Complexity Analysis of Power Amplifier Behavioral Models

Ali Soltani Tehrani, Haiying Cao†, Thomas Eriksson, Christian Fager†

Department of Signals and Systems, Communication Systems group
†Department of Microtechnology and Nanoscience, Microwave Electronics Laboratory
Chalmers University of Technology, Gothenburg, Sweden

***Abstract*— In this paper efficient computer implementations of some of the most commonly used Volterra series based power amplifier behavioral models are proposed. The desired efficiency is in regard to algorithm complexity and floating point operations. Finally a comparative overview of the different behavioral models with respect to their complexity is presented.**

***Index Terms*— Behavioral modeling, complexity, FLOPs, power amplifier, Volterra series.**

I. Introduction

Power amplifiers are one of the main sources of distortion in a wireless communication system. Digital predistortion is widely employed to compensate for the effects of this distortion, and requires behavioral modeling of the power amplifier. From general polynomial theory it is well known that any nonlinear function can be approximated to an arbitrary resolution with a polynomial - the Weierstrass's approximation theory. This has given rise to many different power amplifier behavioral models based on polynomials. Some of the most famous are presented in [1] and evaluated in [2].

From this theory, the Volterra series with infinite order can perfectly reconstruct any nonlinear function - therefore completely accurate. In practical applications however, the Volterra series has to be truncated because the computational complexity grows exponentially. In literature often behavioral models are only evaluated based on the accuracy of the modeling. The accuracy however, will be dependent on how the model is truncated and since the truncation is different between behavioral models, making a fair comparison is difficult. In [2] the number of coefficients has been used as an evaluation measure for comparison, but since it does not take into account the complexity of evaluating the coefficients themselves, it is not an appropriate measure. In order to make a fair comparison between behavioral models, the number of floating point operations is proposed in this paper.

When dealing with complexity it is important to obtain the most efficient algorithm for each model to ensure a fair comparison. The number of multiplications and additions for each algorithm needs to be computed and minimized when possible.

In the next section a short description on complexity is given, some behavioral models are discussed, and a measure is proposed for complexity. The results are analyzed in section III.

II. Complexity

Behavioral models generally have different types of complexity. One type can be called *identification complexity*, which is the amount of computational effort needed to identify the behavioral models parameters. However, it may be of less interest because even if it initially takes a long time, it will only be done once and can be adapted afterwards.

A second type is called *running complexity* - also known as *evaluation complexity*, and is the complexity of evaluating the model equations for each communication symbol. This can be of more practical importance since a more complex model will require more hardware. In this work, the behavioral models are classified with respect to running complexity.

This section gives a short description of the different behavioral models analyzed in this work. A more complete description can be found in [1]. Moreover an efficient computer algorithm is proposed for each behavioral model and the number of multiplications and additions is derived. In order to normalize the comparison, it is assumed that the input is a vector $[x(n)x(n-1)\cdots x(n-M)]$ where M is the memory depth and that the algorithms have access to the different memory depths at no cost (in terms of complexity).

A. Behavioral model complexity

In this section power amplifier behavioral models that are based on polynomials and mainly the Volterra series are presented.

1) Volterra: The classical complex triangular-form baseband equivalent Volterra series with nonlinear order $O = (P-1)/2$ and memory depth of M can be written as [3]:

$$y(n) = y_1(n) + y_3(n) + y_5(n) + \cdots + y_P(n), \quad (1)$$

where

$$\begin{aligned} y_p(n) \quad &= \textstyle\sum_{m_1=0}^{M} \sum_{m_2=m_1}^{M} \cdots \sum_{m_{\frac{p+1}{2}}=m_{\frac{p-1}{2}}}^{M} \\ &\textstyle\sum_{m_{\frac{p+3}{2}}=0}^{M} \cdots \sum_{m_p=m_{p-1}}^{M} h_p(m_1, \cdots, m_p) \\ &x(n-m_1)x(n-m_2)\cdots x(n-m_{\frac{p+1}{2}}) \\ &x^*(n-m_{\frac{p+3}{2}})\cdots x^*(n-m_p). \end{aligned} \quad (2)$$

The algorithm for evaluating (2) can be implemented in two steps:

1) Construct the basis functions (permutations of $x(n)$).
2) Filter the basis with the kernels and sum all the resulting outputs ($h * X$).

