Adaptive DPD Design

By Kerry Schutz and Dick Benson, MathWorks

Digital pre-distortion (DPD) is a baseband signal processing technique that corrects for impairments inherent to RF power amplifiers (PAs). These impairments cause out-of-band emissions or spectral regrowth and in-band distortion which correlates with an increased bit-error-rate (BER). Wideband signals with a high peak-to-average ratio, as is characteristic of LTE/4G transmitters, are particulary susceptible to these unwanted effects.

In this paper, we illustrate a workflow for modeling and simulating PAs and DPDs. The models shown in this article are based on two technical papers, [1] and [2]. We start from PA measurements. From measurements, we derive a static DPD design based on a memory polynomial. Such a polynomial corrects for both the non-linearities and memory effects in the PA. For simulation purposes, we construct a system-level model to evaluate the DPD's effectiveness. Because any PAs characteristics will vary over time and operating conditions, we extend the static DPD design to an adaptive one. We evaluate two adaptive DPD designs, one based on the (Least-Mean Square) LMS algorithm and a second using the Recurive Predictor Error Method (RPEM) algorithm.

The examples used in this article are available for download.

The Workflow

The goal of this work is a simulation model which accurately models the PA's impairments and an adaptive DPD design which mitigates those impairments. We divide the modeling effort into four parts.

1. Model and simulate the PA.
2. Derive DPD coefficients.
3. Evaluate static DPD design.
4. Extend static DPD design to adaptive one.
    a. Evaluate least mean square (LMS)-variant
    b. Evaluate recursive predictor error method (RPEM)-variant

State Your Assumptions

Models are never perfect representations of reality. The key is to simplify reality while maintaining sufficient relevance. We made the following simplifications in order to accelerate the simulations.

1. We model the PA as a discrete time system. In practice, the PA is an analog circuit.
2. We model the PA signal as baseband complex. In reality, the PA signal is a real passband signal (Figure 8).
3. We use double-precision data types and math. In reality, integer data types and math would be used.
4. We do not include quantization effects due to the ADC and DAC (Figure 8).

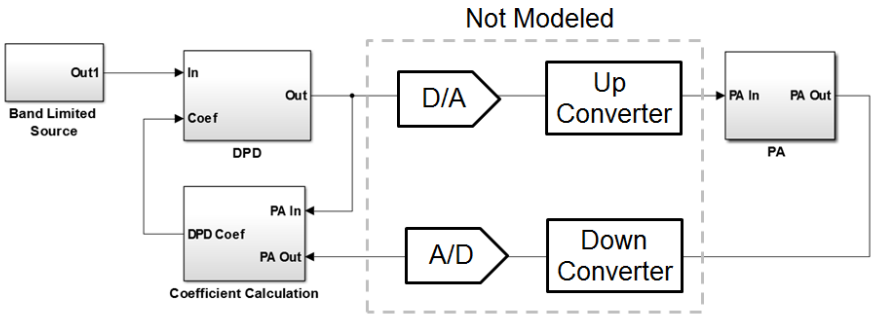


Figure 1. The simplified system-level model does not model the effects of quantization and up/down conversion. These elements are replaced with pass throughs.

Phase 1. Modeling and Simulating the PA

The PA model consists of a Saleh amplifier in series with an asymmetrical complex filter as shown in Figures 2 and 3. The excitation is an additive white Gaussian noise (AWGN) signal that has been low-pass elliptic filtered. We run the model and log the input and ouput signals, x and y, respectively while monitoring their spectrums shown in Figure 4.

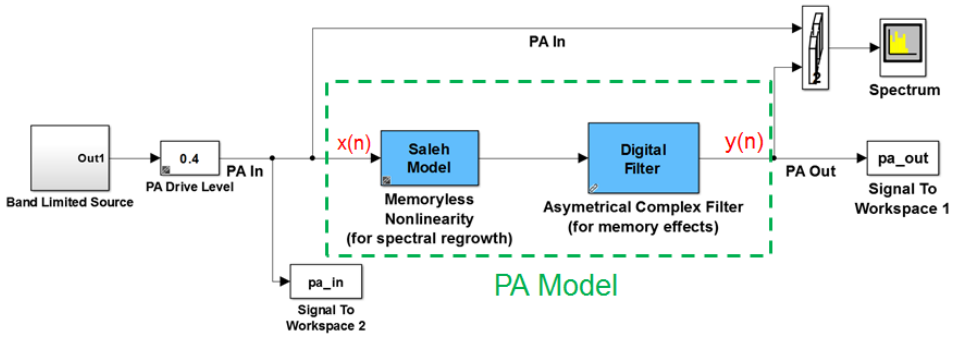


Figure 2. The PA model.

