Low Complexity Stochastic Optimization-Based Model Extraction for Digital Predistortion of RF Power Amplifiers

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

***Abstract*— This paper introduces a low-complexity stochastic optimization-based model coefficients extraction solution for digital predistortion of RF power amplifiers. The proposed approach uses a closed-loop extraction architecture and replaces conventional least squares training with a modified version of the simultaneous perturbation stochastic approximation (SPSA) algorithm that requires a very low number of numerical operations per iteration, leading to considerable reduction in hardware implementation complexity. Experimental results show that the complete closed-loop stochastic optimization-based coefficient extraction solution achieves excellent linearization accuracy while avoiding the complex matrix operations associated with conventional least squares techniques.**

***Index Terms*—Digital predistortion, linearization, model extraction, stochastic optimization, simultaneous perturbation stochastic approximation, power amplifier.**

I. Introduction

IN MODERN wireless base stations, the radio frequency (RF) power amplifier (PA) is an inherently nonlinear device that causes in-band distortion as well as out-of-band spectral growth in the transmitted signal. These effects are particularly severe at high output power levels when the PA is operated in a power efficient mode. Digital predistortion (DPD) is an advanced linearization technique that compensates for PA nonlinear effects by applying an inverted model of the PA to the input signal at digital baseband before amplification [1],[2].

To effectively apply DPD, an accurate PA model must be developed first since it is only when the nonlinear characteristics of the PA are accurately modeled and thus correctly reversed, that the overall system response to a signal flowing serially through the cascade of DPD-PA can become linear. In recent years, a range of advanced behavioral models for RF PAs have been developed, with many derived from the Volterra series [3]-[5]. The coefficients for these models are typically calculated by using least squares (LS) based algorithms in an indirect learning (IDL) architecture [6]-[8]. The LS algorithm offers high accuracy and fast convergence, but it uses complex matrix multiplications and inversions which require substantial hardware resources to execute. Furthermore, the complexity of these matrix operations increases with the number of coefficients employed in the DPD model and the number of samples used in the extraction process [9]. As operating bandwidths in wireless communication systems continue to increase, the nonlinear behavior of the PAs becomes more complicated. It leads that more coefficients will be used in the DPD models which in turn causes the LS operation to become more complex and power consuming. In addition, in future small-cell base stations, cost and energy consumption of the digital part itself is expected to become a major consideration because the power savings in the RF become smaller. Thus, low-complexity coefficient extraction solutions for DPD are highly desirable.

In [10], a stochastic optimization-based DPD coefficient calculation technique was proposed as a low-complexity alternative to the LS solution. It was derived from the simultaneous perturbation stochastic approximation (SPSA) algorithm that uses the measurements of the loss function with a random perturbation on the model coefficients to determine the coefficient updating direction and finally find the optimum solution. By using this approach, the gradient approximation only requires two function measurements per iteration regardless of the number of coefficients involved. The coefficient perturbation process only requires a simple addition and subtraction operation, which leads to substantial savings in hardware resource usage in the model extraction.

Due to limited space, only the basic concept was given and the SPSA was only applied in the indirect learning structure in [10]. In this paper, we first present the complete idea of the SPSA algorithm and then give a step-by-step guide to implementation of an enhanced version of the SPSA algorithm that is specifically suitable for the DPD coefficient extraction. Using information from previous iterations, the proposed modification substantially improves convergence speed. To effectively reuse hardware resources and optimize system performance, a complete SPSA-based model extraction approach using the closed-loop coefficient estimation architecture is also given. Experimental results show that the proposed approach can achieve comparable linearization

This publication was emanated from research supported in part by research grants from Science Foundation Ireland (SFI) and co-funded under the European Regional Development Fund under Grant Numbers 13/RC/2077 and 12/IA/1267. This paper is an expanded version from the IEEE MTT-S International Workshop on Integrated Nonlinear Microwave and Millimetre-wave Circuits (INMMiC), Taormina, Italy, October 1–2, 2015.

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

performance but use considerably less hardware resources compared to the conventional LS algorithm.

The paper is organized as follows. In Section II, the conventional DPD model extraction is briefly reviewed and the associated challenges are highlighted. Stochastic optimization using the SPSA algorithm is introduced in Section III and the proposed novel application-specific solution is given in Section IV. Section V discusses practical application of the technique in the closed-loop coefficient estimation architecture while Section VI reports simulation and experimental results. The overall findings of the work are summarized in Section VII.

II. Conventional DPD Model Extraction

Much effort has been devoted in recent years to developing efficient DPD behavioral models, typically with the goal of reducing the number of terms while maintaining model accuracy [3]-[5]. One recent example, the Decomposed Vector Rotation (DVR) model [11], relates the PA input signal, $\tilde{x}(n)$, and its output, $\tilde{y}(n)$ as shown in (1), where a_i and $c_{s,i}$ are the model coefficients.

$$\begin{aligned}\tilde{y}(n) = &\sum_{i=0}^{M} a_i \tilde{x}(n-i) \\ &+\sum_{s=1}^{S}\sum_{i=0}^{M} c_{s,i,1}\left|\left|\tilde{x}(n-i)\right|-\beta_s\right| e^{j\theta(n-i)} \\ &+\sum_{s=1}^{S}\sum_{i=0}^{M} c_{s,i,21}\left|\left|\tilde{x}(n-i)\right|-\beta_s\right| e^{j\theta(n-i)}\left|\tilde{x}(n)\right| \\ &+\ldots\end{aligned} \tag{1}$$

For calculation of the DPD model coefficient vector, $\hat{\boldsymbol{C}}=[a_1, a_2, \ldots, c_{s,l,l}, \ldots]$, the indirect learning architecture is commonly used [2],[6]-[8]. In this architecture, the model extraction is conducted by using a post-inverse model, where the output of the PA, $\tilde{y}(n)$, is used as input of the model while the input of the PA, $\tilde{u}(n)$, is used as the expected output. Because the post-inverse model has the identical structure as that used for the pre-inverse DPD model, the extracted coefficients for the post-inverse model can be directly copied to the DPD block, as illustrated in Fig. 1.

To extract the post-inverse model coefficients, the standard least squares algorithm can be employed and the coefficient vector, $\hat{\mathbf{C}}$, is given by

$$\hat{\mathbf{C}} = \left(\mathbf{Y}^H\mathbf{Y}\right)^{-1}\mathbf{Y}^H\mathbf{U} \tag{2}$$

where $(\cdot)^H$ represents the Hermitian transpose and $(\cdot)^{-1}$ is the matrix inverse operator. The vector $\mathbf{U}$ is formed from the DPD output signal samples $\tilde{u}(n)$ and $\mathbf{Y}$ is a matrix constructed using measured samples of the PA output, $\tilde{y}(n)$, in which each column corresponds to a term in the DVR behavioral model in (1). The large matrix inversion and multiplication operations required to execute (2) are computationally complex and resource intensive. For instance, to extract 50 coefficients with 8000 digital samples, it requires over 40,000,000 complex multiplication operations. Implementing such algorithms in digital hardware requires substantial dedicated hardware resources and occupies large chip area.

