DPD.

However, this isn't an ideal world. PAs are made from transistors, which are active devices and inherently nonlinear. Now if we use PAs in their "linear" region (linear here is a relative term; hence, the quotation marks) as shown in Figure 2, then the output power is relatively proportional to the input power. The downside of this approach is that the PA is generally used in a very inefficient state with most of the power provided being lost as heat. We often want to use PAs at the point where they are beginning to compress. That means if the input signal is increased by a set amount (say 3 dB), the PA output does not increase by the same amount (maybe only by 1 dB). Obviously, the signal is being significantly distorted at this point by the amplifier.

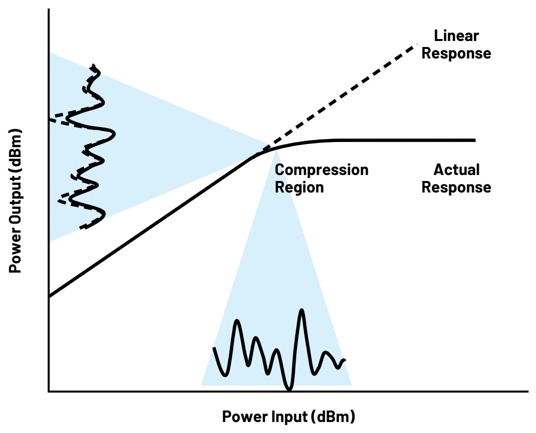


Figure 2. PA input power vs. output power plot (shows projection of sample input/output signals).

This distortion happens at known places in the frequency domain dependent on the input signal. Figure 3 shows these locations and the relationship between the fundamental frequencies and these distortion products. In RF systems, the only distortions we need to compensate for are those close to the fundamental signal, which are the odd order intermodulation products. Filtering in the system takes care of the out of band products (harmonics and even order intermodulation products). Figure 4 shows the output of an RF PA run near its compression point. The intermodulation products (especially the third order) are clearly visible. They look like "skirts" around the desired signal.

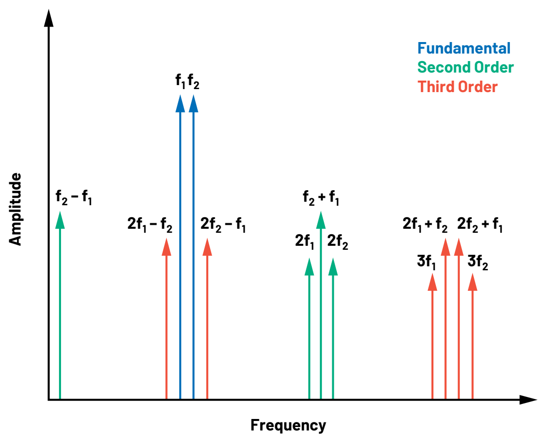
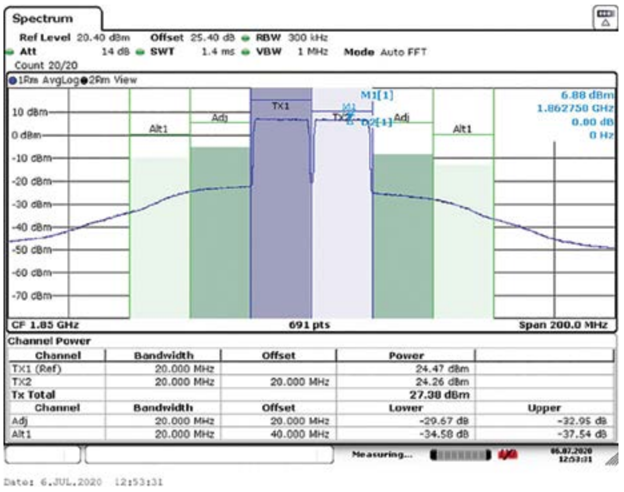


Figure 3. Location of intermodulation and harmonic distortion for 2 tone input.