The second step is common among the different behavioral models, i.e. all outputs are multiplied and summed with the kernels. The first step is where the algorithms generally differ.

In order to obtain efficient algorithms it is necessary to avoid redundant calculations. It can be easily verified that when the first order nonlinearity ($O = 1 \Rightarrow P = 1$) is only considered, all behavioral models have similar basis functions which are just $[x(n)x(n-1)\cdots x(n-M)]$. Therefore, for comparison sake all models will be compared from the second order nonlinearity, i.e. setting $O = 2 \Rightarrow P = 3$. Starting from this order, the Volterra model becomes:

$$\begin{aligned} y_3(n) = \quad &\textstyle\sum_{m_1=0}^{M} \sum_{m_2=m_1}^{M} \sum_{m_3=0}^{M} h_3(m_1, m_2, m_3) \\ &x(n-m_1)x(n-m_2)x^*(n-m_3). \end{aligned} \quad (3)$$

The first step is to construct the basis which are the permutations of $x(n-m_1)x(n-m_2)x^*(n-m_3)$ for the respective m_1,m_2, and m_3. Before looking at the number of multiplications, it is worth to note that if one considers the basis function $x(n)x(n-1)x^*(n-2)$ as an example, it can easily be observed that $x(n-1)x(n-2)x^*(n-3)$ can be constructed as a delayed version of the previous basis - with zero complexity. This can be utilized for every basis function to achieve a more efficient algorithm. Therefore only the permutations that consist of $x(n)$ or $x^*(n)$ terms need to be considered.

In order to construct an efficient algorithm, some terms that will be used often are preconstructed. These consist of the permutations of

$$\begin{aligned} &x(n) \times x^*(n), x(n) \times x^*(n-1), \\ &\quad, \cdots, x(n) \times x^*(n-M). \end{aligned} \quad (4)$$

The total number of coefficients is given in [2] and is equal to *f(M,O)* where:

$$f(M,O) = \binom{M+O}{O}\binom{M+O-1}{O-1}. \quad (5)$$

Of these, the terms that don't consist of $x(n)$ or $x^*(n)$ can be removed according to the previous discussion. These terms can be written as

$$F(M,O) = f(M,O) - f(M-1,O), \quad (6)$$

where $F(M,O)$ is the number terms that have at least one of $x(n)$ or $x^*(n)$. These terms are in the form of $x(n)x(n-m_2)x^*(n-m_3)$ or $x(n-m_1)x(n-m_2)x^*(n)$. Since the permutations of (4) have been precalculated, this means that each of these basis require only one multiplication. Notice that some terms are conjugates of (4). Therefore it can be seen that the total number of multiplications needed for this nonlinear order is equal to $F(M,2)\times 1$ excluding the initial permutations constructed.

The next step is to calculate the number of multiplications for $O = 3 \Rightarrow P = 5$. The basis functions for this order are the permutations of: $x(n-m_1)x(n-m_2)x(n-m_3)x^*(n-m_4)x^*(n-m_5)$. In this formulation one can notice that some multiplications have been already calculated. Reformatting the multiplications as $x(n-m_1)x(n-m_2)x^*(n-m_4)$ it can be seen that this has already been calculated in the previous nonlinear order, and the innovations are only $x(n-m_3)\times x^*(n-m_5)$ which are the precalculated (4). Therefore, for each basis function - $F(M,3)$ - one multiplication has to be done. Following the same procedure one can see that the total number of multiplications can be written as *N(M,O)* where:

$$N(M,O) = (M+1) + \sum_{o=2}^{O} F(M,o), \quad (7)$$

the first part is the initial construction requirement and the second part is for the different nonlinear orders starting from order two.

The second step is to multiply the kernels with the according basis functions, and sum all the outputs. This requires one multiplication and roughly one summation per coefficient. One can see that the cost of constructing the basis functions is comparable with the filtering step for the Volterra series-based behavioral model.

