Direct Error-Searching SPSA-Based Model Extraction for Digital Predistortion of RF Power Amplifiers

Noel Kelly, *Student Member, IEEE* and Anding Zhu, *Senior Member, IEEE*

***Abstract*— This paper presents a low-complexity architecture to extract model coefficients for digital predistortion of radio frequency power amplifiers. The proposed approach directly updates the model coefficients online using a stochastic optimization algorithm that utilizes random perturbation of the model coefficients to determine the coefficient updating direction and converge toward the optimum solution. This technique avoids resource-intensive matrix operations and the requirement for an offline error model in the conventional model extraction techniques and thus drastically reduces the implementation complexity. The complete model extraction solution has been implemented on a field-programmable gate array, and it is shown that the hardware resource usage is remarkably low. Experimental measurements were conducted on a gallium nitride Doherty amplifier excited by Long Term Evolution signals and the results showed that the proposed technique can achieve linearization performance comparable to that obtained by using the conventional and significantly more complex solutions.**

***Index Terms*— Digital predistortion (DPD), linearization, model extraction, power amplifier (PA), stochastic optimization.**

I. Introduction

DIGITAL predistortion (DPD) is an advanced linearization technique that is now widely used to compensate for nonlinear behavior of radio frequency (RF) power amplifiers (PAs) in modern wireless communication systems [1], [2]. DPD uses an inverse model of the nonlinear PA to predistort the input signal at digital baseband. To maximize linearity improvement, an accurate behavioral model is required. In recent years, a range of advanced behavioral models for RF PAs have been developed, including modified versions of the Volterra series [3]–[5] and, more recently, the decomposed vector rotation (DVR)-based model which uses the absolute value operator as the basis function [6].

To extract the coefficient values for these models, least squares (LS)-based algorithms are typically used [7]–[9]. The LS algorithm offers high accuracy and fast convergence but comes with a high implementation cost in terms of hardware resources as it requires complex matrix multiplication and inversion operations [10]. High implementation complexity is particularly undesirable for applying DPD in small-cell base stations that are expected to form a large part of future 5G networks. These stations operate at much lower power levels than those in conventional larger cells. The power efficiency and implementation cost of all components in the transmitter chain, including DPD, must be carefully managed [11], [12]. To reduce the computational complexity, iterative coefficient extraction techniques can be considered. In particular, the recursive least-squares (RLS) algorithm has been employed in DPD model extraction [13], [14]. RLS avoids large matrix inversion, but maintaining an accurate approximation of the Hessian matrix still requires significant complexity, particularly for higher order models. In [15] and [16], the authors proposed a model adaption technique based on the least mean squares (LMS) algorithm where large matrix calculations are avoided and thus implementation complexity is greatly reduced. However, because it uses the first-order approximation, LMS is very sensitive to the adaption step size and it typically struggles to achieve the desired model accuracy [17].

In [18], a stochastic optimization-based DPD coefficient calculation technique was proposed as a low-complexity alternative to the LS solution. It is derived from the simultaneous perturbation stochastic approximation (SPSA) algorithm that uses measurements of the loss function with a random perturbation on the model coefficients to determine the coefficient updating direction and converge toward the optimum solution without involving resource-intensive matrix operations [19], [20]. It is shown in [18] that, after a sufficient number of iterations, the technique can achieve accuracy comparable to the existing LS solutions with over 98% reduction in computational complexity. However, to enable quadratic interpolation, new error model outputs must be calculated at each SPSA iteration. If a large number of training samples are used, the calculation of error model outputs still requires a large number of operations, leading to significant resource usage and cost.

This paper presents an alternative training architecture that removes the requirement for an additional error model in the coefficient extraction procedure. The proposed technique applies the SPSA algorithm directly on the DPD model to find the optimum coefficients. Experimental results show that the proposed approach can achieve comparable linearization performance to existing solutions but offers a further drastic reduction in hardware resource usage compared with the existing technique in [18].

Manuscript received May 10, 2017; revised July 10, 2017; accepted August 3, 2017. Date of publication October 16, 2017; date of current version March 5, 2018. This work was supported in part by the Science Foundation Ireland and in part by the European Regional Development Fund under Grant 13/RC/2077 and Grant 12/IA/1267. *(Corresponding author: Noel Kelly.)*

The authors are with the School of Electrical and Electronic Engineering, University College Dublin, 4 Dublin, Ireland (e-mail: noel.kelly.1@ucdconnect.ie; anding.zhu@ucd.ie).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TMTT.2017.2748128

0018-9480 © 2017 IEEE. Translations and content mining are permitted for academic research only. Personal use is also permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

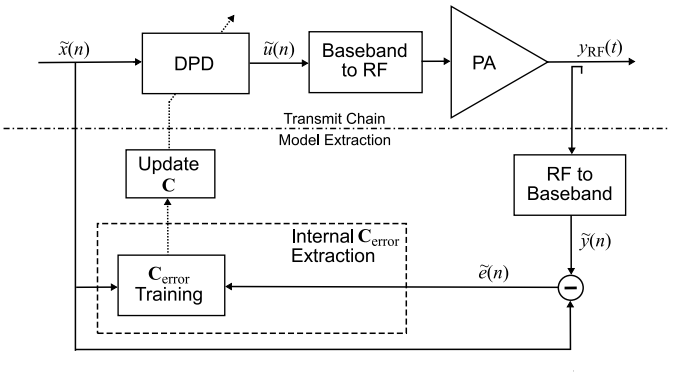


Fig. 1. DPD with direct learning model extraction.

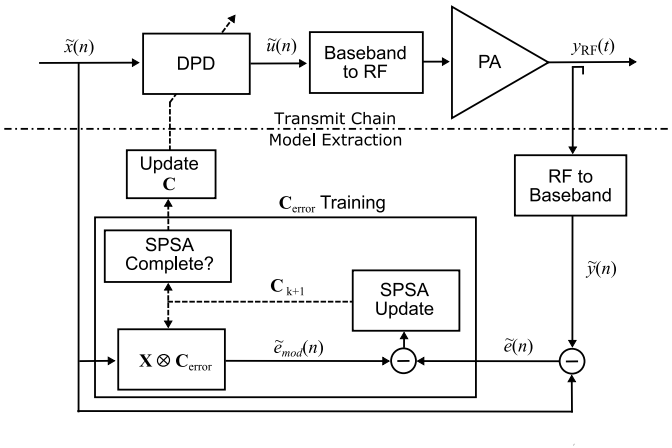


Fig. 2. SPSA-based direct learning DPD [18]. (Note the symbol "⊗" represents the Kronecker product.)

This paper is organized as follows. In Section II, the existing direct learning methods using both LS and the SPSA algorithm are discussed. The direct learning model extraction solution proposed in this paper is detailed in Section III. Section IV outlines the implementation complexity of the proposed technique and quantifies the improvement by comparing it against the existing SPSA-based DPD. Finally, Section V reports experimental results with a conclusion in Section VI.

II. Existing Direct Learning Model Extraction

The principle of DPD is that a digital block is inserted into the transmitter chain to preprocess the input signal before it enters the RF PA. Two primary architectures are commonly used for model extraction: indirect learning (IDL) and direct learning [2], [9], [21]. The IDL architecture estimates the postinverse of the PA first and then copies the coefficients to the preinverse DPD block. The direct learning architecture is usually used in a closed-loop system and it directly compares the PA output with the original input.