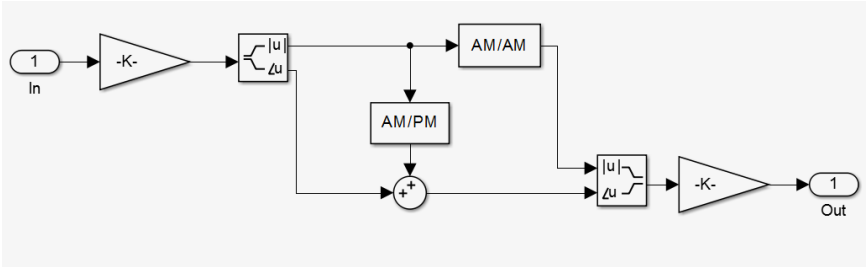


Figure 3. The structure of the Saleh memoryless non-linearity is described in detail in [3].

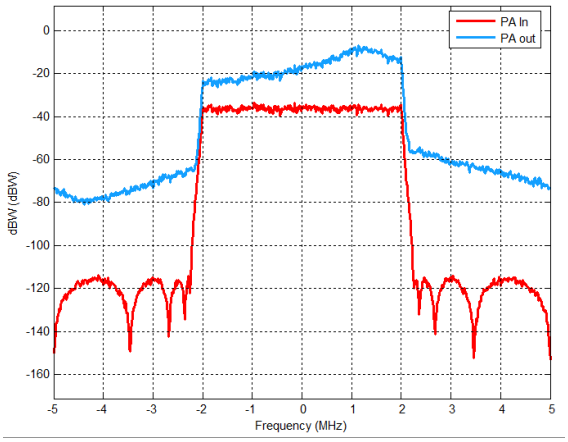


Figure 4. The PA output shows 20 dB of gain at the expense of significant spectral regrowth and in-band distortion.

Phase 2. Deriving the DPD Coefficients

The static DPD design is derived from PA measurements. We illustrate the derivation graphically using Figure 3. The top path in Figure 3 represents our PA model in Figure 2. The PA is partitioned into a non-linear function followed by a linear gain G. The middle path shows the PA running in reverse. This path represents the DPD. Although you cannot physically run a PA in reverse, mathematically you can, and this is the key to the DPD derivation. In reverse, we apply the inverse non-linear operation, $f_{-1}(x_1,x_2,\ldots x_n)$. The bottom path is the cascade of the top two paths, namely, the DPD and the PA.

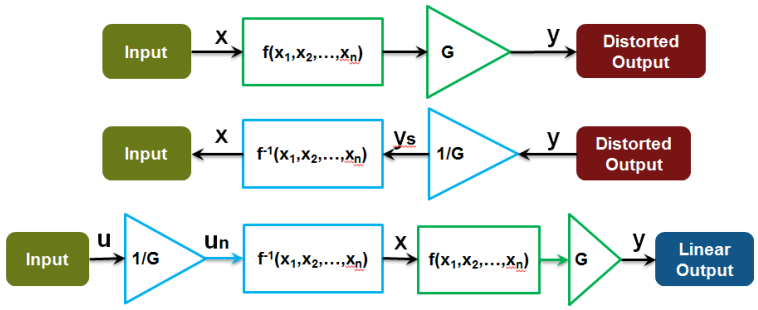


Figure 5. These block diagrams graphically illustrate the DPD derivation and implementation process.

The DPD derivation can now proceeds as follows.

1. Assume a memory polynomial [1] form for the PA's non-linear operator, $f(x_1,x_2,\ldots x_n)$.

$$y_{MP}(n) = \sum_{k=0}^{K-1}\sum_{m=0}^{M-1} a_{km}\, x(n-m)\left|x(n-m)\right|^k \qquad \text{Equation 1}$$

where

x is the PA input

y is the PA output

a_{km} are the PA polynomial coefficients

M is the PA memory depth

K is the degree of PA non-linearity

n is the time index

The input x, the output y, and the coefficients a_{km} are complex valued.