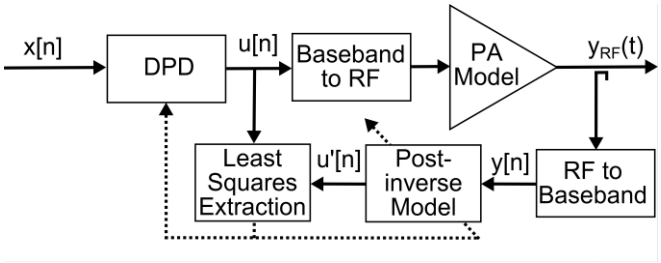

Fig. 1. DPD system with indirect learning model extraction architecture.

To reduce the computational complexity, iterative coefficient extraction techniques can be considered. In particular, the recursive least-squares (RLS) and least-mean-squares (LMS) algorithms have been applied in DPD and PA modeling [12],[13]. The RLS algorithm is an example of a quasi-Newton optimization method in which the coefficient update equation uses an approximation to the Hessian matrix at each iteration. Although the RLS algorithm avoids operations involving large matrices, maintaining an accurate approximation of the Hessian matrix still requires significant complexity, particularly for higher order models. Alternatively, gradient descent-based methods, such as LMS, rely on a simple first order approximation to determine the update direction. In the LMS algorithm, a highly efficient approximation to the loss function gradient vector is used where large matrix calculations are avoided and implementation complexity is greatly reduced. However, LMS is very slow to converge and the algorithm typically struggles to achieve the desired model accuracy.

III. Simultaneous Perturbation Stochastic Approximation Algorithm

SPSA is an efficient stochastic optimization algorithm that follows an iterative procedure where incremental updating of adjustable parameters is used to converge towards the desired minimum or maximum of an objective function. For a given iteration, k, the SPSA algorithm calculates an updated coefficient vector according to:

$$\hat{\mathbf{C}}_{k+1} = \hat{\mathbf{C}}_k - a_k \hat{g}_k\left(\hat{\mathbf{C}}_k\right) \tag{3}$$

where $\hat{\mathbf{C}}_k$ is the existing coefficient vector and a_k is a weighting factor used to control convergence speed. The term $\hat{g}_k(\hat{\mathbf{C}}_k)$ represents the loss function gradient at the coefficient vector $\hat{\mathbf{C}}_k$ and is responsible for determining the direction of the algorithm update. The key idea of SPSA is that, the loss function gradient, $\hat{g}_k(\hat{\mathbf{C}}_k)$, is estimated by using measurements on the loss function instead of conducting direct differential calculation, which substantially reduces the computational complexity [14][15].

Let's take a simple PA forward model, shown in Fig. 2, as an example and assume that the model is constructed by using a nonlinear function with a set of coefficients. The goal is to find the optimum coefficient values that result in the closest match between the predicted output y′[n] and the measured output y[n] with input x[n]. To extract the coefficients, a random perturbation sequence, Δ_k is added and subtracted

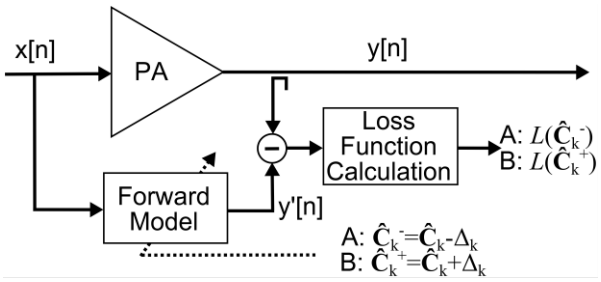

Fig. 2. Loss function measurement in the SPSA algorithm.

(with weighting c_k) from the current forward model coefficient vector, $\hat{\mathbf{C}}_k$, generating two additional coefficient vectors,

$$\begin{aligned} \hat{\mathbf{C}}_k^+ &= \hat{\mathbf{C}}_k + c_k \boldsymbol{\Delta}_k \\ \hat{\mathbf{C}}_k^- &= \hat{\mathbf{C}}_k - c_k \boldsymbol{\Delta}_k \end{aligned} . \tag{4}$$

By applying the two coefficient vectors to the model, two sets of model output data can be obtained and two loss function measurements, $L(\hat{\mathbf{C}}_k^+)$ and $L(\hat{\mathbf{C}}_k^-)$ are performed. The loss function gradient is then simply approximated by

$$\hat{g}_k(\hat{\mathbf{C}}_k) = \frac{L(\hat{\mathbf{C}}_k^+) - L(\hat{\mathbf{C}}_k^-)}{\hat{\mathbf{C}}_k^+ - \hat{\mathbf{C}}_k^-} . \tag{5}$$

and the resulting coefficient update equation is given by:

$$\hat{\mathbf{C}}_{k+1} = \hat{\mathbf{C}}_k - a_k \frac{L(\hat{\mathbf{C}}_k^+) - L(\hat{\mathbf{C}}_k^-)}{\hat{\mathbf{C}}_k^+ - \hat{\mathbf{C}}_k^-} . \tag{6}$$

Because all elements of $\hat{\mathbf{C}}_k$ are randomly perturbed together, the gradient approximation measurements are independent from the number of the coefficients, which significantly simplifies the per-iteration complexity. The perturbation vector itself is randomly generated at each iteration, given by:

$$\boldsymbol{\Delta}_k = \begin{bmatrix} \Delta_{k1} \\ \Delta_{k2} \\ \vdots \\ \Delta_{kW} \end{bmatrix} \tag{7}$$

and a Bernoulli distribution in which +1 and -1 outcomes occur with equal probability is considered a suitable choice to satisfy convergence requirements [15].

Based on the description above, a single iteration of the SPSA algorithm can then be summarized as follows:

Step 1: Generate perturbation vector Δ_k
Step 2: Calculate temporary coefficients $\hat{\mathbf{C}}_k^+$ and $\hat{\mathbf{C}}_k^-$
Step 3: Measure error function values $L(\hat{\mathbf{C}}_k^+)$ and $L(\hat{\mathbf{C}}_k^-)$
Step 4: Obtain gradient approximation $\hat{g}_k(\hat{\mathbf{C}}_k)$
Step 5: Update coefficient vector $\hat{\mathbf{C}}_k$

Using the update equation in (6), SPSA iteratively calculates an optimum coefficient vector for a given system with a very low number of operations per iteration. Specifically, as shown above, at each iteration the algorithm requires only a small number of simple addition and subtraction operations, as defined in (4) and (5), and generation of the perturbation vector in (7). The progression of the SPSA algorithm over 4 iterations across a sample error surface is depicted in Fig. 3.

