

# Analysis of Power Amplifier Modeling Schemes for Crosscorrelation Predistorters

André B.J. Kokkeler

University of Twente

P.O. Box 217, 7500 AE Enschede, The Netherlands.

Phone: +31 53 4894291, Fax: +31 53 4894571,

[kokkeler@cs.utwente.nl](mailto:kokkeler@cs.utwente.nl)

**Abstract**— Amplification of signals with fluctuating envelopes leads to distortion because of non-linear behavior of the Power Amplifier (PA). Digital Predistortion can counteract these non-linear effects. A crosscorrelation predistorter is a digital predistorter, based on the calculation of crosscorrelation functions using coarsely quantized signals. The crosscorrelation functions are transformed to the frequency domain and the spectra are used to calculate the coefficients of the predistorter memory polynomial. This method has reduced complexity and equivalent performance in comparison with existing schemes. In this paper, four alternative schemes to implement a crosscorrelation predistorter are analyzed. The PA characteristics can be determined either directly or indirectly using 'normal' or orthogonal polynomials giving four alternatives. All four alternatives give significant reduction of Adjacent Channel Interference.

**Keywords**— Power Amplifiers, Digital Predistortion, Crosscorrelation, Fourier Transform, Memory Polynomial.

## I. INTRODUCTION

POWER Amplifiers (PAs) are inherently non-linear. One of the techniques to linearize PAs is digital predistortion (see [1]). The general principle is to apply the inverse input-output relation of the PA to the signal at the input of the PA. Predistortion followed by the PA (and its inherent distortion) should result in linear amplification. In digital predistortion, the digital baseband signal is predistorted before it is converted to the analog domain, frequency translated to RF and amplified (see figure 1).

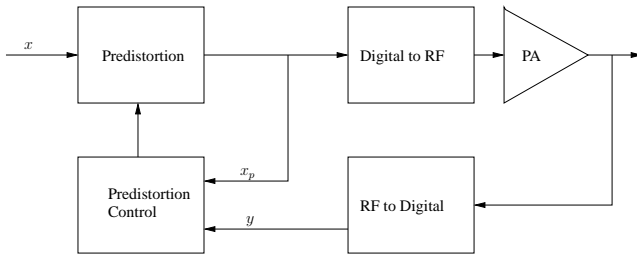


Fig. 1. Digital Predistortion

Because the input-output relation of the PA changes in time due to temperature changes and aging of components in the analog part, a control mechanism constantly adapts the predistortion. A small fraction of the PA output is fed back and converted from RF to baseband. An adaptation algorithm compares this signal with the output of the predistorter. Two different approaches exist: Direct- and Indirect Learning. When using Direct Learning, the input-output relation of the PA is determined. This relation is inverted and used for predistortion. For an example, see [3]. The Indirect Learning scheme was introduced in [4]. The input-output relation of the PA is determined indi-

rectly, the inverse relation directly. The PA (in case of Direct Learning) or the predistorter (in case of Indirect Learning) can be modeled using memory polynomials (see [3]). One can use 'normal' polynomials, which are simply the powers of different orders, or orthogonal polynomials (see [6]).

In this document we present simulation results of a crosscorrelation predistorter based on the four combinations: Indirect Learning - normal polynomials, Direct Learning - normal polynomials, Indirect Learning - orthogonal polynomials and Direct Learning - orthogonal polynomials. First we will introduce a general scheme which fits the four combinations. Second the crosscorrelation predistorter will be described and results from simulations will be presented.