2. Reverse the roles of x and y in 1 to model the DPD's inverse non-linear function, $f_{-1}(x_1,x_2,\ldots x_n)$.

$$x_{MP}(n) = \sum_{k=0}^{K-1}\sum_{m=0}^{M-1} d_{km}\, y_{ss}(n-m)\left|y_{ss}(n-m)\right|^{k} \qquad \text{Equation 2}$$

Note that y(n) has been normalized by the linear gain G and optionally time shifted.

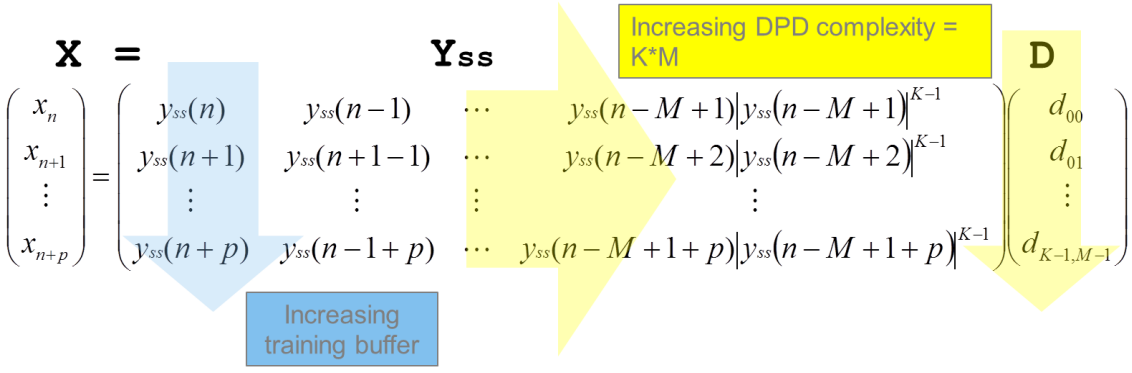
$y_s(n) = y(n)/G$

$y_{ss}(n) = y_s(n+\text{offset})$

where offset is a fixed positive integer.

Timing alignment is critical. If the PA has significant memory it may be necessary to offset y in time before deriving the coefficients. The DPD must correctly account for the PA's positive delay. No realizeable structure can possess negative delay but you can derive the coefficients based on a time shift in the PA output. And remember the PA output is an input in the DPD coefficient derivation. Thus you can make the output x respond to the input y "offset" samples earlier by creating a new sequence $y_{ss}(n) = y_s(n+\text{offset})$ upon which to base the coefficient derivation. In our example, we use an offset of 3 samples.

3. Solve for the DPD coefficients, d_{km}. We re-write Equation 2 as a set of system of linear equations as shown in Figure 9. Solving for d_{km} then amounts to solving an over-determined system of linear equations, Equation 4.



$$\begin{pmatrix} x_n \\ x_{n+1} \\ \vdots \\ x_{n+p} \end{pmatrix} = \begin{pmatrix} y_{ss}(n) & y_{ss}(n-1) & \cdots & y_{ss}(n-M+1)\left|y_{ss}(n-M+1)\right|^{K-1} \\ y_{ss}(n+1) & y_{ss}(n+1-1) & \cdots & y_{ss}(n-M+2)\left|y_{ss}(n-M+2)\right|^{K-1} \\ \vdots & \vdots & \vdots & \vdots \\ y_{ss}(n+p) & y_{ss}(n-1+p) & \cdots & y_{ss}(n-M+1+p)\left|y_{ss}(n-M+1+p)\right|^{K-1} \end{pmatrix} \begin{pmatrix} d_{00} \\ d_{01} \\ \vdots \\ d_{K-1,M-1} \end{pmatrix}$$

Figure 6. Equation 2 rearranged as a system of linear equations in matrix form. The variable **p** denotes the number of measurement samples. The value of **p** is much typically much greater than the product K*M. Thus this is an over-determined system.

The measurements x and y are knowns. A certain value for **K** and **M** is chosen based on the complexity of the PA. We chose **M** = **K** = 5 and thus solved for K*M = 25 complex coefficients.

$d_{km} = Y_{ss} \backslash X$ Equation 4

We use MATLAB's backslash operator to solve this over-determined system of equations in a least-squares sense.