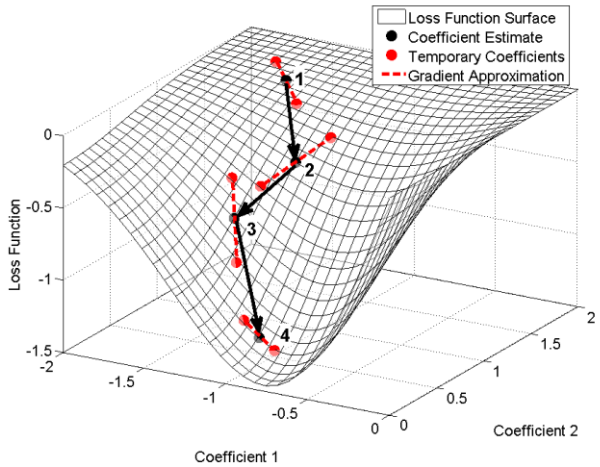

Fig. 3. Evolution of the SPSA algorithm over four iterations.

IV. Modified SPSA

In the general form outlined above, SPSA can be applied across a wide range of optimization scenarios. However, in this work only a single application of the algorithm is considered, namely, calculation of a set of DPD coefficients. This restricted operating environment can be exploited to develop the original algorithm into a higher performing application-specific technique, as discussed below.

A. *Quadratic Interpolation*

In conventional DPD, the normalized mean squared error (NMSE) is often used as a metric to quantify the linearization performance [16]. The NMSE measures the total power of the error vector between the ideal and modeled waveforms, normalized to the ideal signal power. It is defined as:

$$NMSE = \frac{\sum_{n=1}^{N} |e(n)|^2}{\sum_{n=1}^{N} |u_{ideal}(n)|^2} \tag{8}$$

where N is the total number of samples, and $e(n)$ is the difference between the ideal and measured signals

$$e(n) = u_{meas}(n) - u_{ideal}(n) \tag{9}$$

Owing to the requirements of conventional extraction techniques, the vast majority of modern DPD behavioral models are designed to ensure a linear relationship between the model output and its coefficients. For these linear-in-parameters models, it follows that, for each sample, the error measured at the model output has a linear relationship with the model coefficients. This in turn leads that, if we choose NMSE as the loss function, the output of the loss function can be expressed as a quadratic function of the model coefficients [13]. This special feature of the loss function can be used to improve SPSA performance in extracting the DPD coefficients.

In the standard implementation, the loss function gradient is approximated as the slope of a line between two points located in the vicinity of the coefficient estimate, as shown in Fig.3. When the loss function is known to have a quadratic form, however, the interpolated function is no longer an approximation but an accurate representation of the loss

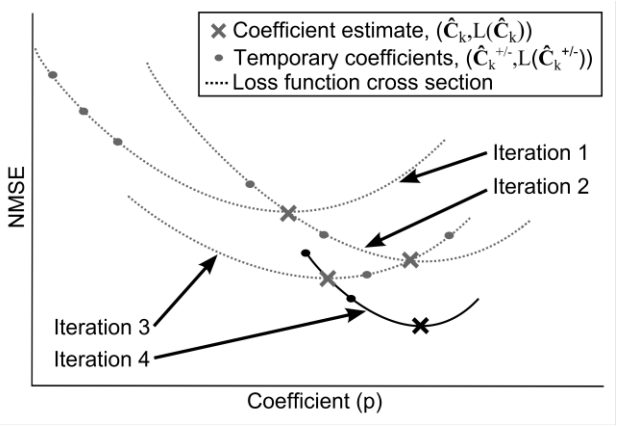

Fig. 4. Evolution of the quadratic interpolation SPSA algorithm over 4 iterations.

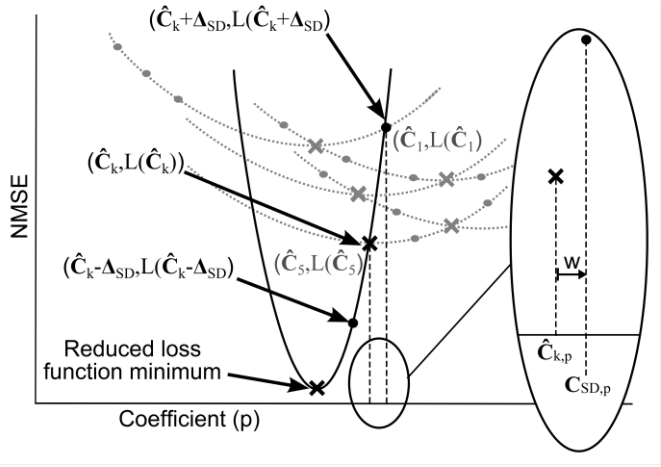

Fig. 5. Steep-descent perturbation vector generation.

function along the chosen perturbation vector. As a result, the updated coefficient estimate can be found by moving directly to the minimum of the interpolated function:

$$\hat{\mathbf{C}}_{k+1} = \hat{\mathbf{C}}_k - \frac{\mathbf{\Delta}_k (L(\hat{\mathbf{C}}_k^+) - L(\hat{\mathbf{C}}_k^-))}{L(\hat{\mathbf{C}}_k^-) + L(\hat{\mathbf{C}}_k^+) - 2L(\hat{\mathbf{C}}_k)} \tag{10}$$

For quadratic modeling of the loss function, three unique points are required. In addition to the two temporary coefficient vectors, $\hat{\mathbf{C}}_k^+$ and $\hat{\mathbf{C}}_k^-$ used in conventional SPSA, a third measurement is taken by directly using the current coefficient, $\hat{\mathbf{C}}_k$, to give $L(\hat{\mathbf{C}}_k)$. Fig. 4 depicts the evolution of this process over a number of iterations for a single coefficient. Successive quadratic interpolations can be made and the coefficient set can be accurately updated to reduce the loss function value at each iteration. Using quadratic interpolation to improve SPSA performance was mentioned in [17] and the Appendix at the end of this paper provides further detail on the derivation of (10).

B. Steep-Descent SPSA Algorithm

Although results show that the performance in quadratic interpolation is consistent, the standard Bernoulli perturbations typically generate shallow quadratic functions that provide only a relatively small improvement at each iteration. This leads to long convergence time in model extraction. To improve convergence speed for SPSA, in this work, we propose to supplement the standard Bernoulli-based perturbations with calculated steep perturbation sequences designed to generate more efficient slices of the loss function surface.

As shown in Fig. 4, as the SPSA algorithm progresses, unique coefficient sets and their related loss function points are generated at each iteration. Because the model is linear-in-parameters, the loss function surface is thus quadratic, which leads that not only the coefficient sets associated with each iteration but also any combination of two different coefficient sets shall fall on a quadratic curve. For instance, as shown in Fig. 5, a coefficient set from iteration 1 and another set from iteration 5 can be used to form a new quadratic curve. This unique feature allows us to "intelligently" use the existing information from previous iterations to choose an efficient search direction.