## II. THE CROSSCORRELATION PREDISTORTER

### A. General Scheme

Figure 1 shows that the predistorted signal  $x_p$  and the PA output (converted to baseband)  $y$  are used to determine the non linear characteristics of the PA. The output can be written as a function of the input:  $y = f_1(x_p)$  (Direct Learning) or vice versa:  $x_p = f_2(y)$  (Indirect Learning). This is generally described by:  $x_2 = f(x_1)$ , where  $x_1 = x_p$  and  $x_2 = y$ , for Direct Learning and  $x_1 = y$  and  $x_2 = x_p$ , for Indirect Learning. The function  $f$  can be described by means of basis functions  $\gamma_k(x)$ :

$$x_2(t) = \sum_{k=1}^K \sum_{\tau=0}^{\tau_{max}-1} a_{k\tau} \gamma_k(x_1(t-\tau)) \quad (1)$$

We can use polynomial basis functions  $\gamma_k = \phi_k$  where  $\phi_k(x) = |x|^{k-1} x$ . These basis functions can lead to instabilities in the calculations further on. In [6] it is suggested to use orthogonal polynomial basis functions  $\psi_k$ . We will only give odd polynomials up to the fifth order (for other basis functions see [6]):

$$\begin{aligned} \psi_1(x) &= x \\ \psi_3(x) &= 15 |x|^2 x - 20 |x| x + 6x \\ \psi_5(x) &= 210 |x|^4 x - 504 |x|^3 x + 420 |x|^2 x \\ &\quad - 140 |x| x + 15x \end{aligned} \quad (2)$$

By using the definitions above, 4 different predistortion schemes exist:

1. Indirect Learning, normal polynomials:  $x_1 = y$ ,  $x_2 = x_p$ ,  $\gamma_k = \phi_k$ .

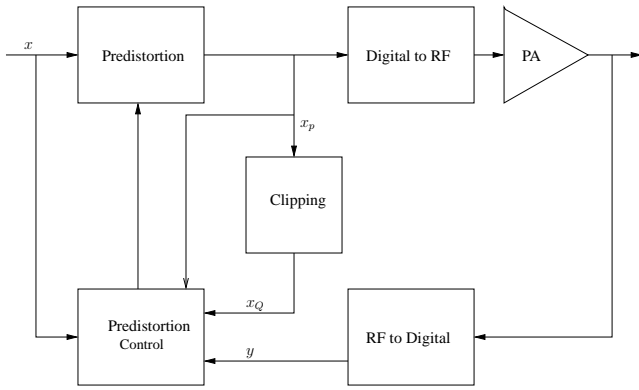


Fig. 2. Digital Crosscorrelation Predistortion

2. Indirect Learning, orthogonal polynomials:  $x_1 = y$ ,  $x_2 = x_p$ ,  $\gamma_k = \psi_k$ .

3. Direct Learning, normal polynomials:  $x_1 = x_p$ ,  $x_2 = y$ ,  $\gamma_k = \phi_k$ .

4. Direct Learning, orthogonal polynomials:  $x_1 = x_p$ ,  $x_2 = y$ ,  $\gamma_k = \psi_k$ .

Using  $x_1$ ,  $x_2$  and  $\gamma$ , a general description of the crosscorrelation predistorter can be given for all 4 predistortion schemes. We define the signals:

$$\gamma_{k\tau}(t) = \gamma_k(x_1(t - \tau)) \quad (3)$$

for  $t \in \mathbb{Z}$ . Via linear combination of these signals, an estimate of signal  $x_2$  can be constructed using the least squares criterion. In the crosscorrelation predistorter however, the signals  $\gamma_{k\tau}$  are not combined directly. First  $\gamma_{k\tau}$  and  $x_2$  are crosscorrelated with a reference signal. We chose to crosscorrelate with a single-bit representation  $x_Q$  of  $x_p$  (see figure 2) because the quantized signal  $x_Q$  has significant power in the adjacent channels when the predistorter is operational.