Spectrum

Ref Level 20.40 dBm Offset 25.40 dB RBW 300 kHz
Att 14 dB SWT 1.4 ms VBW 1 MHz Mode Auto FFT
Count 20/20
1Rm AvgLog 2Rm View

M1[1] 6.88 dBm 1.862750 GHz
D2[1] 0.00 dB 0 Hz

CF 1.85 GHz 691 pts Span 200.0 MHz

Channel Power

Channel	Bandwidth	Offset	Power	
TX1 (Ref)	20.000 MHz		24.47 dBm	
TX2	20.000 MHz	20.000 MHz	24.26 dBm	
Tx Total			27.38 dBm	
Channel	Bandwidth	Offset	Lower	Upper
Adj	20.000 MHz	20.000 MHz	-29.67 dB	-32.95 dB
Alt1	20.000 MHz	40.000 MHz	-34.58 dB	-37.54 dB

Measuring... 06.07.2020 12:53:31

Date: 6.JUL.2020 12:53:31

Figure 4. 2× 20 MHz carriers passed through SKY66391-12 RF PA. The center frequency = 1850 MHz.

DPD aims to characterize this distortion by observing the PA output and, knowing the wanted output signal, altering the input signal so that the PA output is closer to the ideal. This can only be done effectively in fairly specific circumstances. We need to have the amplifier and input signal configured such that the amplifier is compressing somewhat but not completely saturating.

The Math Behind Modeling PA Distortion

Do the sight of Greek letters and other mathematical symbols tend to bring on terrifying flashbacks of college exams of yesteryear? You're not alone! People can be unnecessarily put off by the fundamentals when one of the first references they are given is a math heavy academic paper. The paper "A Generalized Memory Polynomial Model for Digital Predistortion of RF Power Amplifiers"[1] is a seminal work that introduces the widely adopted generalized memory polynomial (GMP) approach to DPD. If you just dabble in the signal processing side of things, it may be a little heavy for an introduction to the subject. So, as a start, let's try and break down the GMP approach and reach a more intuitive understanding of what the math is doing.

The Volterra series is the mathematical backbone of DPD. It is used to model nonlinear systems with memory. *Memory* simply means that the present output of the system can depend on the current and past inputs. Volterra series is very general (and, hence, powerful) and is used in many fields outside of electrical engineering. For PA DPD, the Volterra series can be slimmed down and made such that it is more implementable and stable in real-time digital systems. GMP is one such slimmed down approach.

Figure 5 describes how GMP is used to model the relationship between the input, x, of the PA and its output, y. You'll see the three separate summation blocks of the equation are very similar to each other. Let's focus on the first one for now highlighted in red below. The $|x(\ldots)|^k$ term is referred to as the envelope of the input signal, where k is the polynomial order. l incorporates memory into the system. If $L_a = \{0,1,2\}$, then the model allows the output $y_{GMP}(n)$ to depend on the current input $x(n)$ and past inputs $x(n-1)$ and $x(n-2)$. Figure 6 examines the effect of the polynomial order k on a sample vector. The vector, x, is a single 20 MHz carrier and is plotted at complex baseband. The GMP modeling equation is simplified by removing the memory component. The plots of $x|x|^k$ show a clear similarity to the real-world distortion visible in Figure 4.

Each polynomial order (k) and memory lag (l) has an associated complex weight (a_{kl}). When the complexity of the model has been chosen (which values of k and l will be included), it is then necessary to solve for these weights based on real-world observations of the PAs output for a known input signal. Figure 7 converts the simplified equation into matrix form. The mathematical notation used allows for a concise representation of the model. However, for the actual implementation of DPD on buffers of digital data, it is easiest and more representative to view things in matrix notation.