To explain how this works, consider the example taken above of moving from a point on the quadratic on iteration 1 to the one on iteration 5. After the first iteration we select a point and label it with coordinates ($\hat{\mathbf{C}}_1$, $L(\hat{\mathbf{C}}_1)$) and similarly after the fifth iteration, we select a second point ($\hat{\mathbf{C}}_5$, $L(\hat{\mathbf{C}}_5)$). If we subtract $\hat{\mathbf{C}}_1$ from $\hat{\mathbf{C}}_5$, we find the perturbation vector , $\mathbf{\Delta}_{SD}$ required to move between the points. Now, we define a third coefficient set $\hat{\mathbf{C}}_5 + \mathbf{\Delta}_{SD}$ and obtain the associated loss function point $L(\hat{\mathbf{C}}_5+\mathbf{\Delta}_{SD})$. The three points ($\hat{\mathbf{C}}_1$, $L(\hat{\mathbf{C}}_1)$), ($\hat{\mathbf{C}}_5$, $L(\hat{\mathbf{C}}_5)$), and ($\hat{\mathbf{C}}_5+\mathbf{\Delta}_{SD}$, $L(\hat{\mathbf{C}}_5+\mathbf{\Delta}_{SD})$) can now be used to form a quadratic curve. Using this curve a new minimum point and corresponding coefficient set can be found. Different from the normal Bernoulli-based quadratic searching where the perturbation weighting is constant across the coefficient set, here the weighting varies. If two coefficient sets are located a short distance apart along the horizontal axis but are separated by a large vertical distance, a steep quadratic curve can be formed and thus the minimum point can be found much faster. We call this approach steep-descent SPSA (SD-SPSA).

For the technique to work, the main challenge is in finding a suitable perturbation vector $\mathbf{\Delta}_{SD}$. To further explain, as shown in Fig. 5, at a given iteration k, to calculate $\mathbf{\Delta}_{SD}$, the SD-SPSA algorithm selects a point, denoted ($\hat{\mathbf{C}}_{SD}$, $L(\hat{\mathbf{C}}_{SD})$), on the q-th past loss function, $L_{k-q}(\hat{\mathbf{C}})$, such that there exists a steep slope between it and the current coefficient vector/loss function point, ($\hat{\mathbf{C}}_k$, $L(\hat{\mathbf{C}}_k)$). The perturbation vector required to move between these two points is calculated and designated the new steep descent perturbation, $\mathbf{\Delta}_{SD}$.

Efficient calculation of the numerical values for $\hat{\mathbf{C}}_{SD}$ is necessary if the SD-SPSA algorithm is to be effective. Using data from a past iteration, k-q, the desired $\hat{\mathbf{C}}_{SD}$ coefficients can be expressed in terms of, $\hat{\mathbf{C}}_{k-q}$ and $\mathbf{\Delta}_{k-q}$ as:

$$\hat{\mathbf{C}}_{SD} = \hat{\mathbf{C}}_{k-q} + \left(c_{step} \times \mathbf{\Delta}_{k-q}\right) \tag{11}$$

where c_{step} is a constant determining the location of $\hat{\mathbf{C}}_{SD}$ on the past loss function curve. The value of c_{step} for a given iteration is constant across all terms in the coefficient vector $\hat{\mathbf{C}}_{SD}$. Thus, (11) can be rearranged to give an expression for c_{step} in terms of a single term in $\hat{\mathbf{C}}_{SD}$:

$$c_{step} = \frac{\hat{\mathbf{C}}_{SD,p} - \hat{\mathbf{C}}_{k-q,p}}{\Delta_{k-q,p}}. \quad (12)$$

As shown in Fig. 5, we define a single term in $\hat{\mathbf{C}}_{SD,p}$ as:

$$\hat{\mathbf{C}}_{SD,p} = \hat{\mathbf{C}}_{k,p} + w \quad (13)$$

where w is chosen as a small value to ensure a steep slope between the two points. The resulting numerical value is used to determine c_{step} in (12) which is in turn applied to (11) to find $\hat{\mathbf{C}}_{SD}$. Finally, the desired steep descent perturbation vector, $\mathbf{\Delta}_{SD}$, is calculated as simply the perturbation required to move between the current coefficient set $\hat{\mathbf{C}}_k$ to the set $\hat{\mathbf{C}}_{SD}$:

$$\mathbf{\Delta}_{SD} = \hat{\mathbf{C}}_{SD} - \hat{\mathbf{C}}_k. \quad (14)$$

Once $\mathbf{\Delta}_{SD}$ is determined, the SD-SPSA algorithm follows an identical procedure to the standard quadratic interpolation SPSA for the remainder of the iteration. Two temporary coefficient vectors are formed,

$$\begin{aligned} \hat{\mathbf{C}}_k^+ &= \hat{\mathbf{C}}_k + c_k \mathbf{\Delta}_{SD} \\ \hat{\mathbf{C}}_k^- &= \hat{\mathbf{C}}_k - c_k \mathbf{\Delta}_{SD} \end{aligned} \quad (15)$$

and with the current coefficient vector $\hat{\mathbf{C}}_k$, three loss function measurements are generated. Quadratic interpolation is performed on the resulting three unique loss function points and the updated coefficient estimate is calculated according to (10).

Calculating the steep descent perturbation sequence relies on storing both the perturbation sequence, $\mathbf{\Delta}_k$, and the coefficient vector, $\hat{\mathbf{C}}_k$, from past iterations. The number of loss functions to be stored can be a user-defined parameter of the algorithm. To prevent unnecessary implementation complexity, it is desirable to limit the memory depth as much as possible without impacting performance. It has been observed in this work that, by storing iteration data points at regular intervals instead of consecutively, overall storage requirements can be drastically reduced without significantly impacting on performance. For example, drawing on data from the past 1000 consecutive iterations, similar performance can be achieved by storing 100 data samples taken periodically over the same 1000 iterations.

V. Proposed DPD Coefficient Extraction

Two architectures are commonly used in DPD coefficient extraction: open-loop indirect learning and closed-loop adaptation. As shown in Fig. 1, the indirect learning puts the DPD block outside the training loop and uses the PA output as the input and the DPD output as the expected output to train a post-inverse model first and then copy the coefficients to the pre-inverse (DPD) block. It can converge very quickly but the final accuracy may be affected by linear impairments of the feedback loop [18]. More importantly, a complete post-distortion model must be constructed in the training process, which can increase the hardware implementation cost of the coefficient extraction. In contrast, outlined in Fig. 6, the closed-loop approach places the DPD model inside the estimation loop and iteratively updates the coefficients using a separate error model [18],[19]. On each training run, h, the

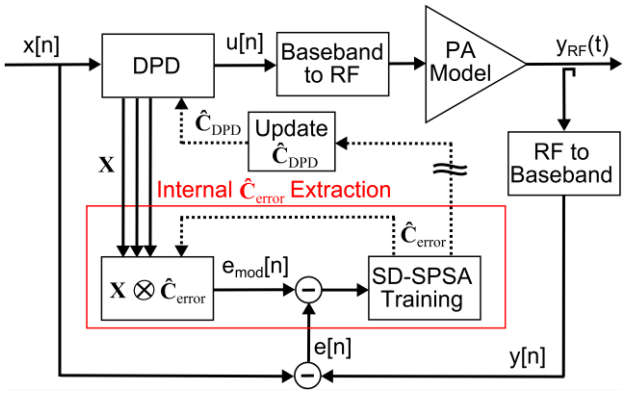

Fig. 6. SD-SPSA in the closed-loop coefficient estimation architecture.