2) Memory polynomial: The memory polynomial model is a commonly used power amplifier behavioral model and is also referred to as the Parallel-Hammerstein model [2]. This behavioral model is expressed as:

$$y(n) = \sum_{p=1}^{P}\sum_{m=0}^{M} h_{p,m}x(n-m)|x(n-m)|^{p-1}. \quad (8)$$

Following the same two step procedure, first the basis functions are created. An efficient method of accomplishing this is by first pre-constructing $|x(n)|^2$ and $|x(n)|$. For the odd order powers $x|n|^2$ can be used by multiplying it with $x(n)$ and for even order terms $|x(n)$. All other $x(n-m)|x(n-m)|^{p-1}$ basis can be constructed by delaying $x(n)|x(n)|^{p-1}$ and hence complexity free. This means that for the first step only $1+P$ multiplications are needed for all combinations. The second step is the same as the Volterra series, one multiplication and one summation per coefficient. It can be noticed that the basis functions in the memory polynomial model are easier to construct than the Volterra series and the main source of complexity is from the filtering.

3) Generalized memory polynomial: This model is similar to the memory polynomial model, with the inclusion of delayed versions and combining them together [4]. A simple lagged term-only representation is shown:

$$\begin{aligned} y(n) = & \sum_{p=0}^{P-1}\sum_{m=0}^{M} h_{p,m,0}x[n-m]|x(n-m)|^p \\ + & \sum_{p=1}^{P-1}\sum_{m=0}^{M}\sum_{g=1}^{G} a_{p,m,g}x[n-m]|x(n-m-g)|^p \\ & +b_{p,m,g}x[n-m]|x(n-m+g)|^p. \end{aligned} \quad (9)$$

With this representation more cross-terms can be exploited. One can observe as in the case for memory polynomial model we are able to reconstruct the basis functions from an initial calculation via delays. In this model an extra G multiplications is needed since $x(n-m)|x(n-m-1)|^{p-1}$ and other similar terms which are not just delayed versions also have to be constructed. When $|x(n)|^2$ and $|x(n)|$ are pre-constructed, for the first step $2P + 2(P-1)G + 2P\min(G, M)$ multiplications are needed. The second step is similar to the previous models.

4) Volterra with Dynamic Deviation Reduction: This behavioral model is a different representation of the Volterra series by reformatting it in terms of dynamics [5]. The cross-terms are organized in terms of the number of dynamics and this gives an extra degree of freedom to truncate the Volterra series. A simple baseband representation of this model can be formulated as:

$$\begin{aligned} y(n) = & \textstyle\sum_{p=1}^{P} h_{p,0} \underbrace{x(n)|x(n)|^{p-1}}_{\text{zero order dynamic}} \\ + & \textstyle\sum_{p=1}^{P}\sum_{m_1=1}^{M} h_{p,m_1} \underbrace{x(n-m_1)|x(n)|^{p-1}}_{1^{\text{st}}\text{ order dynamics path 1}} \\ + & \sum_{p=3}^{P}\sum_{m_2=1}^{M} h_{p,m_2} \underbrace{x^*(n-m_2)x^2(n)|x(n)|^{p-3}}_{1^{\text{st}}\text{ order dynamics path 2}} \end{aligned} \quad (10)$$

where p is odd and 1st order dynamics are shown.

In order to construct the basis functions for this behavioral model, first $|x(n)|^2$ and $x(n)^2$ are constructed. The first line of this model is a static nonlinear model and requires $(P-1)/2$ multiplications. In the second line we notice that there is no possibility of reusing terms from the first line. For path one in the first order dynamics $(P-1)/2 \times M$ multiplications are needed and for path two $((P-3)/2+1) \times M$. The first path only needs real by complex number multiplications while the second line has complex by complex number multiplications as well.

5) Kautz-Volterra: By filtering the input sequence and changing the poles from 0 to an arbitrary complex number and then applying the Volterra series, the Kautz-Volterra behavioral model is proposed in [6]. In this respect it is simple to derive the complexity to be the same as the Volterra series with the addition of the filter - which is one additional multiplication and summation.

B. Complexity measure

Complexity of algorithms is often measured in orders denoted by the Landau symbol $O(.)$, but can also be expressed in FLOPs (floating point operations) which is used in this work. While this measure may not be ideal, it should be suffice to judge between different behavioral models since it evaluates the number of additions and multiplications. Since in the previous sections the number of multiplication and additions were reported, here the conversion into FLOPs is shown in Table I.