Let's briefly look at the second and third lines of the equation in Figure 6 that were ignored for simplicity. Note that if m is set to zero, then these lines become identical to the first one. These lines allow for delays (both positive and negative) to be added between the envelope term and the complex baseband signal. These are called lagging and leading crossterms and can improve the modeling accuracy of DPD significantly. They offer an extra level of freedom in our attempts to model the amplifier's behavior. Note that M_b, M_c, K_b, and K_c do not contain zero; otherwise, we would be repeating terms from the first line.

$$\begin{aligned} y_{GMP}(n) = &\sum_{l\epsilon L_a}\sum_{k\epsilon K_a} a_{kl}x(n-l)|x(n-l)|^k \\ &+ \sum_{k\epsilon K_b}\sum_{l\epsilon L_b}\sum_{m\epsilon M_b} b_{klm}x(n-l)|x(n-l-m)|^k \\ &+ \sum_{k\epsilon K_c}\sum_{l\epsilon L_c}\sum_{m\epsilon M_c} c_{klm}x(n-l)|x(n-l+m)|^k \end{aligned}$$

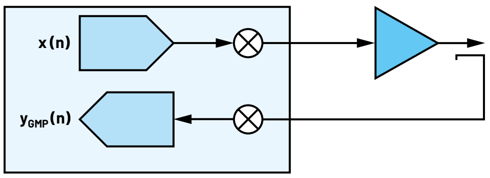


Figure 5. GMP for modeling PA distortion.[1]

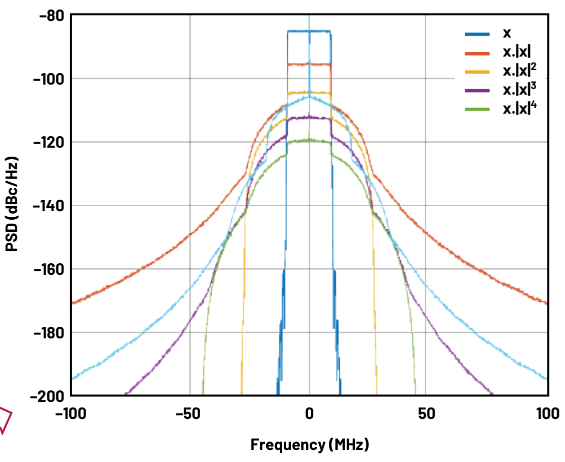
$$\begin{aligned} y_{GMP}(n) = &\boxed{\sum_{l\epsilon L_a}\sum_{k\epsilon K_a} a_{kl}x(n-l)|x(n-l)|^k} \\ &+ \sum_{k\epsilon K_b}\sum_{l\epsilon L_b}\sum_{m\epsilon M_b} b_{klm}x(n-l)|x(n-l-m)|^k \\ &+ \sum_{k\epsilon K_c}\sum_{l\epsilon L_c}\sum_{m\epsilon M_c} c_{klm}x(n-l)|x(n-l+m)|^k \end{aligned}$$

$$\Downarrow$$

Simplify

- $K_a = \{0, 1, 2, 3, 4\}$
- $L_a = 0$

$$y_{GMP}(n) = \sum_{k\epsilon K_a} a_{kl}\boxed{x(n)|x(n)|^k} \Rightarrow$$



Plot Highlighted Element for Each Value of k
y_{GMP}(n) Is the Sum of These Elements, Each Multiplied by a Complex Weight, a_{kl}

Figure 6. Plot of effect of order (k) on signal in frequency domain of a signal x.

Convert From Equations to Matrix Operations on Vectors/Buffers of Data

$$y_{GMP}(n) = \sum_{k\epsilon K_a} a_{kl}x(n)|x(n)|^k \Rightarrow \begin{bmatrix} y_{GMP} \end{bmatrix} = \begin{bmatrix} x & x|x|^1 & x|x|^2 & x|x|^3 & x|x|^4 \end{bmatrix} \begin{bmatrix} a_{00} \\ a_{10} \\ a_{20} \\ a_{30} \\ a_{40} \end{bmatrix} \Rightarrow \hat{y} = Xw$$

- $K_a = \{0, 1, 2, 3, 4\}$
- $L_a = 0$

Figure 7. Convert the simplified equation to matrix operation on data buffers (closer to how it is implemented digitally).