DPD model coefficients vector $\hat{\mathbf{C}}_{DPD}$ is updated according to:

$$\hat{\mathbf{C}}_{DPD,h+1} = \hat{\mathbf{C}}_{DPD,h} - \lambda \hat{\mathbf{C}}_{error} \quad (16)$$

where the error model coefficients vector, $\hat{\mathbf{C}}_{error}$, is trained to model the measured error between the original input $x[n]$ and the PA output $y[n]$,

$$e[n] = x[n] - y[n] \quad (17)$$

and λ is the adaptation factor. The closed-loop converges slower but it can typically achieve greater accuracy when it reaches the steady state. Because the error model uses the same input as the DPD model, calculation of the nonlinear modeling terms can be shared between the two blocks, which can reduce hardware resource usage. In this work, we propose to incorporate the SD-SPSA algorithm developed from Section IV into a closed-loop estimation architecture to demonstrate how the SPSA technique can simplify the DPD model extraction process.

A. SPSA-Based Closed-Loop Coefficient Extraction

Typically, LS estimation is used to calculate the error model coefficients,

$$\hat{\mathbf{C}}_{error} = (\mathbf{X}^H \mathbf{X})^{-1} \mathbf{X}^H \mathbf{E}. \quad (18)$$

where $\mathbf{X}$ is a matrix constructed from the input samples, x[n], in which each column corresponds to a term in the DPD model and $\mathbf{E}$ is the vector of measured errors, e[n]. As mentioned earlier, LS operation involves large matrix operations. In this work, we propose employing the SD-SPSA algorithm to train the error model. Applying SD-SPSA in this architecture, in order to measure the loss function at each iteration, the error model output, $e_{mod}[n]$ in Fig. 6, must be calculated. In other words, we need to run the error model,

$$\mathbf{E}_{mod} = \mathbf{X}\hat{\mathbf{C}}_{error} \quad (19)$$

where $\mathbf{E}_{mod}$ is the vector of error model output samples $e_{mod}[n]$.

Since the error model has identical structure to that used in the DPD block and, at each DPD run, the matrix $\mathbf{X}$ is already generated in the DPD block, it is not necessary to implement a full copy of the DPD model. Instead, in the error model we only need to read out the terms of the matrix $\mathbf{X}$ from the DPD block and multiply with the error coefficients to generate $\mathbf{E}_{mod}$. As shown in Fig. 6, this allows just one set of model terms to be generated in the DPD model and "shared" with the error model where a different coefficient vector is applied. For

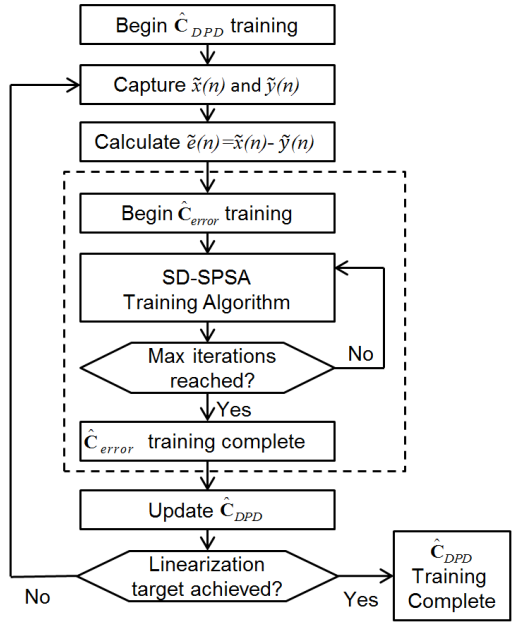

Fig. 7. The closed-loop model extraction procedure.

example, the term $\left||\tilde{x}(n-i)|-\beta_s\right| e^{j\theta(n-i)}$ in the DVR model in (1) can be generated just once and multiplied by a different coefficient in the DPD and error models. Implementation of a complete DPD model structure is highly resource-intensive and represents a significant portion of the overall digital hardware requirements of the DPD system [20], utilizing just one DPD model for both coefficient extraction and predistortion significantly reduces the overall complexity.

The complete closed loop DPD extraction process, as outlined in Fig. 7, is composed of two training loops. On a given closed loop coefficient extraction run, system input and output data is captured and used to generate an error signal. The SD-SPSA algorithm then runs in an internal training loop using the captured data to extract the error model coefficients. The conditions for exiting the internal SD-SPSA training loop can be chosen by the user. In our test, we limit a maximum number of iterations as the exiting criteria. When SD-SPSA training is complete, the calculated error model coefficients are passed out of the internal loop and used to update the DPD coefficient estimate, as in (16). If necessary, after the DPD coefficients are updated, new data is captured and the process is repeated until a predetermined linearization target is achieved.

B. Complexity Analysis

Table I details the SD-SPSA complexity in terms of real multiplications and additions required per iteration. Operations required to generate the error model output at each iteration are also included. Standard LS extraction complexity, as in (18), is included for reference. The results show that the coefficient update complexity in SPSA is greatly simplified. To quantify the improvement, for an example scenario with N=8192 samples and K = 50 model terms, SD-SPSA requires 2,490,518 real multiplications while LS would need 124,483,800 operations, leading to a 98% reduction in computational complexity.

TABLE I
SD-SPSA VS LS COMPLEXITY COMPARISON

Operation	Complexity	
	Real Mults.	Real Additions
SD-SPSA(per iteration)		
Calculate $\hat{\mathbf{C}}^+$ and $\hat{\mathbf{C}}^-$	0	$4\times K$
Measure $L(\hat{\mathbf{C}})$, $L(\hat{\mathbf{C}}^+)$, $L(\hat{\mathbf{C}}^-)$	$4\times N$	$8\times N$
Calculate SD perturbations	K	$4\times K$
Calculate coefficient update	$2\times K$	$2\times K$
Error model calculations:		
Generate $\mathbf{X}\hat{\mathbf{C}}^+$ and $\mathbf{X}\hat{\mathbf{C}}^-$	$6\times N\times K$	$6\times N\times(K-1)+$ $10\times N\times K$
Total:	$4N+3K+$ $6NK$	$8N+10K+$ $6N(K-1)+10NK$
Least Squares (18)		
Matrix multiplications:		
$(K\times N)\times(N\times K)$	$3\times N\times K^2$	$2\times(N-1)\times K^2+$ $5\times N\times K^2$
$(K\times K)\times(K\times N)$	$3\times N\times K^2$	$2\times(K-1)\times K\times N+$ $5\times N\times K^2$
$(K\times N)\times(N\times 1)$	$3\times N\times K$	$2\times(N-1)\times K+$ $5\times N\times K$
Matrix inversion $(K\times K)$	$3\times K^3$	$5\times K^3$
Total:	$3\times(2NK^2+$ $NK+K^3)$	$2\times((N-1)K+$ $(N-1)K^2+$ $(K-1)NK)+5K^3+$ $10NK^2+5NK$

N – number of training samples, K – number of DPD coefficients

It is notable that, although not strictly a part of the algorithm itself, the main complexity of SD-SPSA lands on calculating the error model output and the NMSE. For conducting SD-SPSA, the only additional overhead is to calculate the steep descent perturbations. This amounts to K real multiplications and $4K$ real additions, a minor cost relative to the overall complexity. For completeness, aside from the main steps detailed in Table I, a small number of hardware resources are also required for generation of the random perturbation sequence each iteration.