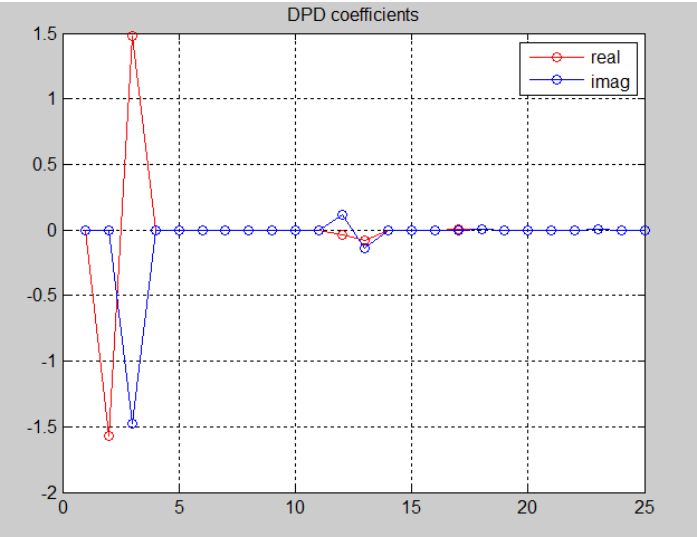


Figure 7. The real and imaginary components of the derived DPD's complex coefficients.

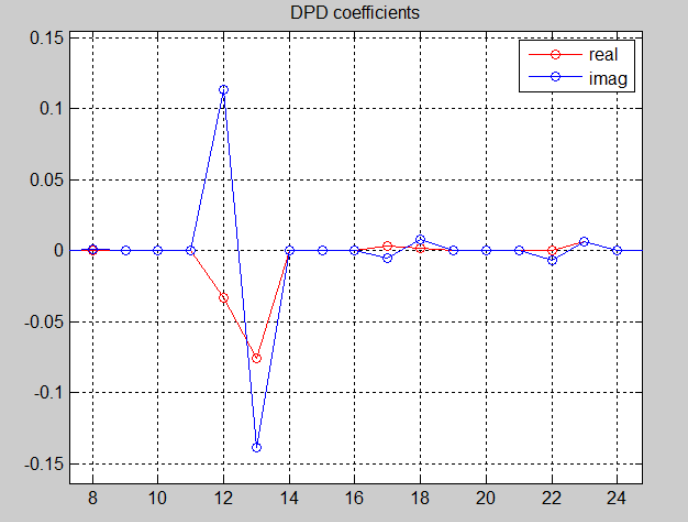


Figure 8. Here we zoom-in on coefficients 8 through 24. These coefficents are vital in reducing spectral regrowth.

Phase 3. Evaluating the Fixed-Coefficient DPD Design

To evaluate the design, we created a system model of the DPD and PA (Figure 7). The first task involves implementing Equation 2. It is straightforward to represent these equations in MATLAB as shown in Table 1.

Equation 2 in symbolic form	$$x_{MP}(n) = \sum_{k=0}^{K-1} \sum_{m=0}^{M-1} d_{km} u(n-m) \lvert u(n-m) \rvert^{k}$$

Equation 2 in MATLAB code	`function x=DPD(u,coef,K,M)`
	`persistent pipe`
	`if isempty(pipe)`
	`  pipe=complex(zeros(M,1));`
	`end`
	`pipe(1:end-1)=pipe(2:end);`
	`pipe(end) = u;`
	`y = complex(0,0);`
	`for k=1:K`
	`  for m=1:M`
	`  y=y+coef((k-1)*M+m)*pipe(m)*abs(pipe(m))^(k-1);`
	`  end`
	`end`

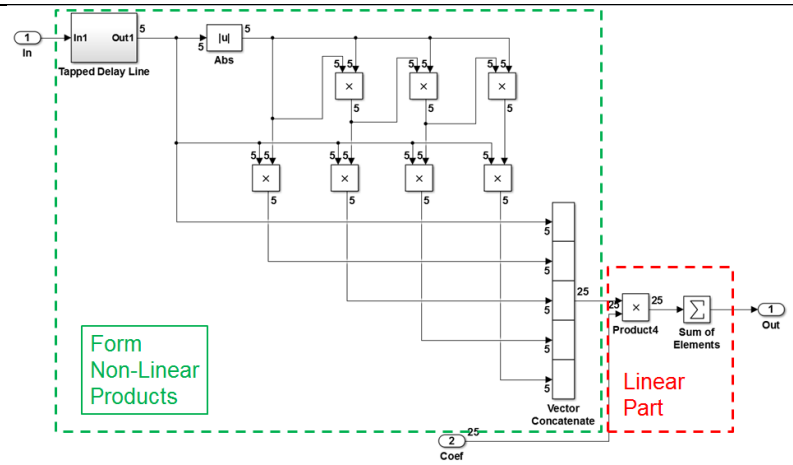
Equation 2 In Simulink block diagram form	

Table 1. Equation 2 can be implemented using either MATLAB code or blocks in Simulink.