So how do we settle on the order of the model, the number of memory terms, and what crossterms we should add? This is where a certain amount of "black magic" comes into things. It is possible to be guided to some degree by our knowledge of the physics of the distortion. The type of amplifier and the material it is manufactured out of and the bandwidth of the signal being played through it all impact on the modeling terms and allow an engineer experienced in the area to put bounds on what model should be used. However, there is also a degree of trial and error involved on top of this.

The last aspect of the problem to be addressed from a mathematical viewpoint, now that a modeling structure is available, is how to solve for the weighting coefficients. In a practical scenario, there is a tendency to solve for the inverse of the model described above. It turns out that there is a nice reciprocity with these model coefficients, in that the same weights can be used to postdistort the captured PA output vector to remove nonlinearities and to predistort the transmitted signal sent through the PA so that the PA output appears as linear as possible. Figure 8 shows a block diagram of how the weight coefficient estimation and predistortion are carried out.

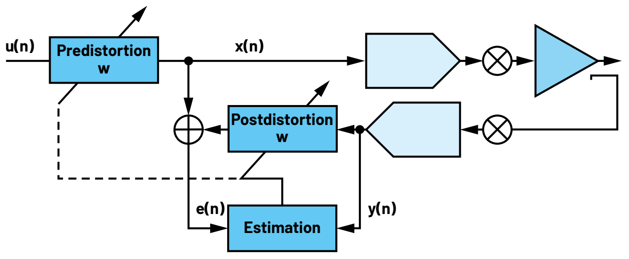


Figure 8. Block diagram describing indirect implementation of modeling and predistortion.

For the inverse model, the matrix equation given in Figure 7 is swapped around to give x̂ = Yw. Here the matrix Y is formed in the same way as X was in the other case, as shown in Figure 9. For this example, a memory term has been included and the number of polynomial orders included has been reduced. To solve for w, we need to get the inverse of Y. Y is not square (it's a tall, slim matrix) so this is achieved using the matrix "pseudo inverse" (see Equation 1). This solves for w in a least squares sense, that is, it minimizes the square of the difference between x̂ and Yw, which is what we want!

$$w = (Y^H Y)^{-1} Y^H x \tag{1}$$

This can be refined a bit further to take into account that it's being applied in a live environment with varying signals. Here the coefficients are constrained by being updated from their previous value. μ is a constant value between 0 and 1 that controls how much the weights can change per iteration. If μ = 1 and $w_0 = 0$, then this equation reverts to the basic least squares solution immediately. If μ is set to a value less than 1, then it will take a number of iterations for the coefficients to converge.

$$w_{i+1} = w_i + \mu (Y^H Y)^{-1} Y^H e,\ e = x - \hat{x} \tag{2}$$

Note that the modeling and estimation techniques described here are not the only way to do DPD. Techniques such as dynamic deviation reduction (DDR)-based modeling can also be used instead of or in addition to it. The estimation techniques described for solving for the coefficients can also be done in numerous ways. Given this is a short article and not a book, let's leave it there.

How Do We Implement This in a Microprocessor?

Okay, so the math has been thoroughly covered. The next question is how is it applied in real-world communication systems? It is implemented in the digital baseband, generally in a microprocessor or an FPGA. ADI's RadioVerse® transceiver products such as the ADRV902x family have built-in microprocessor cores, with a structure specifically put in place to allow for easy DPD implementation.

There are two distinct aspects to DPD implementation in embedded software. The first is the DPD actuator, which is where the predistortion of the live transmitted data is performed in real time, and the second is the DPD adaption engine, which is where the DPD coefficients are updated based on observations of the PA output.