A. LS-Based Direct Learning

Fig. 1 shows the block diagram of a DPD system with direct learning. Many behavioral models can be used for constructing the DPD function. In this paper, we use the DVR-based model [6]. The predistorted signal $\tilde{u}(n)$ is given by

$$\begin{aligned}\tilde{u}(n) = &\sum_{i=0}^{M} \tilde{a}_i \tilde{x}(n-i) \\ &+ \sum_{s=1}^{S}\sum_{i=0}^{M} \tilde{c}_{s,i,1}\big||\tilde{x}(n-i)| - \beta_s\big|e^{j\theta(n-i)} \\ &+ \sum_{s=1}^{S}\sum_{i=0}^{M} \tilde{c}_{s,i,21}\big||\tilde{x}(n-i)| - \beta_s\big|e^{j\theta(n-i)}|\tilde{x}(n)| \\ &+ \ldots \end{aligned} \tag{1}$$

where $\tilde{x}(n)$ is the baseband input signal. The constants M and S are the memory length and the number of thresholds, respectively. The operator $|\cdot|$ denotes the absolute value operation and $\theta(n)$ is the phase of the signal $\tilde{x}(n)$. It is common to consider DPD processing to take place in blocks of N samples. In this paper, we represent vectors containing samples of the signals $\tilde{x}(n)$, $\tilde{u}(n)$, and $\tilde{y}(n)$ in Fig. 1 using the notation $\mathbf{x}$, $\mathbf{u}$, and $\mathbf{y}$, respectively. For convenience, we also group the model coefficients in (1) into a single coefficient vector

$$\mathbf{C} = [\tilde{a}_1, \tilde{a}_2, \cdots, \tilde{c}_{s,1,1}, \cdots]. \tag{2}$$

The direct learning model extraction architecture iteratively adjusts the coefficients in $\mathbf{C}$ to minimize the error between $\mathbf{y}$ and $\mathbf{x}$. As shown in Fig. 1, this means the DPD model is located inside the training loop. The update equation is given by

$$\mathbf{C}_{h+1} = \mathbf{C}_h - \lambda \cdot \Delta\mathbf{C}_h \tag{3}$$

where λ is a scalar adaption factor and h is the iteration index. $\Delta\mathbf{C}_h$ is the coefficient updating vector modeling the error component in the DPD coefficient vector. Provided the error is sufficiently small, the error in the DPD output signal, $u_{\text{error},h}$, can be approximated by the error in the PA output

$$\mathbf{u}_{\text{error},h} = G^{-1}(\mathbf{y}_h - \mathbf{x}) \approx \mathbf{y}_h - \mathbf{x} \tag{4}$$

where $G^{-1}(\cdot)$ is the inverse transfer function of the PA. The coefficient update vector $\Delta\mathbf{C}_h$ can then be estimated using LS

$$\Delta\mathbf{C}_h = (\mathbf{X}^H\mathbf{X})^{-1}\mathbf{X}^H(\mathbf{u}_{\text{error},h}) \tag{5}$$

where $\mathbf{X}$ is the DPD model regression matrix generated using the DPD input signal $\mathbf{x}$ [22].

B. SPSA-Based Direct Learning

The LS operation in (5) involves large matrix operations, which are hardware demanding and time consuming. To reduce complexity, in [18], we proposed to replace LS with the SPSA algorithm. SPSA is a stochastic optimization algorithm that iteratively measures a loss function with a random perturbation on the model coefficients to determine the coefficient updating direction and finally find the optimum solution. The coefficient perturbation process only requires a simple addition and subtraction operation, and all coefficients are randomly perturbed together, which leads to substantial savings in hardware resource usage in model extraction.

To replace LS with SPSA to calculate $\Delta\mathbf{C}_h$ in (5), we first need to construct an error model with output $\mathbf{e}_{\text{mod}}$ given by

$$\mathbf{e}_{\text{mod}} = \mathbf{X}\mathbf{C}_{\text{err}}. \tag{6}$$

The goal of the SPSA process is to find the optimum error coefficient vector $\mathbf{C}_{\text{err}}$ to enable $\mathbf{e}_{\text{mod}}$ approach $\mathbf{u}_{\text{error}}$, so that $\mathbf{C}_{\text{err}}$ can then be used to replace $\Delta\mathbf{C}_h$ in (3) to update the DPD coefficients in the next direct learning iteration. As mentioned above, SPSA is an iterative search algorithm and thus, as shown in Fig. 2, multiple internal iterations are required to find the optimum $\mathbf{C}_{\text{err}}$ coefficients each time.

The iteration procedure is outlined below. Note that we use the index k to denote internal iterations as opposed to the index h in (3). The iteration begins with perturbing the current estimate of the error model coefficient vector $\mathbf{C}_{\text{err},k}$ with a random perturbation vector, Δ_k, weighted by a scalar, c_k, to generate two additional coefficient vectors

$$\begin{aligned}\mathbf{C}^{+}_{\text{err},k} &= \mathbf{C}_{\text{err},k} + c_k\Delta_k \\ \mathbf{C}^{-}_{\text{err},k} &= \mathbf{C}_{\text{err},k} - c_k\Delta_k.\end{aligned} \tag{7}$$

The perturbation vector Δ_k is given by

$$\Delta_k = [\Delta_{k,1}, \Delta_{k,2}, \cdots, \Delta_{k,K}]. \tag{8}$$

where K is the number of terms in the model and each entry $\Delta_{k,i}$ is either $+1$ or -1 with equal probability. Using the perturbed coefficients in (7) and the current coefficient set $\mathbf{C}_{\text{err},k}$, three error model outputs are calculated

$$\begin{aligned}\mathbf{e}_{\text{mod},k} &= \mathbf{X}\mathbf{C}_{\text{err},k} \\ \mathbf{e}^{+}_{\text{mod},k} &= \mathbf{X}\mathbf{C}^{+}_{\text{err},k} \\ \mathbf{e}^{-}_{\text{mod},k} &= \mathbf{X}\mathbf{C}^{-}_{\text{err},k}.\end{aligned} \tag{9}$$

By comparing with the desired error vector $\mathbf{u}_{\text{error}}$, three normalized mean square error (NMSE) loss function measurements, $L(\mathbf{C}_{\text{err},k})$, $L(\mathbf{C}^{+}_{\text{err},k})$, and $L(\mathbf{C}^{-}_{\text{err},k})$ can be obtained. As discussed in [18], since the NMSE is quadratically related to the model coefficients, the next updated coefficient estimate, $\mathbf{C}_{\text{err},k+1}$, can be found by moving directly to the minimum point of the quadratic curve formed by the three loss function measurements. Specifically, the new error coefficient vector is calculated by

$$\mathbf{C}_{\text{err},k+1} = \mathbf{C}_{\text{err},k} - c_k\Delta_k\mu_k \tag{10}$$

where the perturbation update weighting, μ_k is given by[1]

$$\mu_k = \frac{L\left(\mathbf{C}^{+}_{\text{err},k}\right) - L\left(\mathbf{C}^{-}_{\text{err},k}\right)}{2\left[L\left(\mathbf{C}^{+}_{\text{err},k}\right) + L\left(\mathbf{C}^{-}_{\text{err},k}\right) - 2L\left(\mathbf{C}_{\text{err},k}\right)\right]} \tag{11}$$

As shown in Fig. 3, as the updating process iterates, the NMSE is reduced in each iteration. The training finishes when the desired accuracy is reached.

This method avoids the gradient calculations and resulting resource-intensive matrix operations required by LS.

[1]Due to a typing error, the scaling factor of 2 was missing in the denominator part of the quadratic SPSA coefficients updating equation given in [18]. It is included in (11) now.

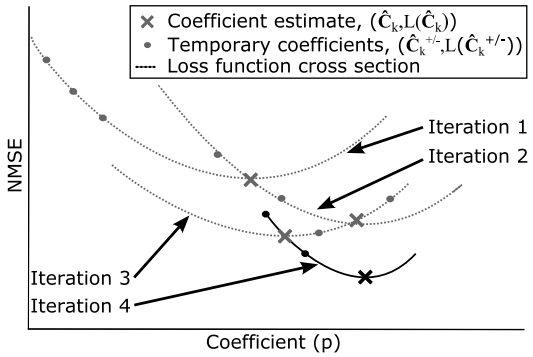


Fig. 3. Evolution of the quadratic SPSA interpolation.

As shown in [18], this approach achieves a reduction in per-iteration computational complexity of over 98%, while maintaining comparable performance to conventional LS. However, in terms of implementation complexity, the introduction of a secondary error model is not desirable. At each SPSA iteration, new error model outputs must be calculated according to the expressions in (9). Note that the matrix $\mathbf{X}_h$ has dimensions $N \times K$, where N is the number of training samples and K is the number of terms in the DPD model. In a practical system, this calculation requires a large number of operations, leading to significant resource usage and cost.

III. Direct Error-Searching SPSA-Based Model Extraction

To further reduce the implementation complexity of the technique in [18], this paper proposes a novel extraction method where the SPSA algorithm is applied directly to the DPD model rather than to the error model.