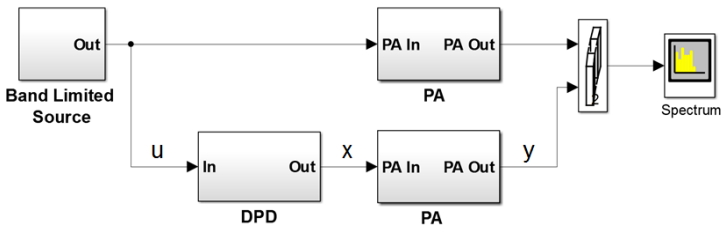


Figure 9. DPD verification model.

The results of simulating the verification model for different values of K and M are shown in Figures 8 and 9.

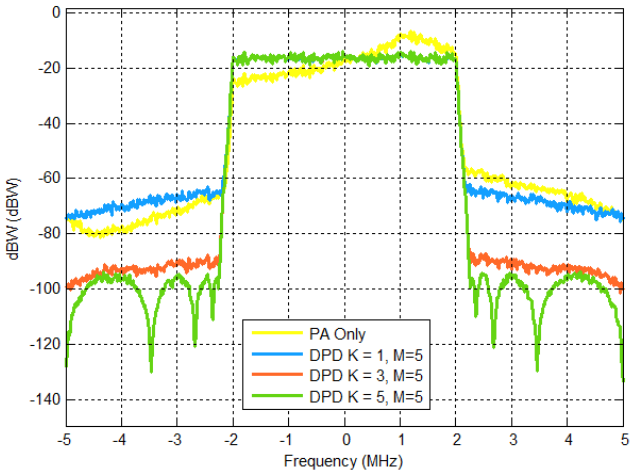


Figure 10. This plot shows the importance of modeling non-linearities. All DPD variations adequately correct for in-band distortion but it is clear that a purely linear DPD is not sufficient for reducing spectral regrowth.

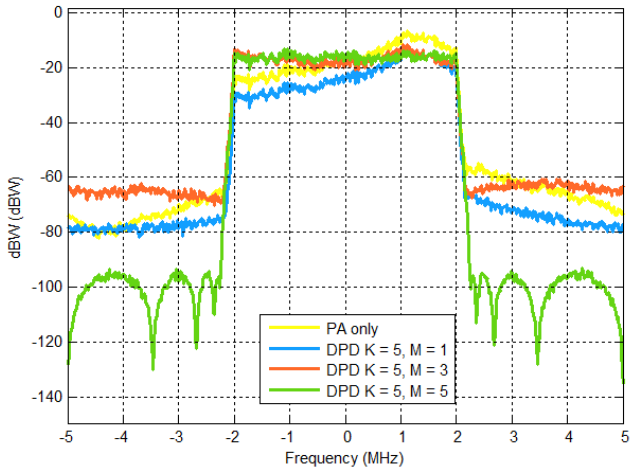


Figure 11. This plot shows the importance of modeling memory effects. Only the M = 5 DPD variant performs well.

Phase 4. Going Adaptive

Although our DPD design shows significant promise, it does not lend itself well to an adaptive implementation. Additionally, the matrix inverse math and the large buffers required to solve an over-determined system of equations is not feasible to implement in hardware.

We used an indirect learning architecture [2] to implement an adaptive DPD, Figure 10. The design consists of a Coefficient Calculation subsystem and a DPD subsystem. This is a streaming implementation with no matrix inverse.