The key to how DPD and many other signal processing concepts are implemented in real time in a microprocessor or similar is through the use of lookup tables (LUTs). LUTs allow for expensive run-time calculations to be replaced with a simpler array indexing operation. Let's consider how the DPD actuator applies predistortion to a transmitted sample of data. The notation is as shown in Figure 8, where u(n) is the raw sample of data to be transmitted and x(n) is the predistorted version. Figure 10 shows the calculations required to obtain one predistorted sample for a given scenario. This is a relatively limited example with the highest polynomial order being third order and only one memory tap and a single crossterm. Even for this case, there are clearly a lot of multiplication, power of, and addition calculations needed to gain this one sample of data.

This is where LUTs come into play to ease the real-time computation burden. Figure 10 can be rewritten as Figure 11 where the data that will be entered in the LUTs become more evident. Each LUT contains the result of the highlighted element of the equation for a large number of possible values for |u(n)|. The resolution depends on the size of the LUT that can be implemented in the available hardware. The magnitude of the current input sample is quantized depending on the resolution of the LUT and used as an index to access the correct LUT element for that given input.

$$
\begin{aligned}
x(n) = u(n)&\boxed{[a_{00} + a_{10}|u(n)|^1 + a_{20}|u(n)|^2]} + && \boxed{\text{LUT1}}\\
u(n-1)&\boxed{[a_{01} + a_{11}|u(n-1)|^1 + a_{21}|u(n-1)|^2]} + && \boxed{\text{LUT2}}\\
u(n)&\boxed{[b_{201}|u(n-1)|^2]} && \boxed{\text{LUT3}}
\end{aligned}
$$

Figure 11. Regrouping equation elements to show LUT's structure.

$$\hat{x} = Yw$$

- $K_a = \{0, 1, 2\}$
- $L_a = \{0, 1\}$
- $y_{delay} = [0, y\,(1{:}\ end - 1)]$

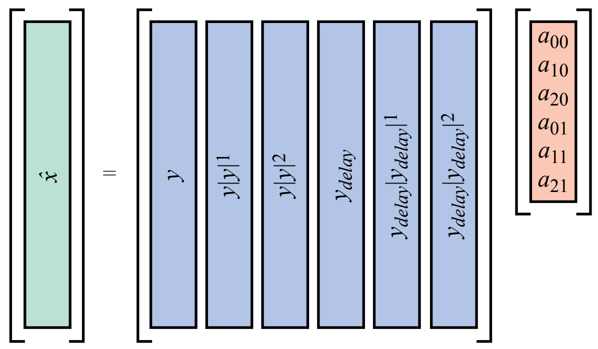


Figure 9. Inverse approach equation in matrix form. Some memory has been included here.

- $K_a = \{0, 1, 2\}$
- $L_a = \{0, 1\}$
- $K_b = \{2\}$
- $L_b = \{0\}$
- $M_a = \{1\}$

$$
\begin{aligned}
x(n) = {} & a_{00}u(n) + a_{10}u(n)|u(n)|^1 + a_{20}u(n)|u(n)|^2 + a_{01}u(n-1)\\
& + a_{11}u(n-1)|u(n-1)|^1 + a_{21}u(n-1)|u(n-1)|^2 + b_{201}u(n)|u(n-1)|^2
\end{aligned}
$$

Figure 10. Predistortion calculation for third-order case with one memory tap and one third-order crossterm element.

Figure 12 shows how the LUTs are incorporated into the full predisortion actuator implementation for our example case. Note that this is just one possible implementation out of many. One example of a change that could be made while still maintaining the same output is that the delay element, z^{-1}, could be moved to the right of LUT2.