Single-bit quantization is defined as:

$$x_Q(x_p) = \text{sign}(\text{Re}(x_p)) + j\text{sign}(\text{Im}(x_p)) \quad (4)$$

where  $\text{sign}()$  is defined as:

$$\text{sign}(x) = \begin{cases} -1 & x < 0 \\ 1 & x \geq 0 \end{cases} \quad (5)$$

$\text{Re}(x)$  indicates the real part of a complex value  $x$  and  $\text{Im}(x)$  the imaginary part. Using a single-bit quantized value as one of the operands of a crosscorrelation drastically reduces the complexity. The multiplications involved become very simple to implement: the second operand of the multiplication is either left unchanged or its sign is reversed if the first operand equals 1 or -1 respectively. The number of points (or lags) of the crosscorrelation is even and equals  $N$ . Using vector notations, the single-bit quantized signal equals:

$$\mathbf{x}_Q = \left[ x_Q(x_p(\frac{1}{2}N + 1)), \dots, x_Q(x_p(T - \frac{1}{2}N + 1)) \right]^T \quad (6)$$

$T$  is the number of consecutive samples, available to the predistorter and is generally much larger than  $N$ . We define the matrices  $\mathbf{\Gamma}_{k\tau}$  for  $k = 1, \dots, K$  and  $\tau = 0, \dots, \tau_{max} - 1$ , and  $\mathbf{X}_2$  as:

$$\mathbf{\Gamma}_{k\tau} = \begin{bmatrix} [\gamma_{k\tau}(1), \dots, \gamma_{k\tau}(T - N + 1)]^T, \dots, \\ [\gamma_{k\tau}(N), \dots, \gamma_{k\tau}(T)]^T \end{bmatrix}^T \quad (7)$$

$$\mathbf{X}_2 = \begin{bmatrix} [x_2(1), \dots, x_2(T - N + 1)]^T, \dots, \\ [x_2(N), \dots, x_2(T)]^T \end{bmatrix}^T \quad (8)$$

The crosscorrelations are defined as:

$$\mathbf{r}_{k\tau} = \mathbf{\Gamma}_{k\tau} \mathbf{x}_Q^* \quad (9)$$

$$\mathbf{r} = \mathbf{X}_2 \mathbf{x}_Q^* \quad (10)$$

The crosscorrelation vectors  $\mathbf{r}_{k\tau}$  and  $\mathbf{r}$  are tapered with a Hanning taper:

$$h(j) = \frac{1}{2}(1 - \cos(2\pi(j - 1)/(N - 1))) \quad (11)$$

$$\mathbf{H} = \text{diag}(h(1), \dots, h(N)) \quad (12)$$

where 'diag()' indicates a diagonal  $N \times N$  matrix. A Discrete Fourier Transform is applied to the tapered crosscorrelation vectors:

$$\mathbf{f}_{k\tau} = \mathbf{W} \mathbf{H} \mathbf{r}_{k\tau} \quad (13)$$

$$\mathbf{f} = \mathbf{W} \mathbf{H} \mathbf{r} \quad (14)$$

where  $\mathbf{W}$  equals the DFT kernel. The elements  $w_{pq}$  of the kernel are defined as:

$$w_{pq} = \exp^{-i2\pi \frac{p-1}{N}(q-1)} \quad (15)$$

If the vectors (spectra)  $\mathbf{f}_{k\tau}$  are concatenated into a matrix  $\mathbf{F} = [\mathbf{f}_{10}, \dots, \mathbf{f}_{K0}, \dots, \mathbf{f}_{1\tau_{max}}, \dots, \mathbf{f}_{K\tau_{max}}]$  and the vector  $\mathbf{a}$  is defined as  $\mathbf{a} = [a_{10}, \dots, a_{K0}, \dots, a_{1\tau_{max}}, \dots, a_{K\tau_{max}}]$ , the memory polynomial predistorter is described in the frequency domain as:

$$\mathbf{f} = \mathbf{F} \mathbf{a} \quad (16)$$

The least squares solution minimizes the *absolute* error over the frequency domain. Because the signal has a relatively low power spectral density in the adjacent channels, solving equation 16 can result in solutions where the *relative* errors in the adjacent channels are (too) large. For that reason, we minimize

the relative error. This is realized by normalizing the spectra with  $\mathbf{f}$ :

$$g_{k\tau}(n) = \frac{f_{k\tau}(n)}{f(n)} \quad (17)$$

$$\mathbf{g}_{k\tau} = [g_{k\tau}(1), \dots, g_{k\tau}(N)]^T \quad (18)$$

$$\mathbf{G} = [\mathbf{g}_{10}, \dots, \mathbf{g}_{k0}, \dots, \mathbf{g}_{1\tau_{\max}}, \dots, \mathbf{g}_{k\tau_{\max}}] \quad (19)$$