A. Error Analysis

In a DPD system, shown in Fig. 1, the goal of the model extraction process is to find a set of ideal coefficients that can generate the ideal output $\mathbf{u}_{\text{ideal}}$ that enters the PA to generate the perfect output, $\mathbf{y}_{\text{ideal}}$ that is equal to the original input $\mathbf{x}$. Clearly, $\mathbf{u}_{\text{ideal}}$ is not available before we find the ideal coefficients, but it can be expressed as

$$\mathbf{u}_{\text{ideal}} = \mathbf{u}_h - \mathbf{u}_{\text{error},h} \tag{12}$$

where $\mathbf{u}_h$ is the existing DPD output that can be obtained by

$$\mathbf{u}_h = \mathbf{X}\mathbf{C}_h \tag{13}$$

where $\mathbf{C}_h$ is the model coefficients vector. As in (4), for a given set of DPD coefficients, the error signal $\mathbf{u}_{\text{error},h}$ at the DPD output can be approximated by the error measured at the PA output, if the error is relatively small [22], [23]. Substituting (4) into (12) gives an approximate value for the ideal DPD output

$$\mathbf{u}_{\text{ideal}} \approx \mathbf{u}_h - (\mathbf{y}_h - \mathbf{x}). \tag{14}$$

If we know $\mathbf{u}_{\text{ideal}}$, the optimization task now is to find the ideal DPD coefficients, $\mathbf{C}_{\text{ideal}}$, that minimize the error between the

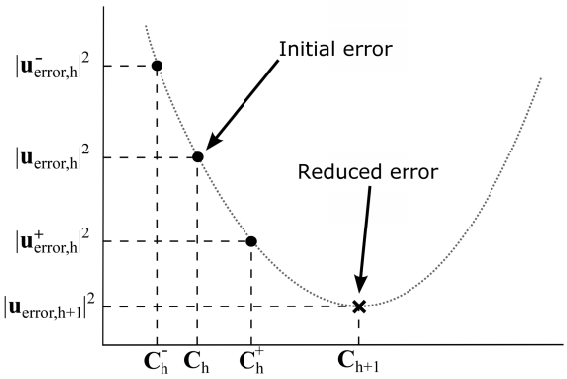


Fig. 4. Single SPSA iteration of the proposed algorithm.

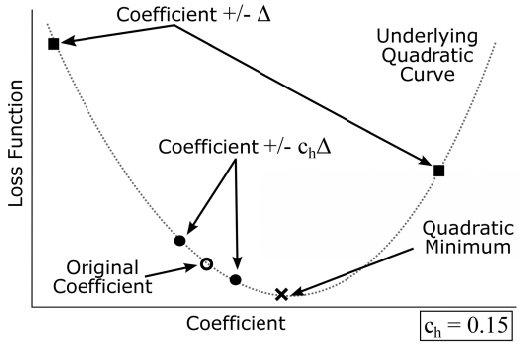


Fig. 5. Quadratic fitting.

DPD model output, $\mathbf{u}_h$, and its ideal value, $\mathbf{u}_{\text{ideal}}$

$$\mathbf{C}_{\text{ideal}} = \arg\min_{\mathbf{C}_h} \left(|\mathbf{u}_h - \mathbf{u}_{\text{ideal}}|^2\right). \tag{15}$$

Considering the quadratic feature of SPSA, this optimization problem can be solved by using SPSA directly, as illustrated in Fig. 4, where the coefficients $\mathbf{C}_h$ are on the horizontal axis, while the magnitude square of the output error signal $|u_{\text{error},h}|^2$ is on the vertical axis. Because both $\mathbf{u}_h$ and $\mathbf{u}_{\text{ideal}}$ are linearly related to the model coefficients, $\mathbf{u}_{\text{error},h}$ is also linearly related to $\mathbf{C}_h$. Therefore, $|u_{\text{error},h}|^2$ is quadratic in relation to $\mathbf{C}_h$. It follows that the next error and the corresponding coefficients can be found by simply forming a quadratic curve using SPSA. As shown in Fig. 4, assuming that $\mathbf{C}_h$ corresponds to the existing error $|u_{\text{error},h}|^2$, if we perturb the coefficients to generate two other errors $|u^+_{\text{error},h}|^2$, $|u^-_{\text{error},h}|^2$, we can form a quadratic curve. The minimum point of the curve is the next error value $|u_{\text{error},h+1}|^2$, and thus, the new corresponding coefficients $\mathbf{C}_{h+1}$ can be found.

The difference compared with the existing approaches described in Section II is shown as

$$\begin{aligned} &\text{Existing: } \mathbf{C}_h \Rightarrow \mathbf{u}_{\text{error},h} \Rightarrow \Delta\mathbf{C}_h \Rightarrow \mathbf{C}_{h+1} \\ &\text{Proposed: } \mathbf{C}_h \Rightarrow \mathbf{u}_{\text{error},h} \Rightarrow \mathbf{u}_{\text{error},h+1} \Rightarrow \mathbf{C}_{h+1} \end{aligned} \tag{16}$$

where in the existing approaches, the next coefficients vector is updated using an error coefficients vector generated from the existing error, while in the proposed approach, we find the next DPD error from the SPSA quadratic curve fitting and thus find the next optimum coefficient vector directly. This approach is much simpler in terms of computational complexity and hardware implementation compared with the existing approaches, discussed as follows.

B. SPSA Training

To apply SPSA, we first define a loss function. In this paper, we propose to use a loss function that is given as the residual sum of squares (RSS) between the DPD output $\mathbf{u}$ and the ideal DPD output $\mathbf{u}_{\text{ideal}}$ for a given set of measurements

$$\text{RSS}(\mathbf{u}_{\text{ideal}}, \mathbf{u}) = \sum_{n=1}^{N} |\tilde{u}(n) - \tilde{u}_{\text{ideal}}(n)|^2 \tag{17}$$

where N is the total number of training samples. It is similar to the standard NMSE. The difference is that the division operation in the NMSE definition is no longer required here as $\mathbf{u}_{\text{ideal}}$ is constant across all three measurements during a single SPSA iteration.

As discussed earlier, to perform quadratic SPSA, three loss function measurements are required. One measurement is conducted by using the existing coefficients $\mathbf{C}_h$, while the other two measurements are obtained by applying a random perturbation vector Δ_h, weighted by a scalar c_h, to the existing coefficients $\mathbf{C}_h$ to generate two additional DPD output signals. Intuitively, we would think that we have to feed the coefficients through the DPD block three times to generate three DPD outputs, which will complicate the process. In fact, by taking advantage of the quadratic property of SPSA, it is possible to obtain the other two outputs with simple addition and subtraction operations on the existing output, explained as follows.

First of all, note that the two additional perturbed DPD outputs are used to form the quadratic curve and find the next DPD error only. They will not enter the PA to produce new final outputs. The perturbation error levels therefore do not affect the real-time system operation. Second, because we have ensured a quadratic relationship between the coefficients and loss function, with the same input signal, all possible loss function measurements lie on the same quadratic curve, no matter what weighting factor c_h is used. As shown in Fig. 5, two solid square points are generated from $c_h = 1$, while two solid round dot points are generated with $c_h = 0.15$. All four points lie on the same quadratic curve. This means that no matter what weighting factor is used, the next minimum point can always be found after three measurements. Although the perturbation errors are bigger in the case of $c_h = 1$ than those with $c_h = 0.15$, these errors do not affect the real system operation since the perturbed coefficients will not be used directly in the real-time DPD operation. As a result, to reduce computational complexity, here we choose $c_h = 1$ and the perturbed model outputs, $\mathbf{u}^+$ and $\mathbf{u}^-$ are given by

$$\begin{aligned} \mathbf{u}^+ &= \mathbf{X}(\mathbf{C}_h + \Delta_h) \\ \mathbf{u}^- &= \mathbf{X}(\mathbf{C}_h - \Delta_h). \end{aligned} \tag{18}$$

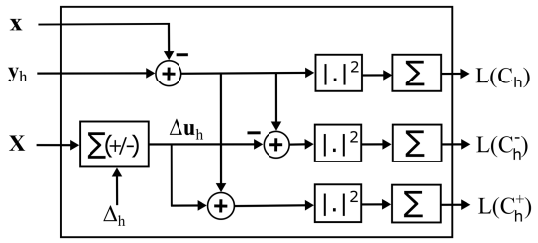


Fig. 6. Loss function measurement implementation.