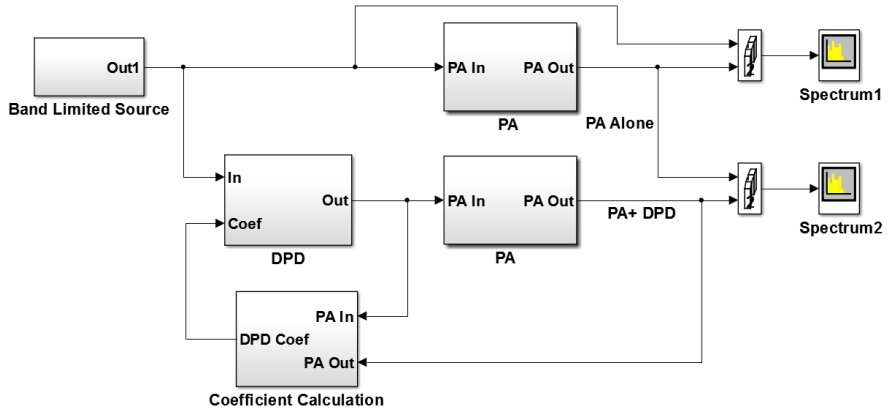


Figure 12. This is our adaptive DPD testbench. The Coefficient Calculation subsystem samples the PA input and output and performs matrix computations to derive a set of DPD coefficients. The DPD subsystem applies these coefficients to a memory polynomial and outputs a pre-distorted waveform.

The DPD subsystem is identical in form to that shown in Table 1. Here we focus on the coefficient calculation subsystem, Figure 11.

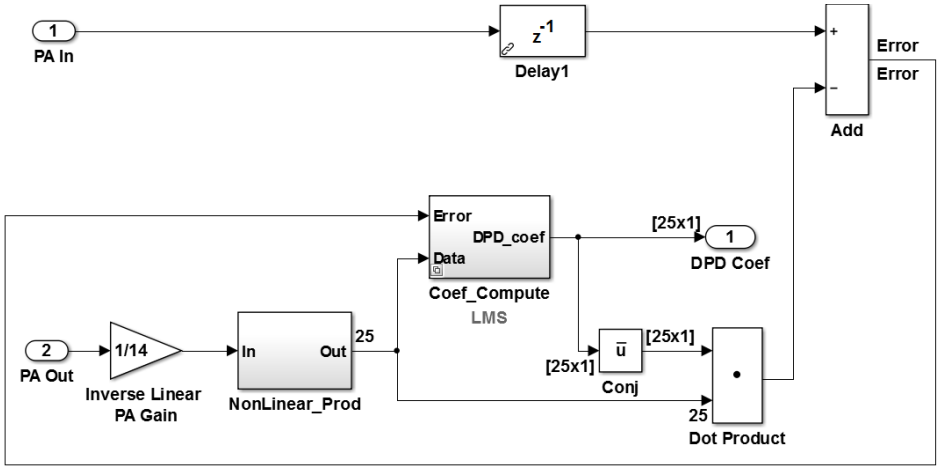


Figure 13. High-level view of the coefficient calculation subsystem.

The adaptive DPD architecture has two copies of the DPD algorithm, one for learning the coefficients and the other for implementing them. The combination of the **NonLinear_Prod** (Figure 12) and **Coef_Compute** subsystems implement the learning copy of the DPD subsystem.

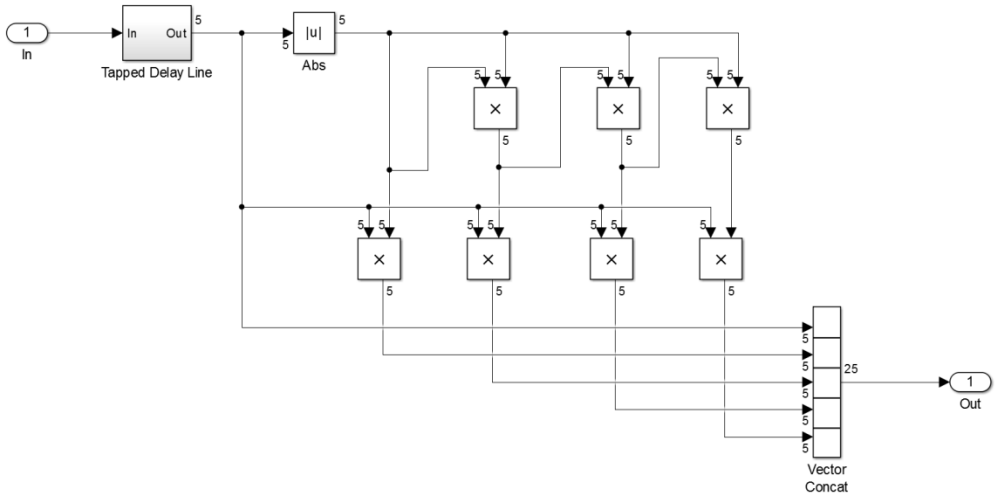


Figure 14. The **NonLinear_Prod** subsystem implements the non-linear multiplies in the DPD equation, Equation 2.