The  $N$  element vector  $\mathbf{g}$  is defined as:

$$\mathbf{g}(n) = 1, \quad n = 1, \dots, N \quad (20)$$

The normalized version of equation (16) becomes:

$$\mathbf{g} = \mathbf{G}\mathbf{a} \quad (21)$$

The least-squares solution  $\hat{\mathbf{a}}$  equals:

$$\hat{\mathbf{a}} = (\mathbf{G}^H \mathbf{G})^{-1} \mathbf{G}^H \mathbf{g} \quad (22)$$

where  $^H$  indicates the complex conjugate transpose. To reduce the effects of noise on the correlation functions  $\mathbf{g}_{k\tau}$ , we select only those vector elements which represent the power in the primary channel and the first adjacent channel leading to vectors  $\mathbf{g}$  and  $\mathbf{g}_{k\tau}$  with limited length.

### B. Implementation Aspects

The complexity of current polynomial predistorters is at least  $O(T)$ ; the number of complex multiplications scales linearly with the number of samples used to update the predistorter polynomial coefficients  $\hat{\mathbf{a}}$ . In the crosscorrelation predistorter, the size of the vectors  $\mathbf{f}_p$  and  $\mathbf{f}_{k\tau}$  is reduced to  $N$  (instead of  $T$ ). In general  $N$  is much smaller than  $T$ . If an FFT is used to transform the vectors from the time domain to the frequency domain, the complexity is  $O(N \log_2 N)$ . The reduction of the length of the vectors is due to the crosscorrelation. The crosscorrelation is easy to implement. Due to the single-bit quantization of  $x_p$ , no full-precision complex multiplications are required, so the overall complexity of the crosscorrelation predistorter is determined by the FFT.

### C. The inverse of the PA memory polynomial

In the Indirect Learning scheme, the estimate of the polynomial coefficients ( $\hat{\mathbf{a}}$ ) can be directly used in the predistorter; the incoming data  $x$  is predistorted according to the memory polynomial determined by the polynomial coefficients  $\hat{\mathbf{a}}$ . In the Direct Learning scheme, however, the inverse of the memory polynomial has to be determined. A procedure to approximate the inverse of a memory polynomial in case of a Direct Learning scheme, was presented in [3]. The predistorted signal  $x_p$  is generated according to the following algorithm:

$$x_p(t) = \frac{1}{\beta_0(|x_p(t)|)} \left( x(t) - \beta_1(|x_p(t-1)|)x_p(t-1) \right) \quad (23)$$

where

$$\beta_\tau(x) = \sum_{k=1}^K \tilde{a}_{k\tau} x^{k-1} \quad (24)$$

$$\tilde{a}_{k\tau} = \hat{a}_{k\tau} \quad (25)$$

The problem with this algorithm is that  $|x_p(t)|$  needs to be known in order to calculate  $x_p(t)$ . In [3], it is suggested to use  $x_p^{initial} = x(t)$  as an initial value. The algorithm then calculates a new estimate of  $x_p(t)$ , uses this as the next initial value, calculates a new estimate etc. This process is repeated until a stable estimate of  $x_p(t)$  is obtained. The algorithm gives satisfactory results if the PA satisfies 2 conditions. First, the memory effects should not be severe which means that the  $\beta_1$  values should be significantly smaller than the  $\beta_0$  values. Second, the distortion should be weak; the first order parameter  $\hat{a}_{10}$  should be close to 1. To cope with amplifiers with severe distortion and memory effects, we modified the algorithm. In case of severe memory effects, where  $\hat{a}_{11} > \hat{a}_{10}$ , we assume that  $\hat{a}_{k2} = \hat{a}_{k0}$  and use an additional delay in the predistorter:

$$\begin{aligned} \tilde{a}_{k\tau} &= \hat{a}_{k\tau} \quad \text{if } \hat{a}_{11} \leq \hat{a}_{10} \\ &= \hat{a}_{k\tau+1} \quad \text{if } \hat{a}_{11} > \hat{a}_{10} \end{aligned} \quad (26)$$