For the SD-SPSA algorithm, past iteration data must be stored. For each iteration, the saved data set contains the original coefficient vector and the associated perturbation vector. Assuming 32 bit accuracy for each complex value, for a DPD model with K coefficients, this corresponds to $32\times K\times 2$ bits of data per stored iteration. For the results presented in this paper, the SD-SPSA algorithm stores information from 100 past iterations, leading to a total storage requirement of $32\times K\times 2\times 100$ bits of data. A typical DPD model with $K=50$ will thus only require 320 Kb of storage space. Considering the Xilinx Virtex 7 FPGA family has between 28,620 Kb and 67,680 Kb of on-device storage capacity in the form of individual 36Kb block RAM units, this memory requirement is well within the capacity of on-board storage in modern FPGA chips [21].

VI. SIMULATION AND EXPERIMENTAL RESULTS

In this section, simulation results for the SD-SPSA algorithm are first presented before the complete DPD coefficient estimation architecture is evaluated in a full RF test bench.

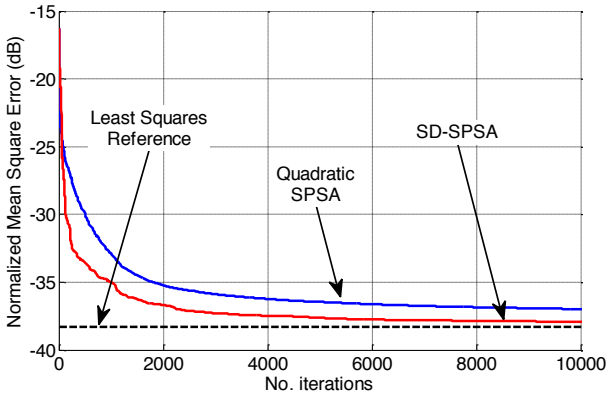

Fig. 8. Quadratic and SD-SPSA NMSE performance.

TABLE II
SD-/QUADRATIC SPSA FORWARD MODELING PERFORMANCE

NMSE Accuracy (dB)	No. of Iterations Required	
	Quadratic SPSA	Proposed SD-SPSA
-30	550	330
-35	1,900	1,050
-36	3,400	1,400
-37	12,500	2,500
-37.5	32,000	4,000
-38	257,000	14,000
-38.2	865,000	30,000

Least Squares NMSE: -38.32 dB.

A. *PA Modeling Simulation*

The performance of DPD model extraction solutions heavily relies on the ability of the algorithms to achieve the desired modeling accuracy. To confirm the accuracy of the proposed SD-SPSA algorithm, a forward PA modeling scenario is considered in this subsection.

The training data consists of 14,500 input/output samples captured from an LDMOS Doherty PA operated at 37 dBm and excited by a 20 MHz W-CDMA signal. A DVR model, as given in (1), with parameters S=8 and M=3 serves as the behavioral model. Both the quadratic SPSA and the SD-SPSA algorithms were tested. The SD-SPSA algorithm was operated with a 2:1 ratio of Bernoulli to steep descent perturbations and 100 past data points were stored with a sampling interval of every 10 iterations. The LS estimation, outlined in (2), was also taken as the reference.

Fig. 8 reports the NMSE performance over iterations. Both the quadratic SPSA and the SD-SPSA follow a similar training pattern in which convergence speed is high for an initial period of approximately 2,000 iterations before slowing down drastically as the NMSE approaches that of the LS solution. For the quadratic SPSA, this plateau effect occurs earlier than in the SD-SPSA solution. After 10,000 iterations, the SD-SPSA algorithm converges to a value close to the LS reference. Table II reports the number of iterations required to reach a chosen accuracy level for each algorithm and again, the fast converging SD-SPSA algorithm is shown to outperform standard quadratic SPSA. In addition, in less than 5,000 iterations the SD-SPSA algorithm achieves an NMSE level within 1 dB of the LS reference and after 30,000 iterations the difference between the two is approximately 0.1 dB. Thus it is shown that using the SD-SPSA algorithm, it is indeed possible to achieve accuracy levels comparable to those obtained with conventional least squares techniques.

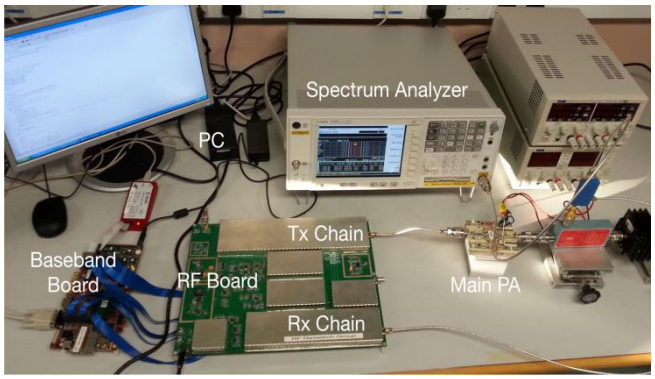

Fig. 9. DPD test platform setup.

TABLE III
SD-SPSA DPD PERFORMANCE FOR 20 MHZ LTE SIGNAL

Scenario	NMSE (dB)	ACPR (dB)	
		-20 MHz	+20 MHz
No DPD	**-20.9**	**-32.7**	**-31.0**
SD-SPSA (2 runs)	-33.0	-45.6	-48.3
SD-SPSA (4 runs)	-39.2	-55.7	-57.9
SD-SPSA (6 runs)	-43.6	-58.7	-58.0
SD-SPSA (8 runs)	-46.9	-59.5	-58.4
SD-SPSA (10 runs)	**-45.7**	**-59.2**	**-59.3**
Least Squares (10 runs)	**-45.6**	**-59.4**	**-59.6**

B. *DPD Experimental Test*

The proposed model extraction solution was then evaluated in a full RF test bench. The test setup was the same as that used in [22], shown in Fig. 9. An LDMOS Doherty PA again was operated at 2.14 GHz. The input signal was generated in MATLAB running on a PC before it was passed to the RF board for modulation and up-conversion and finally sent to the PA. At the PA output, the signal with a pre-determined block length was down-converted and demodulated to baseband, and then captured and returned to the PC for coefficient extraction. Following the model extraction process illustrated in Fig. 7, the experimental procedure is as follows:

i) Capture PA input/output signal without DPD.
ii) Using the captured input/output data, apply SD-SPSA to calculate the error model coefficients vector $\hat{\mathbf{C}}_{error}$ in the inner loop of training.
iii) Update DPD coefficients using Equation (16).
iv) Using the updated DPD coefficients, generate the predistorted input signal, upload to the RF test bench.
v) Re-capture PA output signal generated by the new predistorted signal.
vi) Repeat steps ii) to v) until the desired linearization performance is achieved.