The DPD coefficients, d_{km}, are computed on-the-fly using one of two possible algorithms, the LMS algorithm or the recursive predictor error method (RPEM) algorithm as shown in Figures 13 and 14 respectively.

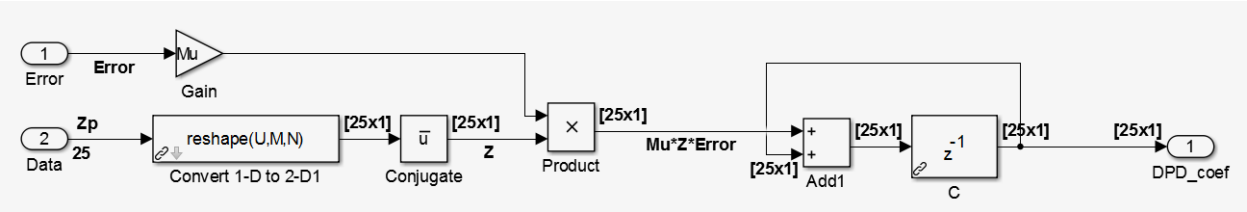


Figure 15. The LMS-based coefficient computation is relatively simple in terms of the required resources.

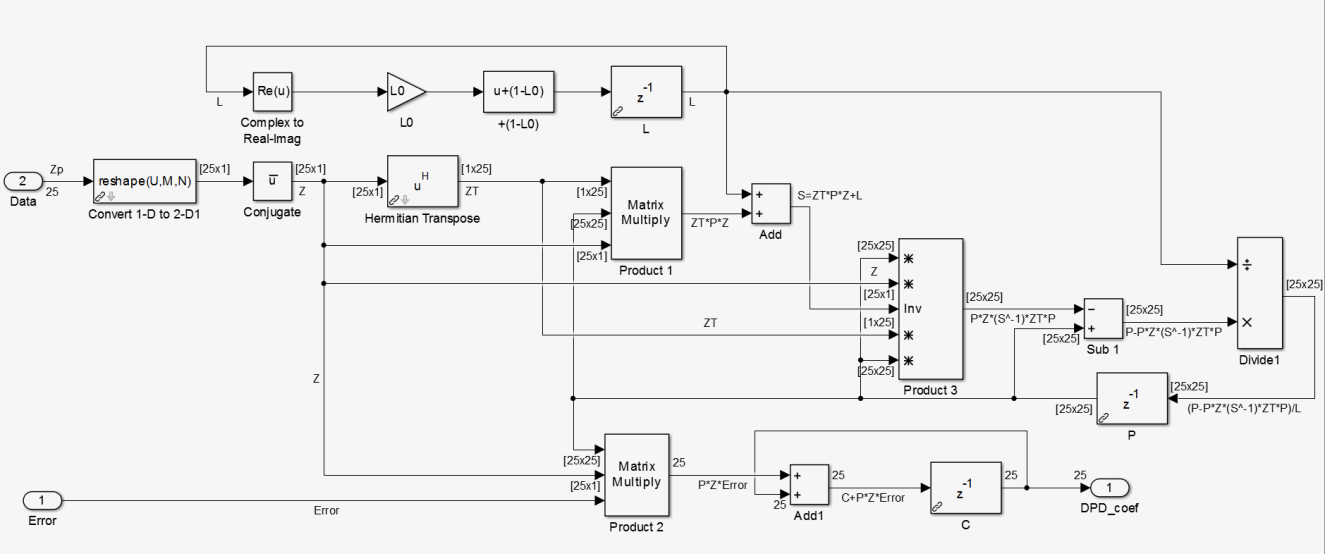


Figure 16. The RPEM-based coefficient computation is far more complex than the LMS. This algorithm is summarized in Equation 24 in [2].

Both the LMS and RPEM algorithms require forming an error signal in a feedback loop. The error is the difference between the measured PA input and the estimated PA input. The algorithm attempts to drive this error to zero and in doing so converge on a best estimate of the DPD coefficients. We compare the performance of the two methods in Figures 15 and 16. The RPEM method provides significantly better results in terms of both spectral growth reduction and rate of convergence.