Instead of using  $x_p^{initial}(t) = x(t)$  we used as initial value:

$$x_p^{initial}(t) = \frac{1}{\tilde{a}_{10}} \left( x(t) - \beta_1(|x_p(t-1)|)x_p(t-1) \right) \quad (27)$$

The part between the large brackets takes into consideration the memory effects from the beginning. The effects of the first order component differing from 1 are canceled by division with  $\tilde{a}_{10}$ .

## III. SIMULATION RESULTS

A simulator, based on the structure presented in the previous sections has been implemented. As a signal source we used two data generators: one for the real part of the signal and one for the imaginary part. A data generator is based on the signal source used in [2] and generates sine waves with different consecutive frequencies, equal amplitudes and randomly selected phase. The oversampling factor is 10. We used 2 PA models, both presented in [5].

### A. PA model 1

This model is a Wiener-Hammerstein model. It consists of an IIR filter  $H(z)$  followed by a memoryless polynomial which on its turn is followed by a second IIR filter  $G(z)$ . The filter specifications are:

$$H(z) = \frac{1 + 0.5z^{-2}}{1 - 0.2z^{-1}}, \quad G(z) = \frac{1 - 0.1z^{-2}}{1 - 0.4z^{-1}} \quad (28)$$

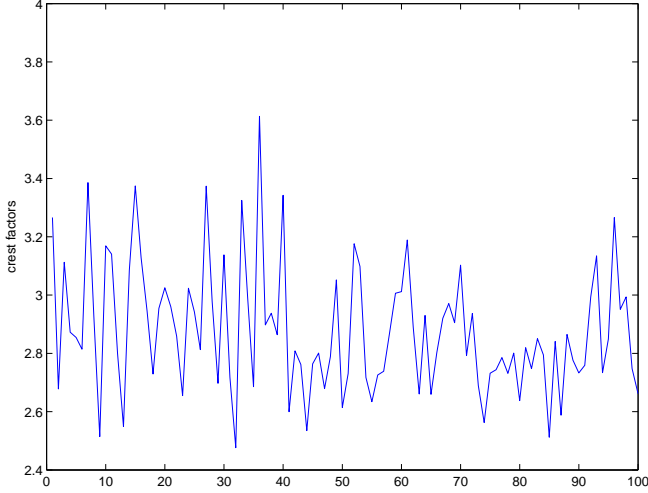


Fig. 3. Crest-factors of the generated datasets

The memoryless polynomial equals:

$$w(n) = \sum_{l=1}^L b_l v(n) |v(n)|^{l-1} \quad (29)$$

where  $v$  indicates the output of filter  $H$  and  $w$  the input of filter  $G$ . The coefficients are:  $b_1 = 1.0108 + 0.0858j$ ,  $b_3 = 0.0879 - 0.1583j$ ,  $b_5 = -1.0992 - 0.8891j$ . All other  $b$ -values equal zero. The standard deviation of the complex input signal is  $4.24 \cdot 10^{-2}$ . In our simulations, we used blocks of 8K samples ( $T = 8192$ ) and 64-point crosscorrelation functions ( $N = 64$ ). For the predistorter, we used only odd polynomials up to the fifth order ( $k = 1, 3, 5$ ) and the memory depth is 2 ( $\tau = 0, 1$ ).