The sampling rate of the test platform was 368.64 MHz. The closed-loop coefficient adaptation factor λ was 0.7 and on each run 15,000 iterations of the SD-SPSA algorithm were used to calculate the error model coefficients.

1) *Performance with 20 MHz LTE Signal*

A 20 MHz single band LTE signal with 6.5 dB peak to average power ratio (PAPR) was used in the first test. The

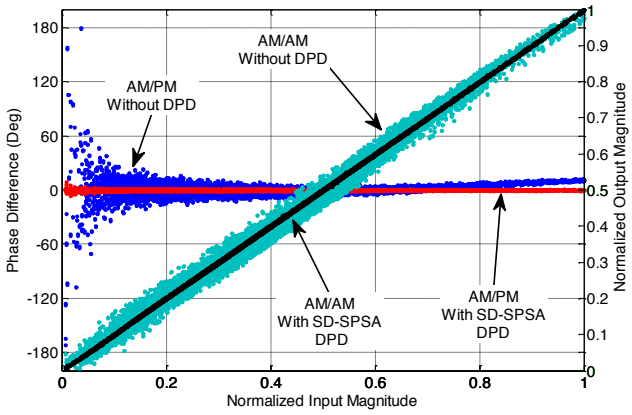

Fig. 10. AM/AM and AM/PM plots for 20 MHz LTE signal with and without DPD

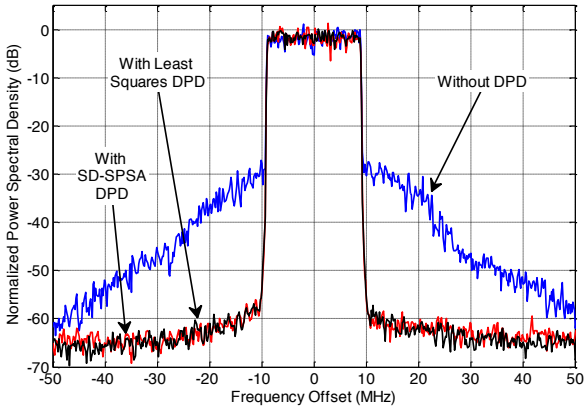

Fig. 11. Output spectra for 20 MHz LTE signal.

predistortion model was a DVR-based function, with S=8 and M=3. Table III reports the performance of the SD-SPSA algorithm over the course of a series of closed loop estimation training runs in terms of adjacent channel power ratio (ACPR) and NMSE. After 10 runs, strong linearization performance is achieved. This is confirmed in Fig. 10 where the AM/AM and AM/PM plots are compared for the signal with and without DPD.

For comparison between the proposed solution and existing methods, linearization results for a conventional LS algorithm applied in the same closed loop architecture are reported. As in the SD-SPSA simulations a training set of 15,000 input/output samples was captured to perform each training run using the LS algorithm. Results are reported in Fig. 11 and further detailed in Table III where it can be seen that the proposed SD-SPSA solution achieves comparable time-domain NMSE and frequency domain ACPR performance.

We also conducted tests using the LMS and RLS algorithms. Due to the limited number of training samples available, LMS could not converge properly and thus the performance is very poor with NMSE only reaching -30 dB. RLS can achieve similar performance to the SPSA, but RLS requires more complex operations at each iteration.

2) Performance with 60 MHz UMTS Signal

To test performance in a wideband extraction scenario, a 60 MHz 12-carrier UMTS signal was used. The signal had 6.5 dB

TABLE IV
SD-SPSA DPD PERFORMANCE FOR 60 MHZ UMTS SIGNAL

Scenario	NMSE (dB)	ACPR (dB)	
		-5 MHz	+5 MHz
No DPD	**-9.92**	**-35.6**	**-30.1**
SD-SPSA (10 runs)	**-39.23**	**-51.32**	**-51.06**
Least Squares (10 runs)	**-39.66**	**-51.64**	**-52.01**

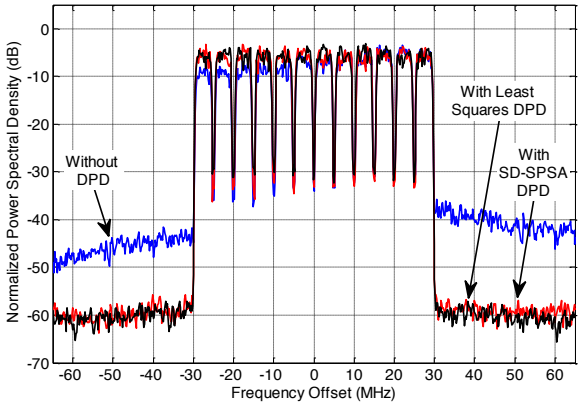

Fig. 12. Output spectra comparison for 60 MHz, 12-carrier UMTS signal

PAPR and was applied in the test bench setup in Fig. 9 with an LDMOS Doherty PA with average output power of 34 dBm. A DVR predistortion model with S=8 and increased memory length M=5 was used to account for increased memory effects due to the wider bandwidth. The SD-SPSA algorithm was band-limited to an observation bandwidth of 140 MHz.

Linearization performance is reported in Table IV for DPD with the closed loop coefficient estimation using both conventional LS and SD-SPSA. Both techniques are shown to achieve similar linearization performance, reducing the ACPR by approximately 20 dB and the NMSE by approximately 30 dB. Fig. 12 reports the measured spectra at the PA output for the wideband signal, confirming the strong linearization performance of the SD-SPSA algorithm, directly comparable to the LS estimation.

VII. CONCLUSION

This work presents a novel DPD model extraction solution based on a stochastic optimization technique incorporated in a closed-loop coefficient estimation architecture. The proposed algorithm avoids computationally intensive Hessian and gradient calculations, instead using loss function measurements to approximate the gradient and iteratively update the coefficient estimate. The proposed closed-loop technique also avoids generating a second set of model terms as required in indirect learning structures. This further reduces the system complexity, making it well-suited to low-cost FPGA implementation. Experimental results show that the proposed approach achieves excellent linearization performance, with accuracy comparable to that achieved using the conventional LS method.

APPENDIX

The quadratic interpolation SPSA update equation in (10) can be developed as follows.

For a linear-in-parameters model, NMSE is a quadratic function of the model coefficients [13]. Thus, for a given iteration, *k*, of the SPSA algorithm, a 2-dimensional section of the loss function can be defined along a given direction (determined by the perturbation sequence) in terms of any one of the model coefficients $\hat{\mathbf{C}}_{k,p}$ as:

$$f(\hat{\mathbf{C}}_k) = \phi_{1,k,p}\left(\hat{\mathbf{C}}_{k,p}\right)^2 + \phi_{2,k,p}\hat{\mathbf{C}}_{k,p} + \phi_{3,k,p} \qquad \text{(A.1)}$$

where the unique quadratic function parameters $\phi_{1,k,p}$, $\phi_{2,k,p}$, and $\phi_{3,k,p}$ exist for each model coefficient p and iteration k. The minimum of (A.1) represents the best performance that can be achieved by varying the weighting factor for a given set of fixed coefficient and perturbation vectors.