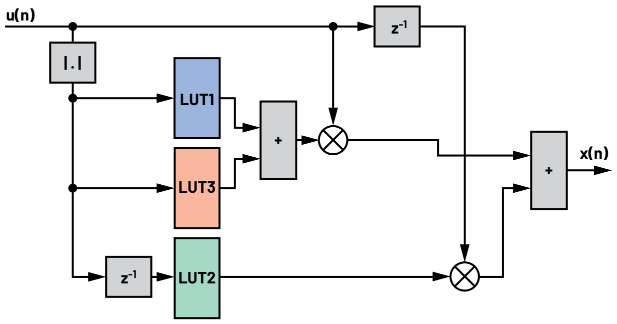


Figure 12. Block diagram of possible implementation of DPD using LUTs.

The adaption engine is tasked with solving for the coefficients that are used to calculate the LUT values in the actuator. This involves solving for the w vector described in equations 1 and 2. The pseudoinverse matrix operation, $(Y^H Y)^{-1} Y^H$, is computationally hungry. Equation 1 can be rewritten as

$$Y^H Y w = Y^H x \quad (3)$$

If $C_{YY} = Y^H Y$ and $C_{Yx} = Y^H x$, then Equation 3 becomes

$$C_{YY} w = C_{Y x} \quad (4)$$

C_{YY} is a square matrix and can be decomposed into the product of an upper triangular matrix, L, and its conjugate transpose ($C_{YY} = L^H L$) using Cholesky decomposition. This allows us to solve for w by introducing a dummy variable z and solving for it as shown:

$$L^H z = C_{Yx} \quad (5)$$

Then substitute this dummy variable back in to solve for

$$L w = C_{Yx} \quad (6)$$

Because L and L^H are upper and lower triangular matrices, respectively, Equation 5 and Equation 6 are easy to solve with minimal computational expense to give w. Every time the adaption engine is run and new values for w are found, it is necessary to update the actuator LUTs to reflect them. The adaption engine may be performed at set regular intervals or at more irregular intervals based on the observation of the PA output or the operator's knowledge of changes to the signal to be transmitted.

The implementation of DPD in an embedded system requires a lot of checks and balances to ensure the stability of the system. It is of utmost importance that the transmitted data buffer and capture buffer data are time aligned to ensure the mathematical relationship established between them is correct and holds true as it is applied over time. If this alignment is lost, then the coefficients returned by the adaption engine will not predistort the system correctly and may lead to instability in the system. The predistorted actuator output should also be checked to ensure that the signal will not saturate the DAC.

Conclusion

Hopefully, this article has cleared up some of the mystery around DPD by examining the underlying mathematics and its implementation in hardware. This is just the tip of the iceberg on this fascinating topic and may prompt the reader to look further into the application of signal processing techniques in communications systems. The study of Pratt and Kearney is a good source about DPD applied to an ultrawide bandwidth use case in a wired communications system.[2] ADI's RadioVerse transceiver products are uniquely placed to incorporate algorithms such as DPD as they provide highly integrated RF hardware and configurable software tools to customers.

References

[1] Dennis R. Morgan, Zhengxiang Ma, Jaehyeong Kim, Michael G. Zierdt, and John Pastalan. "A Generalized Memory Polynomial Model for Digital Predistortion of RF Power Amplifiers." *IEEE Transactions on Signal Processing*, Vol. 54, No. 10, October 2006.

[2] Patrick Pratt and Frank Kearney. "Ultrawideband Digital Predistortion (DPD): The Rewards (Power and Performance) and Challenges of Implementation in Cable Distribution Systems." *Analog Dialogue*, Vol. 51, No. 3, July 2017.

About the Author

Claire Masterson is a systems engineer in the Wireless Systems Group at Analog Devices Limerick working on systems implementation, software development, and algorithm development and verification. She received a B.A.I. and Ph.D. from Trinity College Dublin and joined ADI in 2011 after graduating. She is particularly interested in the application of digital signal processing in real-world systems, particularly in 5G and 6G system development and next-generation DPD implementations. She can be reached at claire.masterson@analog.com.