The performance of the different schemes is determined by the ACI levels: the power in the channel adjacent to the primary channel. This power is related to the power in the primary channel and for that reason measured in dB's. The performance that is achieved depends on the input signal. New settings of the predistorter are based on data that already has been transmitted. New data might contain signal-excursions which have not been accounted for. This can lead to limited suppression of ACI levels and in severe cases to settings from which the predistorter cannot recover. For that reason, we simulated for all 4 predistortion schemes, 100 cycles where every cycle consisted of the following stages: generation of an 8K samples dataset for cycle  $i$ , predistortion of this dataset using the predistorter settings found in cycle  $i - 1$ , distortion, determination of the memory polynomial and determination of the new predistorter settings by averaging the memory polynomial found (weight = 0.25) with the averaged previous polynomials (weight = 1). For all four predistortion schemes the same data is used and the crest-factors (ratio of the peak amplitude and the standard deviation) of the generated data are given in figure 3.

The bandwidth of the adjacent channel equals the bandwidth of the primary channel. The average ACI levels in the adjacent channel are presented with- and without predistortion in figure 4.

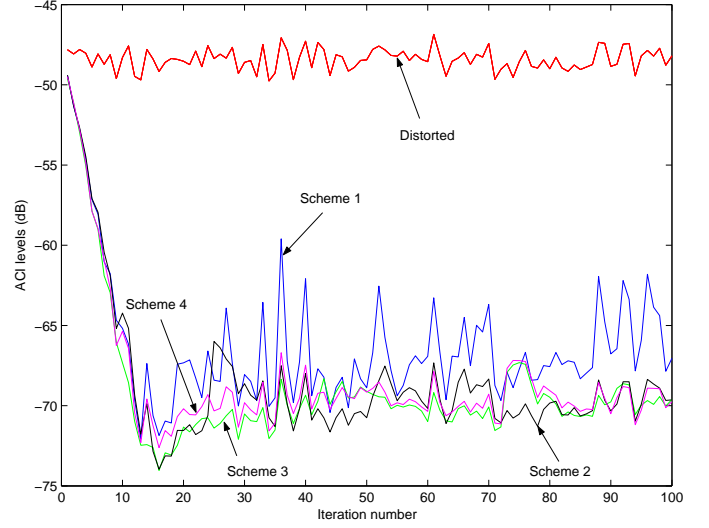


Fig. 4. ACI levels with and without predistortion for PA model 1

We see that after 10 cycles, the predistorter system stabilizes. Scheme 1 (Indirect Learning - normal polynomials) has a slightly worse performance than the other schemes. On average, the suppression of power in the adjacent channels is 20 dB.

#### B. PA model 2

The second PA model is a parallel Wiener model (see [5]):

$$y(n) = \sum_{i=1}^3 F_i(H_i(x(n))) \quad (30)$$

$$H_1(z) = 1, \quad H_2(z) = \frac{1 + 0.3z^{-1}}{1 - 0.1z^{-1}}, \quad H_3(z) = \frac{1 - 0.2z^{-1}}{1 - 0.4z^{-1}} \quad (31)$$

$$F_i(x) = \sum_{l=1}^L a_{il} x |x|^{l-1} \quad (32)$$

The values for the polynomial distortion coefficients are:

$$\begin{aligned} a_{11} &= 1.0108 + 0.0858j, & a_{31} &= 0.0879 - 0.1583j, \\ a_{51} &= -1.0992 - 0.8891j, & a_{12} &= 0.1179 + 0.0004j, \\ a_{32} &= -0.1818 + 0.0391j, & a_{52} &= 0.1684 + 0.0034j, \\ a_{13} &= 0.0473 - 0.0058j, & a_{33} &= 0.0395 + 0.0283j, \\ a_{53} &= -0.1015 - 0.0196j \end{aligned} \quad (33)$$

We used the same signal source as for PA model 1 with the only difference that the standard deviation equals  $8.5 \cdot 10^{-2}$ . In figure 5, the average ACI levels with and without predistortion are given.