Finding the coefficient set corresponding to the quadratic minimum can be formulated as a Newton-based optimization problem. For a simple quadratic minimization problem Newton's method is given by:

$$\hat{\mathbf{C}}_{n+1} = \hat{\mathbf{C}}_n - \mu\frac{f'(\hat{\mathbf{C}}_n)}{f''(\hat{\mathbf{C}}_n)} \qquad \text{(A.2)}$$

where $\hat{\mathbf{C}}_{n+1}$ is the updated optimum parameter estimate, $\hat{\mathbf{C}}_n$ the current parameter set, and $f'(\hat{\mathbf{C}}_n)$, $f''(\hat{\mathbf{C}}_n)$ are the first and second derivatives of the function $f(\hat{\mathbf{C}}_n)$ for which the minimum is sought. For the quadratic in (A.1), the first and second derivatives are given by:

$$f'(\hat{\mathbf{C}}_{k,p}) = 2\phi_{1,k,p}\hat{\mathbf{C}}_{k,p} + \phi_{2,k,p} \qquad \text{(A.3)}$$

and

$$f''(\hat{\mathbf{C}}_{k,p}) = 2\phi_{1,k,p}\,. \qquad \text{(A.4)}$$

Newton's method is derived from a second order truncated Taylor series, so only a single iteration is needed to reach the quadratic minimum:

$$\arg\min_{\hat{\mathbf{C}}}(f(\hat{\mathbf{C}}_k)) = \hat{\mathbf{C}}_{k,p} - \frac{2\phi_{1,k,p}\hat{\mathbf{C}}_{k,p} + \phi_{2,k,p}}{2\phi_{1,k,p}}\,. \qquad \text{(A.5)}$$

Substituting for $\phi_{1,k,p}$ and $\phi_{2,k,p}$ using the algebraic expressions in terms of the three measured points $(\hat{\mathbf{C}}_{k,p}, L(\hat{\mathbf{C}}_k))$, $(\hat{\mathbf{C}}^+_{k,p}, L(\hat{\mathbf{C}}^+_k))$, and $(\hat{\mathbf{C}}^-_{k,p}, L(\hat{\mathbf{C}}^-_k))$, (A.5) is given by:

$$\arg\min_{\hat{\mathbf{C}}}(f(\hat{\mathbf{C}}_k)) = \hat{\mathbf{C}}_{k,p} - \frac{\boldsymbol{\Delta}_{k,p}\left(L\left(\hat{\mathbf{C}}^+_k\right) - L\left(\hat{\mathbf{C}}^-_k\right)\right)}{L\left(\hat{\mathbf{C}}^-_k\right) + L\left(\hat{\mathbf{C}}^+_k\right) - 2L\left(\hat{\mathbf{C}}_k\right)}\,. \qquad \text{(A.6)}$$

The quadratic interpolation SPSA algorithm generates the updated optimum coefficient estimate using the interpolated quadratic minimum for each term in the coefficient vector. Thus the complete update algorithm at each iteration is a generalized version of (A.6):

$$\hat{\mathbf{C}}_{k+1} = \hat{\mathbf{C}}_k - \frac{\boldsymbol{\Delta}_k\left(L\left(\hat{\mathbf{C}}^+_k\right) - L\left(\hat{\mathbf{C}}^-_k\right)\right)}{L\left(\hat{\mathbf{C}}^-_k\right) + L\left(\hat{\mathbf{C}}^+_k\right) - 2L\left(\hat{\mathbf{C}}_k\right)} \qquad \text{(A.7)}$$

where $\boldsymbol{\Delta}_k$ and $\hat{\mathbf{C}}_k$ are the complete perturbation and coefficient vectors respectively.

REFERENCES

[1] J. Wood, "Behavioral modeling and linearization of RF power amplifiers." Norwood, MA: Artech House, 2014

[2] F. Luo, "Digital front-end in wireless communications and broadcasting." Cambridge, U.K.: Cambridge Univ., 2011.

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

[7] A. Zhu, P. J. Draxler, J. J. Yan, T. J. Brazil, D. F. Kinball, 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.

[8] 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.

[9] 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.

[10] N. Kelly and A. Zhu, "A modified simultaneous perturbation stochastic optimization algorithm for digital predistortion model extraction," *Int. Integr. Nonlinear Microw. Millimetre-Wave Circuits Workshop (INMMIC)*, Taormina, Italy, Oct. 2015, pp. 1-3.

[11] 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.

[12] F. M. Ghannouchi, O. Hammi, and M. Helaoui, "Characterization and identification techniques," in *Behavioral Modeling and Predistortion of Wideband Wireless Transmitters*, 1st ed. London, U.K., Wiley, 2015, ch. 8, pp. 170-183.

[13] P. S. R. Diniz, "Fundamentals of adaptive filtering," in *Adaptive Filtering Algorithms and Practical Implementation*, 3rd ed., New York: Springer, 2008.

[14] J. C. Spall, "Multivariate stochastic approximation using a simultaneous perturbation gradient approximation," *IEEE Trans. Automat. Contr.*, vol. 37, no. 3, pp. 332-341, Mar. 1992.

[15] 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.

[16] M. S. Muha, C. J. Clark, A. A. Moulthrop, and C. P. Silva, "Validation of power amplifier nonlinear block models," in *IEEE MTT-S Int. Microw. Symp. Dig.*, 1999, vol. 2, pp. 759-762.

[17] A. V. Keerthi and P. Choudary, "Method and apparatus to optimize adaptive radio-frequency systems," U.S. Patent 0258 591, Oct. 20, 2011.

[18] 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, February 2015.

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

[20] L. Guan and A. Zhu, "Low-cost FPGA implementation of Volterra series-based digital predistorter for RF power amplifiers," *IEEE Trans. Microw. Theory Techn.*, vol. 58, no. 4, pp. 866 - 872, April 2010.
[21] *7 Series FPGAs Overview*, 1st ed., Xilinx, Inc., San Jose, CA, 2015.
[22] L. Guan, R. Kearney, C. Yu, and A. Zhu, "High performance digital predistortion test platform development for wideband RF power amplifiers," *Int. J. Microw. Wireless Technologies*, vol. 5, no. 2, pp. 149-162, April 2013.

Noel Kelly (S'15) received the BE degree in electronic engineering from the School of Electrical and Electronic Engineering, University College Dublin (UCD), Ireland in 2012 before joining the RF and Microwave Research Group at UCD. He is originally from Sligo, Ireland and is currently working towards the PhD degree in electronic engineering.

His research interests include low complexity digital predistortion architectures, efficient field programmable gate array (FPGA) 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 University College Dublin (UCD), Dublin, Ireland, 2004.

He is currently a Senior Lecturer with the School of Electrical and Electronic Engineering, UCD. His research interests include high-frequency nonlinear system modeling and device characterization techniques with a particular emphasis on behavioral modeling and linearization for RF power amplifiers (PAs). He is also interested in wireless and RF system design, digital signal processing, and nonlinear system identification algorithms.