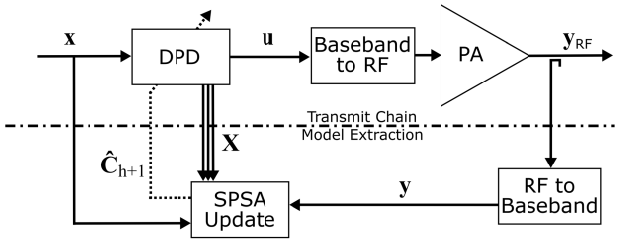


Fig. 7. Proposed direct error-searching SPSA model extraction.

Expanding (18) gives

$$\mathbf{u}^+ = \mathbf{X}(\mathbf{C}_h + \Delta_h) = \mathbf{u}_h + (\mathbf{X}\Delta_h)$$
$$\mathbf{u}^- = \mathbf{X}(\mathbf{C}_h - \Delta_h) = \mathbf{u}_h - (\mathbf{X}\Delta_h) \tag{19}$$

meaning the coefficient perturbation can be expressed as an additional signal, $\Delta\mathbf{u}_h$, to be added to and subtracted from the original DPD output

$$\mathbf{u}^+ = \mathbf{u}_h + \Delta\mathbf{u}_h$$
$$\mathbf{u}^- = \mathbf{u}_h - \Delta\mathbf{u}_h \tag{20}$$

where

$$\Delta\mathbf{u}_h = \mathbf{X}\Delta_h. \tag{21}$$

Since the elements in Δ_h are either +1 or −1, $\Delta\mathbf{u}_h$ in (21) can be calculated with simple addition and subtraction operations between each column of the model regression matrix, $\mathbf{X}$ defined in Section II-A

$$\begin{bmatrix} X_{1,1} & X_{1,2} & X_{1,3} & \cdots \\ X_{2,1} & X_{2,2} & X_{2,3} & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix} \begin{bmatrix} -1 \\ +1 \\ -1 \\ \vdots \end{bmatrix} = \begin{bmatrix} -X_{1,1} + X_{1,2} - X_{1,3} + \cdots \\ -X_{2,1} + X_{2,2} - X_{2,3} + \cdots \\ \vdots \end{bmatrix} \tag{22}$$

After obtaining $\Delta\mathbf{u}_h$ and considering (14) and (17), three loss function measurements can be conducted

$$L(\mathbf{C}_h) = \sum_{n=1}^{N} |\tilde{y}_h(n) - \tilde{x}(n)|^2$$
$$L(\mathbf{C}_h^+) = \sum_{n=1}^{N} |\tilde{y}_h(n) - \tilde{x}(n) + \Delta\tilde{u}_h(n)|^2$$
$$L(\mathbf{C}_h^-) = \sum_{n=1}^{N} |\tilde{y}_h(n) - \tilde{x}(n) - \Delta\tilde{u}_h(n)|^2 \tag{23}$$

as shown in Fig. 6. The results from (23) along with the three coefficients sets can be used to form a quadratic curve for each coefficient and then new coefficients can be found by using the same equations in (10) and (11). The updated DPD coefficients produce a closer match to $\mathbf{u}_{\text{ideal}}$ at the DPD model output. This operation significantly simplifies the updating process and reduces system cost. More details on the digital implementation will be given in Section IV.

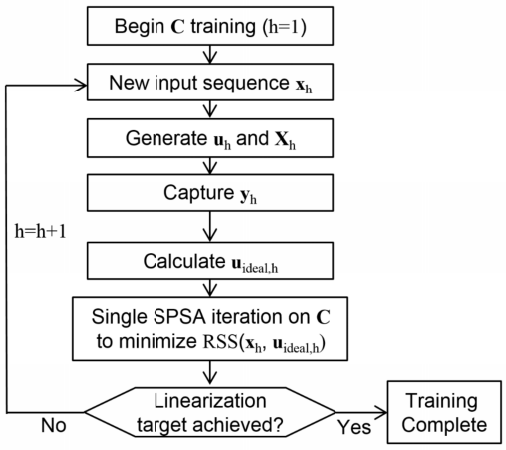


Fig. 8. Proposed model training routine.

The fact that the measurements in (23) are performed at the DPD output is important for two reasons. First, the desired quadratic interpolation SPSA algorithm can be directly applied. Second, the three loss function measurements can be conducted at the same time and there is no need to feed the two perturbed DPD output signals through the PA to measure the loss function, which means that the model training process does not disrupt the real-time DPD operation. The block diagram of the proposed full system is shown in Fig. 7.

C. Complete Model Extraction Procedure

Despite the use of the term "ideal" in the notation, the approximation in (4) limits the accuracy of the calculated $\mathbf{C}_{\text{ideal}}$ coefficients. The signal $\mathbf{u}_{\text{ideal}}$ is not the "truly" ideal DPD output but rather an estimated ideal output based on the approximation in (4). Thus, even if we could train the DPD coefficients to exactly fit $\mathbf{u}_{\text{ideal}}$, the error at the PA output would not be completely removed. The traditional direct learning architecture faces an identical problem. Furthermore, in the above training, we assume $\mathbf{u}_{\text{ideal}}$ is generated from a fixed set of input signal samples $\mathbf{x}$. In real operation, the input signal $\mathbf{x}$ is randomly generated over time and a different $\mathbf{x}$, e.g., $\mathbf{x}_h$ corresponds to a new $\mathbf{u}_{\text{ideal},h}$. To perform accurate coefficient extraction in this environment, multiple iterations of the coefficient extraction process are required.

The complete model training flow is depicted in Fig. 8. The training procedure can be described as follows.

1) Set iteration index, $h = 1$, and choose initial DPD coefficient set $\mathbf{C}_1$.

2) Measure the PA output signal, $\mathbf{y}_h$.
3) Calculate the approximate DPD error signal, $\mathbf{u}_{\text{error},h}$ according to (4).
4) Calculate the ideal DPD signal, $\mathbf{u}_{\text{ideal}}$ according to (14).
5) Measure $RSS\left(\mathbf{u}_{\text{ideal}}, \mathbf{u}_h\right)$, $RSS\left(\mathbf{u}_{\text{ideal}}, \mathbf{u}_h^+\right)$, and $RSS\left(\mathbf{u}_{\text{ideal}}, \mathbf{u}_h^-\right)$.
6) Calculate $\mathbf{C}_{h+1}$ using the SPSA update equations in (10) and (11) to minimize error between DPD output $\mathbf{u}_h$ and $\mathbf{u}_{\text{ideal}}$.
7) Generate new DPD output using $\mathbf{C}_{h+1}$ and pass to the PA. If linearization criterion is satisfied, finish training, if not, update $h = h + 1$ and return to step 2.

This iterative training process shares similarities with the conventional direct learning procedure. At each iteration, the DPD coefficients are updated to approach an estimate of the ideal predistorted signal. Provided the approximation in (14) holds, the error at the PA output is reduced after each DPD coefficient update. It follows that the accuracy of the approximation in (4), on which (14) is based, also improves with each update and the DPD coefficients are trained to approach a more accurate estimate of the ideal output at each iteration.

D. Comparison With the Existing Approaches

Although an iterative coefficient updating process is employed, the proposed technique in this paper is fundamentally different from the conventional iterative approaches. In the existing systems, LMS may be employed but its performance is poor because LMS is a first-order approximation algorithm that converges very slowly and it is sensitive to the adaptation size. The second-order approximation approaches, such as LS and RLS, can achieve high accuracy but come with high implementation complexity.

The proposed approach is also different from the conventional SPSA where the gradient is approximated by the first-order line fitting. Such linear approximations are highly sensitive to the choice of weighting factor and can often struggle to converge [19]. By exploiting the quadratic relationship that exists between the loss function and the DPD coefficients, the optimum minimum point can be found directly using quadratic curve fitting instead of gradient approximation. This approach is equivalent to the second-order approximation that significantly speeds up the convergence and guarantees the optimum point can be found at each iteration. Furthermore, due to the quadratic relationship, the perturbation step size is no longer relevant because all the cost function measurement points will fall on the same curve which leads that we could use any perturbation step size. In this paper, we directly use $+/-1$, which enables the cost function to be directly obtained by adding and subtracting the basis waveforms that have already been generated by the DPD model, dramatically simplifying the hardware implementation.