Also for PA model 2, Scheme 1 (Indirect Learning - normal polynomials) has a slightly worse performance. On average, the suppression of power in the adjacent channels equals 35 dB. Remarkable however is the good performance of Scheme 3 (Direct Learning - normal polynomials). This scheme gives an additional suppression of 5 dB for this PA model.

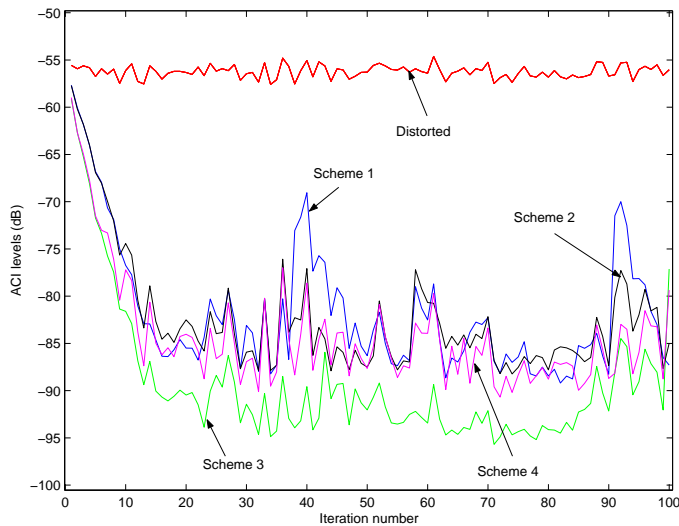


Fig. 5. ACI levels with and without predistortion for PA model 2

#### IV. CONCLUSIONS

In this paper, four different PA-predistortion schemes, in combination with a crosscorrelation predistorter, have been investigated: Indirect Learning - normal polynomials, Direct Learning - normal polynomials, Indirect Learning - orthogonal polynomials and Direct Learning - orthogonal polynomials. A simulation model has been implemented and the performance of the different schemes has been determined. Performance is defined as the power in the adjacent channel (ACI-levels) related to the power in the primary channel. All four schemes give stable performance and a significant reduction of ACI levels. The performance depends on the input data and the PA model. The combination Indirect Learning - normal polynomials gives in general the worst performance and largest variations. The other three schemes have more or less equal performance. The Direct Learning - orthogonal polynomials scheme in combination with PA model 2, has a performance which is better than the performance of other schemes.

#### REFERENCES

- [1] F. Zavosh, M. Thomas, C. Thron, T. Hall, D. Artusi, D. Anderson, D. Ngo, and D. Runtun, "Digital predistortion techniques for RF power amplifiers with CDMA applications," *Microwave Journal*, Oct. 1999.
- [2] J. K. Cavers, "Amplifier linearization using a digital predistorter with fast adaptation and low memory requirements", *IEEE Trans. Veh. Technol.*, vol. 39, no. 4, Nov. 1990, pp. 374-383.
- [3] J. Kim and K. Konstantinou, "Digital predistortion of wideband signals based on power amplifier model with memory," *Electronics Letters*, vol. 37, no. 23, Nov. 2001, pp. 1417-1418.
- [4] C. Eun and E.J. Powers, "A predistorter design for a memory-less nonlinearity preceded by a dynamic linear system", *Proc. GLOBECOM 1995*, pp. 152-156.
- [5] L. Ding, G.T.Zhou, Z. Ma, D.R. Morgan, J.S. Kenney, J. Kim, and C.R. Giardina, "A robust digital baseband predistorter constructed using memory polynomials," *IEEE Trans. on Communications*, March 2002 (submitted).
- [6] R. Raich, H. Qian, and G.T. Zhou, "Digital baseband predistortion of nonlinear power amplifiers using orthogonal polynomials," Available: [http://users.ece.gatech.edu/qianhua/publications\\_files/Hua\\_ICASSP03.pdf](http://users.ece.gatech.edu/qianhua/publications_files/Hua_ICASSP03.pdf)