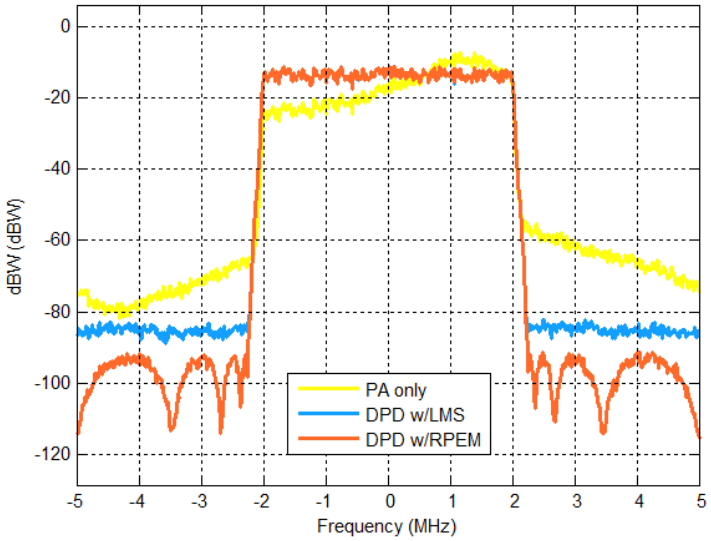


Figure 17. This plot compares the effectiveness of the LMS and RPEM algorithms at steady-state.

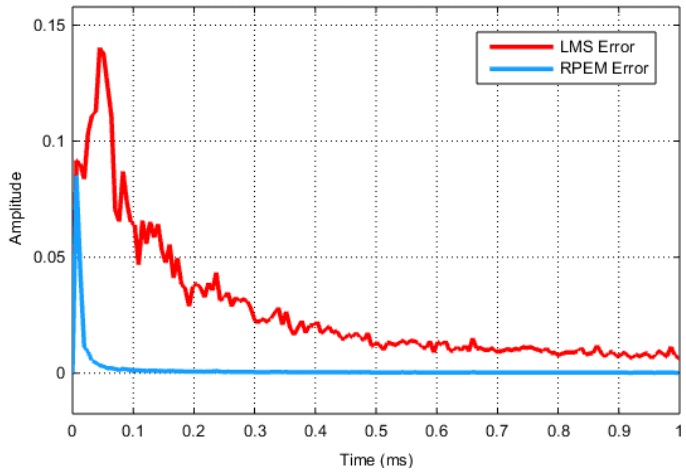


Figure 18. This plot compares the rate of coefficient convergence for the LMS and RPEM algorithms.

The RPEM algoritms converges to the same set of coefficients as computed from off-line PA measurements while the LMS algorithm never quite converges to this ideal solution. The RPEM algorithm does have its downside however. The RPEM algorithm using M = K = 5 requires approximately 75,300 multiplies per update. The LMS algorithm for the same M and K requires approximately 100 multiplies per update. The RPEM algorithms is thus 753 times more computationally expensive. In its current form the RPEM algorithm is not well suited for hardware implementation. On the other hand, the LMS-based approach is much better suited to hardware implementation but is relatively lacking in spectral regrowth reduction.

Summary

This paper demonstrated a workflow for modeling and simulating PAs and DPD. We showed the DPD design process progressing from an off-line batch processing derivation involving matrix inverse computations to a fully streaming and adaptive implementation involving no matrix inverses. We used three figures of merit to evaluate the adaptive DPD design's effectiveness: spectral regrowth reduction, rate of convergence, and computational complexity. We compared two variants of the adaptive indirect learning architecture, one based on the LMS algorithm and the second based on the RPEM algorithm. The RPEM algorithms was shown to be superior in terms of spectral regrowth reduction and rate of convergence but is prohibitive in terms of computational cost.

We used a combination of MATLAB and Simulink for this project. MATLAB was used to define system parameters, test individual algorithms, and perform off-line compuations. Simulink was used to integrate the individual PA and DPD components together into a test harness where feedback loops, data sizes, and design partitions are easily discerned.

Products Used
MATLAB
Simulink
Signal Processing Toolbox
DSP System Toolbox

References

1. Morgan, Ma, Kim, Zierdt, and Pastalan. "A Generalized Memory Polynomial Model for Digital Predistortion of RF Power Amplifiers". IEEE Trans. on Signal Processing, Vol. 54, No. 10, Oct. 2006

2. Li Gan and Emad Abd-Elrady. "Digital Predistortion of Memory Polynomial Systems Using Direct and Indirect Learning Architectures". Institute of Signal Processing and Speech Communication, Graz University of Technology.

3. Saleh, A.A.M., "Frequency-independent and frequency-dependent nonlinear models of TWT amplifiers," IEEE Trans. Communications, vol. COM-29, pp.1715-1720, November 1981