While the optimization process is changed to the second-order approximation, the proposed approach still keeps the core feature of SPSA, namely, the optimum solution is found using loss function measurements with simultaneous random perturbation on model coefficients. It enables the algorithm to achieve comparable accuracy to LS solutions but with a very low implementation cost. We call the proposed approach "direct error-searching" SPSA-based model extraction.

TABLE I
SD-SPSA PER-ITERATION COMPLEXITY [18]

Operation	Real Mults.	Real Adds.
Calculate $L(\mathbf{C}_h)$, $L(\mathbf{C}_h^+)$, and $L(\mathbf{C}_h^-)$	$4 \times N$ (32768)	$8 \times N$ (65536)
Coefficient perturbation and update	$4 \times K$ (200)	$10 \times K$ (500)
Error model calculations	$6NK$ (2457600)	$4N(K-1)+10NK$ (5701632)
Total	$4N+4K$ $+6NK$ (2490568)	$8N+10K+10NK$ $+4N(K-1)$ (5767668)

(No. of training samples: N=8192, No of DPD coefficients: K=50)

IV. HARDWARE IMPLEMENTATION AND COMPLEXITY COMPARISON

To quantify the improvement offered by the direct error-searching SPSA approach outlined above, we compare it with the generic SPSA-based DPD extraction method proposed in [18], which we refer to from here on as SD-SPSA. It is worth noting that the SD-SPSA method has already achieved a 98% reduction compared with conventional LS algorithms, as discussed in detail in [18]. To avoid replication, in this paper, we only discuss further reduction from the SD-SPSA solution, without comparison with LS.

A. Operations per Iteration

Both the SD-SPSA and the direct error-searching SPSA use multiple iterations, in this section, we compare the complexity per iteration.

The SD-SPSA solution directly replaces LS with SPSA in the closed-loop direct learning architecture. In this case, the SPSA algorithm calculates the error coefficients, $\mathbf{C}_{\text{err}}$, using an error model in the same format as the LS estimation technique would be applied. The solution in [18] employs a novel steep descent SPSA algorithm to increase the training speed and results showed that the linearization performance is comparable with the existing LS solutions. However, as discussed in Section II-B, additional processing associated with the error model substantially increases resource usage.

Table I reports the operations that account for the majority of the real multiplication and addition operations used in a single iteration of the algorithm in [18]. Generating the error model output accounts for the vast majority of the complexity. Taking a typical example of a DPD model with 50 terms ($K = 50$) and using 8192 samples to calculate the NMSE on each iteration ($N = 8192$), running the error model accounts for over 98% of the total real multiplications and over 99% of the total real addition operations performed each iteration.

The solution proposed in this paper removes the need for an error model. It finds the next coefficients directly by perturbing

TABLE II
PROPOSED APPROACH PER-ITERATION COMPLEXITY

Operation	Real Mults.	Real Adds.
Generate $\Delta\mathbf{u}$	0 (0)	$2NK + 2N(K-1)$ (1622016)
Measure $L(\mathbf{C}_h)$, $L(\mathbf{C}_h^+)$, and $L(\mathbf{C}_h^-)$	$6 \times N$ (49152)	$12 \times N$ (98304)
Calculate coefficient update	$2 \times K$ (100)	$2 \times K$ (100)
Total	$6N + 2K$ (49252)	$2NK + 2N(K-1)$ $+12N + 2K$ (1720420)

(No. of training samples: N=8192, No of DPD coefficients: K=50)

the DPD model. As discussed in Section III-B, the real and imaginary components of each term in Δ_h are limited to values of $+/-1$. In this scenario, the $\Delta\mathbf{u}_h$ vector can be calculated using only simple addition and subtraction operations across the columns of the matrix $\mathbf{X}$. This amounts to a substantial reduction in the number of operations required per iteration and also allows the SPSA update equation to be evaluated without the need for costly multiplication operations.

Table II reports the number of real multiplication and addition operations required per iteration for the proposed approach. Removing the need to run an offline error model each iteration substantially reduces the number of multiplication operations required. To quantify the improvement, resource usage for a practical training scenario with K =50 and N =8192 is shown in brackets. Referring back to Table I, in this scenario, the SD-SPSA technique in [18] requires approximately 2.5×10^6 real multiplications and 5.7×10^6 real additions. By comparison, as shown in Table II, the proposed error-matching SPSA uses approximately 5×10^4 real multiplications and 1.7×10^6 real additions.

It is also important to note that, in addition to the complexity reported in Table I, the technique in [18] requires the DPD matrix $\mathbf{X}$ and the captured PA output $\mathbf{y}$ to remain constant throughout the offline training segment. This leads to a complexity tradeoff in the design. On the one hand, as proposed in [18], the matrix $\mathbf{X}$ can be stored after it has been generated by the DPD model and reused for each SPSA iteration, but this requires a large block of memory resources. On the other hand, the system could store only the input signal $\mathbf{x}$ and implement a secondary offline model to generate $\mathbf{X}$ each iteration, reducing the memory requirements but significantly increasing the computational complexity.

B. Hardware Implementation Complexity

The proposed extraction solution was synthesized for implementation in a field-programmable gate array (FPGA). The designed system performs all of the computation necessary to update the model coefficients each iteration: it takes the signals, $\mathbf{x}$, $\mathbf{X}$, and $\mathbf{y}$ as inputs and processes them to generate the SPSA update vector.

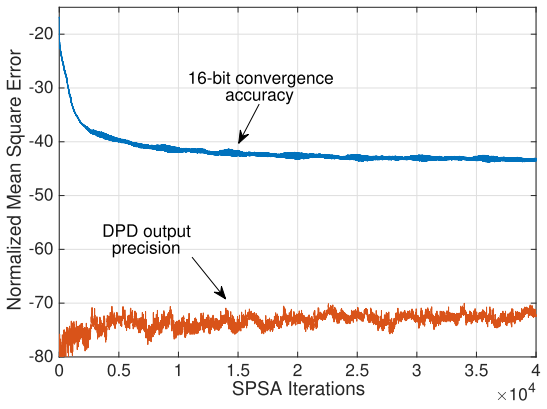


Fig. 9. Proposed hardware implementation accuracy.

TABLE III
HARDWARE IMPLEMENTATION DETAILS

Operation	Slice LUTs	Slice Registers	Power Consumption
Signal Generation	342	447	27 mW
Loss Function Calculation	3269	5000	137 mW
SPSA Update	423	766	30 mW
Total	4034 (3.98%*)	6213 (3.06%*)	194 mW

*% usage of total available resources for Xilinx XC7K160T [24]

The complex samples are represented using 32-b precision, 16 b each for the real and imaginary components. A hardware simulation was performed to confirm the design feasibility. The simulation used digital baseband samples and a full precision model of the DPD-PA transmit chain. The DPD model is a DVR-based function with $S = 8$ and $M = 3$, and the PA is modeled using a dynamic deviation reduction-based Volterra series with nonlinear terms up to the ninth order and memory length 3 [8]. The PA model is based on data measured from a gallium nitride (GaN) Doherty PA. A 20-MHz Long Term Evolution (LTE) signal serves as the system input. The primary objective of the simulation is to confirm that the 16-b implementation provides sufficient precision that the algorithm convergence is not affected. Fig. 9 shows the NMSE measured after each iteration between the DPD output signal generated using the 16-b FPGA coefficients and the output signal generated using full precision coefficients. The NMSE remains below -70 dB throughout the 40 000 iteration training period, confirming there is no significant degradation due to rounding error. Fig. 9 also reports the NMSE measured between the system input and output for the bit-accurate model extraction scenario. The system NMSE reaches the full precision performance of approximately -43 dB, confirming the implemented design achieves sufficient precision.

Table III reports digital hardware resource usage for the implemented system. To demonstrate its simplicity, the complete extraction solution is implemented in lookup tables

(LUTs) and registers only—no specialized DSP units are used. The system loads the DPD matrix columns serially, meaning that the hardware usage reported in Table III is independent of the DPD model length. Serial loading allows a single accumulator to calculate $\Delta \mathbf{u}_h$. This reduces hardware usage but requires K clock cycles per input sample, where K is the number of terms in the DPD model and thus columns in the DPD matrix. Alternatively, if maximum throughput is required, more than one column may be loaded simultaneously; full parallel loading corresponds to the greatest hardware resource usage but allows a new sample to be processed on every clock cycle. This is one of a number of design tradeoffs that can be made in implementing the proposed system. By requiring only three loss function measurements to perform a coefficient update, the flexibility of the SPSA algorithm permits a large number of possible implementation strategies, tuned to different criteria such as minimum resource usage, maximum data throughput, or minimum power consumption. It is expected that, compared with the proof-of-concept solution reported in Table III, even lower resource usage can be achieved in the future systems by leveraging known application scenarios and employing more advanced hardware implementation tools.

Table III also includes power consumption figures for each of the main blocks of the proposed system. The power consumption measurements were obtained using the post implementation power analysis tool in the Xilinx Vivado integrated design environment software. It should be stressed that these figures are reported only to provide the reader with an approximate breakdown of the power consumption in the circuit. In a real implementation, power consumption of the real circuit is highly dependent on the application scenario, e.g., signal bandwidth/sampling rate, number of coefficients used, clock rate, digital circuit chip types (e.g., FPGA part number), and implementation strategy. As a result, the power consumption can vary largely in different cases. Nonetheless, it is interesting to note that the main SPSA operation, i.e., the model coefficient update, only requires an estimated 30 mW to operate. This emphasizes the very low implementation cost of the SPSA algorithm. The majority of the power consumption is due to the loss function calculation. As discussed later in Section VI, the loss function measurement may be implemented in a highly efficient way in the analog domain in the future that may provide the potential for further power reduction.

To quantify the reduction in resource usage, we compare the proposed technique with the offline SD-SPSA solution in [18]. In fact, the two designs share much of their infrastructure. We first simplify the SD-SPSA method by swapping NMSE with an RSS loss function and also ignoring the added complexity of the offline steep-descent calculation. The reduced complexity gives the SD-SPSA technique an advantage in the comparisons but allows a clearer comparison between the key features of the systems, namely, generating the loss function measurement signals. For a fair comparison, we use an implementation of the SD-SPSA algorithm in [18], in which the loss function is measured three times each iteration. In terms of computational complexity, the primary advantage of the proposed SPSA technique is in generating the RSS input signals, as shown in Fig. 6. Table IV compares guideline resource usage between the SD-SPSA solution and the error-searching SPSA technique. It can be seen that the complex multiplications required to run the error model consume far more resources than the simple addition and subtraction operations of the error-searching technique reported in Table III. Note that the resource usage in Table IV is for an efficient serial-loading implementation of the SD-SPSA algorithm, where only three complex multiplications are required. For higher throughput scenarios, complexity will increase with the number of parallel operations.

TABLE IV
ERROR SIGNAL GENERATION COMPLEXITY

Operation	Slice LUTs	Slice Registers
SD-SPSA	3297	3546
Proposed Approach	342	447

TABLE V
HARDWARE RESOURCE USAGE COMPARISON

	SD-SPSA	Proposed Approach	Reduction
Slice LUTs	8591	4034	53%
Slice Registers	10874	6213	43%
Memory (kb)	13369	0	100%

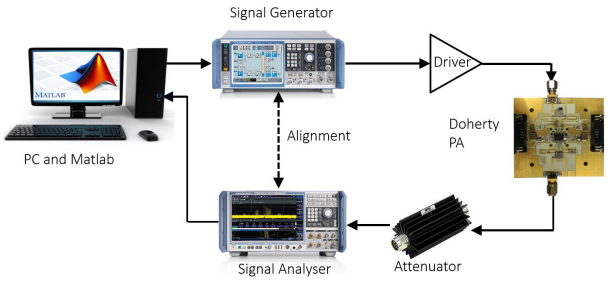


Fig. 10. Experimental test setup.

Table V compares the overall resource usage between the two architectures. In addition to the reduced computational resource requirements (i.e., raw LUTs and flip flops), the proposed technique no longer requires large blocks of signal samples to be stored. In [18], the algorithm requires the full $\mathbf{X}_h$ matrix and the measured error vector to be stored during the SPSA training run. Assuming 32-b accuracy for each complex value, for a test scenario with $K = 50$ and $N = 8192$, this amounts to a storage requirement of $32 \times ((N \times K) + N) = 13\,369$kb. As shown in Table V, the proposed technique requires no additional memory resources, substantially reducing implementation complexity.

V. EXPERIMENTAL RESULTS

A full RF test bench, as shown in Fig. 10, was set up to evaluate the performance of the proposed architecture in a

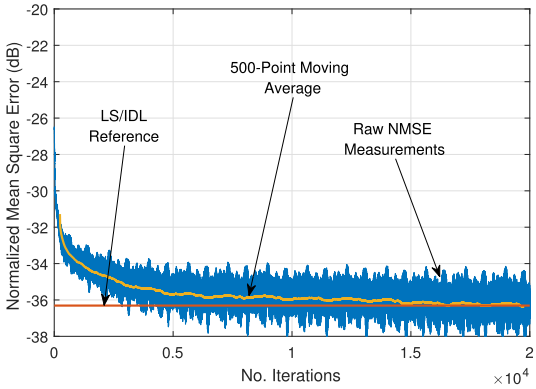


Fig. 11. Measured NMSE over 20 000 iterations with 20-MHz LTE signal.

real DPD training scenario. To evaluate the online coefficient updating process in hardware, an automated test routine was developed to run multiple iterations of the algorithm. The test bench is centered on a master PC which is responsible for the signal generation and control of the measurement equipment to capture the PA input and output signals. DPD coefficients are calculated in the PC according to the proposed SPSA algorithm by using the captured error signal.

At the beginning of each iteration, the predistorted signal is loaded onto a Rohde and Schwarz SMW200A vector signal generator where it is up-converted to RF before transmission to the PA. A linear driver boosts the signal power before it enters the DUT, a high-efficiency 20-W GaN Doherty PA operated with an average output power of 36 dBm. At the PA output, an attenuator reduces the signal power before a Rohde and Schwarz FSW signal and spectrum analyzer captures the signal for demodulation, sampling and upload to the PC. This process repeats automatically and the DPD coefficients are updated each iteration until a desired linearization target (e.g., maximum number of iterations or specified NMSE threshold) is reached. Accounting for settling time, instrument setup, and data transfer, each iteration lasts approximately 1 s.

At the PC, the DVR-based model in (1) is used to generate the predistorted signal. The model is set up with $S = 8$ and $M = 3$. The sampling frequency for the signal generator and spectrum analyzer is 184.32 MHz. A block of 15 000 signal samples are loaded into the signal generator and captured at the signal analyzer each iteration. To recreate a realistic training scenario the input signal ($\mathbf{x}$ in Fig. 7) is varied for each new SPSA iteration by selecting a different set of 15 000 contiguous samples from a large stored data set.

A. 20-MHz Single-Carrier LTE Signal

The performance was first tested using a 20-MHz single-carrier LTE signal with 6.5 dB peak to average power ratio. The carrier frequency was 1.84 GHz and the average input signal power to the PA was 20.5 dBm. Fig. 11 reports convergence performance in terms of NMSE measured between the system input and output signals. As mentioned above, the input signal is varied on each iteration to recreate a realistic training scenario, this causes the measured NMSE to fluctuate around a decreasing mean value as the algorithm converges.

TABLE VI
20-MHz LTE Measurement Results

Scenario	Iterations	NMSE (dB)	ACPR (dB) +/-20MHz
No DPD	-	-26.46	-34.0/-34.4
Proposed DPD	5,000	-35.7	-46.2/-45.1
	10,000	-35.9	-47.7/-48.3
	15,000	-36.1	-47.9/-48.4
	20,000	-36.2	-47.3/-48.3
Least Squares	5	-36.3	-48.9/-49.3

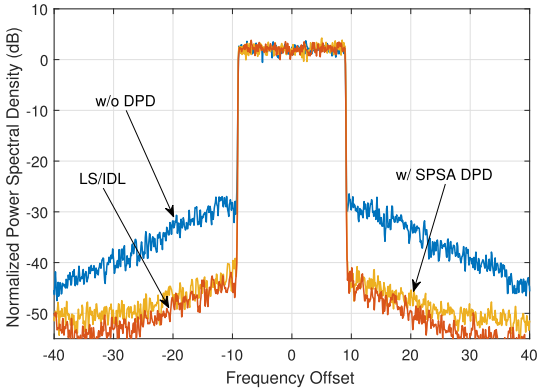


Fig. 12. Measured PA output spectra for a 20-MHz LTE signal.

A 500-point moving average for the measured NMSE is included in Fig. 11 to illustrate the overall convergence trend. The reference NMSE measurement taken when the predistortion coefficients are extracted using a standard LS/IDL method is also reported. The convergence curve follows a familiar pattern for SPSA with an initial fast training period that rapidly slows down as the coefficients approach their optimum values. Table VI compares the performance in terms of time domain NMSE and frequency domain adjacent channel power ratio (ACPR) with the conventional LS/IDL method. After 20 000 iterations, the NMSE performance of the proposed method is within 0.1 dB of the conventional LS/IDL reference case, and the difference between the two techniques in terms of ACPR is less than 2 dB.

Fig. 12 shows the measured PA output spectra with and without DPD applied. After 20 000 iterations, the linearization performance is close to that of the conventional LS/IDL approach. Fig. 13 shows the AM/AM and AM/PM characteristics before and after linearization using the SPSA-calculated coefficients. Compared with LS, the proposed approach requires a large number of iterations, but as discussed earlier, the computational complexity at each iteration is very low and, thus, the overall operation can be very fast. In a practical implementation, the training time depends on the number of the sampling points used and the sampling rate of the signal at the DPD output. Assuming a sampling rate of 400 MS/s, and capturing 8192 samples per iteration, a 20 000 iteration training run would take approximately 400 ms to complete.

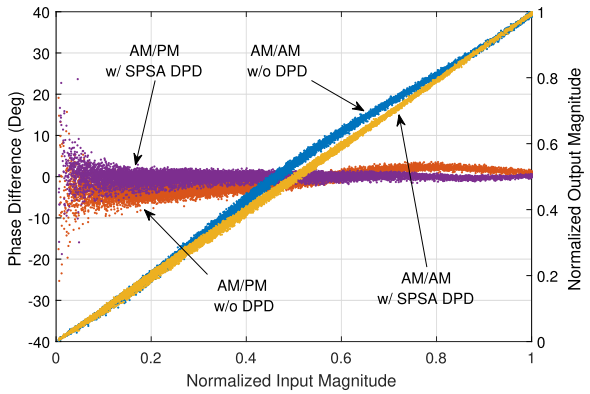


Fig. 13. AM/AM and AM/PM plots for a 20-MHz LTE signal with and without DPD.

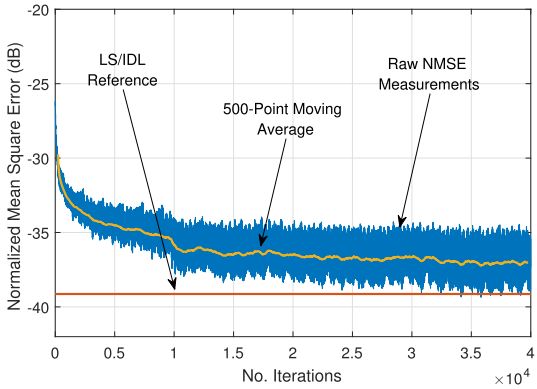


Fig. 14. Measured SPSA convergence for a 40-MHz LTE signal.

TABLE VII
40-MHz LTE MEASUREMENT RESULTS

Scenario	Iterations	NMSE (dB)	ACPR (dB) +/-30 MHz
No DPD	-	-26.4	-31.1/-32.0
Proposed DPD	40,000	-37.3	-46.0/-45.8
Least Squares	5	-39.1	-48.1/-48.0

B. 40-MHz Dual-Carrier LTE Signal

The proposed technique was also evaluated using a 40-MHz dual-carrier LTE signal with 9.5 dB peak to average power ratio. The carrier frequency was again 1.84 GHz and the average signal power at the PA input was 20.5 dBm. Fig. 14 shows the algorithm convergence over the course of 40 000 iterations. The increased number of iterations illustrates that, although the convergence speed slows as the iteration number increases, the NMSE continues to improve. With 10 000 iterations, the SPSA test case NMSE is approximately 4 dB worse than the LS/IDL reference, however, after 40 000 iterations, the gap between the two is reduced to 1.8 dB. Table VII reports the NMSE and ACPR measurements for the test scenario. In terms of ACPR, after 40 000 iterations, the measured values for the SPSA DPD are within 2 dB of the LS/IDL reference. Fig. 15 shows the linearized output spectrum where the reduction

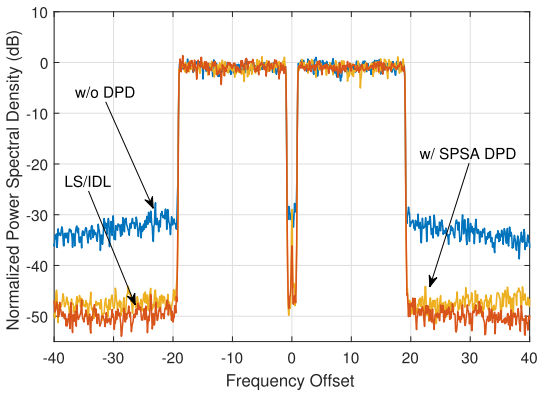


Fig. 15. Measured PA output spectra for a 40-MHz LTE signal.

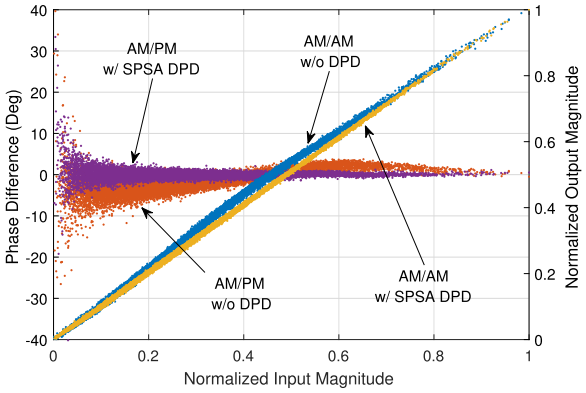


Fig. 16. AM/AM and AM/PM plots for a 40-MHz LTE signal with and without DPD.

in out-of-band spectral regrowth is comparable between the SPSA and LS/IDL techniques. Finally, the AM/AM and AM/PM curves in Fig. 16 show successful linearization using the SPSA-calculated DPD coefficients.

VI. CONCLUSION

A low-complexity DPD model extraction technique has been presented. The proposed solution integrates the SPSA algorithm into the direct learning architecture and uses a modified iteration technique for extracting DPD coefficients. Measurement results indicated that the proposed technique can achieve comparable linearization performance to the existing LS-based solutions but with considerably lower implementation cost.

Because the algorithm is based on stochastic search, multiple iterations would be required to find the final optimum solution. One may argue that the total computational complexity of the proposed approach, i.e., computation per iteration $\times$ number of iterations, may be comparable with or even higher than what LS requires, since LS can converge within a very few iterations while SPSA requires tens of thousands iterations. When making this comparison, a few points should be considered. First of all, it is worth mentioning that a large number of iterations for SPSA training are only required at the system startup. When the DPD system is running in real time, a much smaller number of iterations are typically

required to keep the performance in the acceptable range. The complexity of real time maintenance is therefore very low. This is in contrast to LS, where the full operation must be conducted at each update cycle. If, therefore, considering a life-time operation, the computational complexity and power consumption of the SPSA approach would be much lower compared with the LS approach.

Second, although computational operation is an important concern, there are many other factors that need to be taken into account when implementing a DPD system, including component count, silicon area, and overall system cost, as highlighted in [12]. The LS algorithm offers high accuracy and fast convergence but comes with a high implementation cost in terms of hardware resources as it requires complex matrix multiplication and inversion operations. These operations require special DSP circuits, e.g., dedicated microprocessors, to implement, which can occupy a large silicon area and be costly. High implementation cost is particularly undesirable in 5G small-cell base stations since these small cell stations operate at much lower power levels and the overall cost of the system shall be very low compared with those in conventional larger cells. The cost of each component in the transmitter chain must be carefully managed. Including a complex LS engine in the transmitter for DPD model extraction would not be a favorable solution. Furthermore, in the future wireless systems, in particular small cells, the system will become much more integrated. Including a large silicon area in the transceiver would not be desirable. By employing the iterative approaches, e.g., the proposed SPSA, the required silicon area and cost are fractional compared with that required by LS, which makes them far more attractive.

In addition, we shall point out that the proposed SPSA-based technique in this paper is substantially different from the conventional approaches. In the existing algorithms, such as RLS, the computational complexity is heavily dependent on the number of coefficients and the model structure used. In the proposed approach, all the model coefficients are perturbed at the same time and they are extracted based solely on measurements of the loss function instead of gradient calculation. The model extraction is thus independent of the number of coefficients and it does not require knowledge of the model structure or nonlinear term construction. It makes model extraction much more flexible and the new coefficient set can be generated with very few operations. Currently, the loss function measurement, i.e., RSS calculation, consumes the majority of power. It is envisioned that, in the future systems, the loss function may be implemented in a highly power efficient manner in the analog domain that can enable the total power consumption to be further reduced. This is a unique ability offered by the proposed approach.

In conclusion, although the total number of operations may be comparable with the conventional LS when all iterations are considered, the SPSA-based approach has many unique advantages. These advantages make the proposed technique as an attractive solution for DPD systems in the future 5G small-cell networks, where energy efficiency and cost-effective implementation are expected to become critical requirements.

REFERENCES

[1] J. G. Wood, *Behavioral Modeling and Linearization of RF Power Amplifiers*. Norwood, MA, USA: Artech House, 2014.

[2] R. N. Braithwaite, "General principles and design overview of digital predistortion," in *Digital Front-End in Wireless Communications and Broadcasting*, F.-L. Lou, Ed. Cambridge, U.K.: Cambridge Univ. Press, 2011, ch. 6, pp. 143–191.

[3] F. M. Ghannouchi and O. Hammi, "Behavioral modeling and predistortion," *IEEE Microw. Mag.*, vol. 10, no. 7, pp. 52–64, Dec. 2009.

[4] 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 Process.*, vol. 54, no. 10, pp. 3852–3860, Oct. 2006.

[5] A. Zhu, J. C. Pedro, and T. J. Brazil, "Dynamic deviation reduction-based Volterra behavioral modeling of RF power amplifiers," *IEEE Trans. Microw. Theory Techn.*, vol. 54, no. 12, pp. 4323–4332, Dec. 2006.

[6] A. Zhu, "Decomposed vector rotation-based behavioral modeling for digital predistortion of RF power amplifiers," *IEEE Trans. Microw. Theory Techn.*, vol. 63, no. 2, pp. 737–744, Feb. 2015.

[7] L. Ding *et al.*, "A robust digital baseband predistorter constructed using memory polynomials," *IEEE Trans. Commun.*, vol. 52, no. 1, pp. 159–165, Jan. 2004.

[8] A. Zhu, P. J. Draxler, J. J. Yan, T. J. Brazil, D. F. Kimball, and P. M. Asbeck, "Open-loop digital predistorter for RF power amplifiers using dynamic deviation reduction-based Volterra series," *IEEE Trans. Microw. Theory Techn.*, vol. 56, no. 7, pp. 1524–1534, Jul. 2008.

[9] 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.

[10] L. Guan and A. Zhu, "Optimized low-complexity implementation of least squares based model extraction for digital predistortion of RF power amplifiers," *IEEE Trans. Microw. Theory Techn.*, vol. 60, no. 3, pp. 594–603, Mar. 2012.

[11] R. Q. Hu and Y. Qian, "An energy efficient and spectrum efficient wireless heterogeneous network framework for 5G systems," *IEEE Commun. Mag.*, vol. 52, no. 5, pp. 94–101, May 2014.

[12] J. Wood, "System-level design considerations for digital pre-distortion of wireless base station transmitters," *IEEE Trans. Microw. Theory Techn.*, vol. 65, no. 5, pp. 1880–1890, May 2017.

[13] N. Zheng, Y. Chen, X. Wu, and J. Shi, "Digital predistortion based on QRD-RLS algorithm and its implementation using FPGA," in *Proc. 1st Int. Conf. Inf. Sci. Eng.*, Dec. 2009, pp. 200–203.

[14] Y. Li and X. Zhang, "Adaptive digital predistortion based on MC-FQRD-RLS algorithm using indirect learning architecture," in *Proc. 2nd Int. Conf. Adv. Comput. Control*, vol. 4. Mar. 2010, pp. 240–242.

[15] G. Montoro, P. L. Gilabert, E. Bertran, A. Cesari, and J. A. Garcia, "An LMS-based adaptive predistorter for cancelling nonlinear memory effects in RF power amplifiers," in *Proc. Asia–Pacific Microw. Conf.*, Dec. 2007, pp. 1–4.

[16] P. L. Gilabert, E. Bertran, G. Montoro, and J. Berenguer, "FPGA implementation of an LMS-based real-time adaptive predistorter for power amplifiers," in *Proc. Joint IEEE North-East Workshop Circuits Syst. TAISA Conf.*, Jun. 2009, pp. 1–4.

[17] F. M. Ghannouchi, O. Hammi, and M. Helaoui, *Behavioral Modeling and Predistortion of Wideband Wireless Transmitters*, 1st ed. West Sussex, U.K.: Wiley, 2015.

[18] N. Kelly and A. Zhu, "Low-complexity stochastic optimization-based model extraction for digital predistortion of RF power amplifiers," *IEEE Trans. Microw. Theory Techn.*, vol. 64, no. 5, pp. 1373–1382, May 2016.

[19] J. C. Spall, "Multivariate stochastic approximation using a simultaneous perturbation gradient approximation," *IEEE Trans. Autom. Control*, vol. 37, no. 3, pp. 332–341, Mar. 1992.

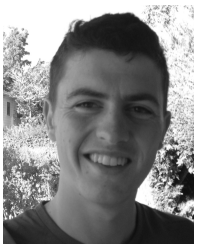
[20] J. C. Spall, "An overview of the simultaneous perturbation method for efficient optimization," *John Hopkins APL Tech. Dig.*, vol. 19, no. 4, pp. 482–492, 1998.

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

[22] L. Guan and A. Zhu, "Dual-loop model extraction for digital predistortion of wideband RF power amplifiers," *IEEE Microw. Compon. Lett.*, vol. 21, no. 9, pp. 501–503, Sep. 2011.

[23] R. N. Braithwaite, "Closed-loop digital predistortion (DPD) using an observation path with limited bandwidth," *IEEE Trans. Microw. Theory Techn.*, vol. 63, no. 2, pp. 726–736, Feb. 2015.

[24] *7 Series FPGAs Overview*, Xilinx Inc., San Jose, CA, USA, 2015.



Noel Kelly (S'15) received the B.E. and Ph.D. degrees in electronic engineering from the School of Electrical and Electronic Engineering, University College Dublin, Dublin, Ireland, in 2012 and 2017, respectively.

His current research interests include low-complexity digital predistortion architectures, efficient field-programmable gate array implementation solutions, and digital predistortion applications for satellite communications.

Anding Zhu (S'00–M'04–SM'12) received the B.E. degree in telecommunication engineering from North China Electric Power University, Baoding, China, in 1997, the M.E. degree in computer applications from the Beijing University of Posts and Telecommunications, Beijing, China, in 2000, and the Ph.D. degree in electronic engineering from the University College Dublin (UCD), Dublin, Ireland, in 2004.

He is currently a Professor with the School of Electrical and Electronic Engineering, UCD. His current research interests include high-frequency nonlinear system modeling and device characterization techniques with a particular emphasis on behavioral modeling and linearization of RF power amplifiers for wireless communications, high-efficiency power amplifier design, wireless transmitter architectures, digital signal processing, and nonlinear system identification algorithms.