TABLE I
NUMBER OF FLOPS FOR DIFFERENT OPERATIONS

Operation	Number of FLOPs
Conjugate	0
Real addition	1
Real multiplication	1
Complex addition	2
Complex multiplication	6
$\lvert.\rvert^2$	3
Complex-real multiplication	2
Square-root	$6 \sim 8$

Other measures such as time of execution, size of hardware, number of transistor gates and etc. are not considered in this work.

III. RESULTS

All results are obtained by increasing the nonlinear order (with $O > 1$) and memory depth , setting $G = 3$ in the generalized memory polynomial and $D = 2$ in the Volterra with dynamic deviation reduction. In Fig. 1 the average number of FLOPs per kernel (coefficient) is shown.

In some of the models, there may be several ways to generate the same number of coefficients that result in different number of FLOPs. For example a memory polynomial model with $M = 4$ and $O = 3$ results in 15 coefficients and 125 FLOPs while the same model with $M = 2$ and $O = 5$ has 15 coefficients and 127 FLOPs. In this figure, the configuration that yields the lowest amount of FLOPs is consistently used. This does not take into consideration how accurate each configuration can model the power amplifier, and is not the focus of this work.

The minimum number of FLOPs per coefficient is when the basis functions have zero complexity and only the second step of the algorithm has to be evaluated. This is one complex multiplication and one complex addition resulting in 8 FLOPs per coefficient. From Fig.1 it can be seen that the memory polynomial model is very close to

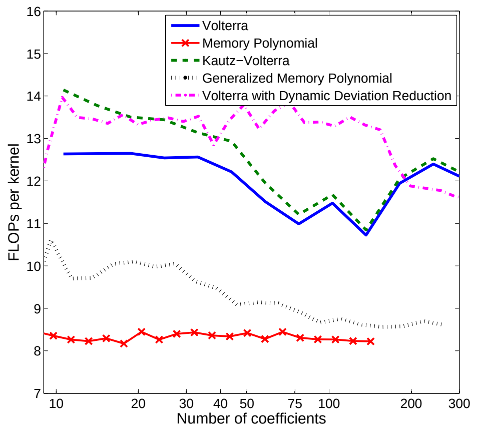

Fig. 1. FLOPs per kernel for different behavioral models.

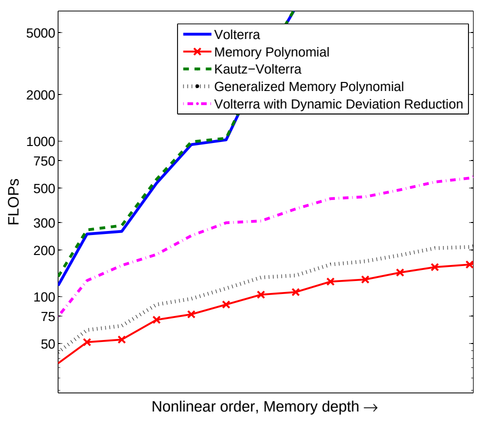

Fig. 2. Total number of FLOPs vs nonlinear order + memory depth for different behavioral models.

this minimum and is the easiest model to construct. The generalized memory polynomial also yields similar results when the number of coefficients increase, but costs slightly higher than the memory polynomial model.

Because Volterra with dynamic deviation reduction has limited basis reusability, it is the most complex model per coefficient initially. This limitation is because there is little possibility of delaying one basis function to generate another since all basis functions contain the term $x(n)$. As the number of coefficients increase with a fixed number of dynamics, the reusability improves and this model becomes slightly better than the Volterra and Kautz-Volterra models. Regarding the latter two models, it can be observed that as the memory increases the amount of reusability of the basis functions also increase, hence the negative slope that can be seen in Fig. 1. However, when the nonlinear order increases, the reusability is not possible in the same way, resulting in the positive slope.

In Fig. 2 the rate of the increase in complexity is shown. The horizontal axis represents the memory depth + nonlinear order and the vertical axis is the total number of FLOPs.

From this figure, it can be noticed that the total complexity for the Kautz-Volterra model is the highest. Comparing to Fig. 1 where Volterra with dynamic deviation reduction was more complex it can be concluded that the higher complexity is due to the exponential growth in parameters for the Kautz-Volterra and Volterra models.

IV. Conclusion

Efficient computer algorithms for hardware implementation of some of the most used polynomial based power amplifier behavioral models have been derived. The implementation was divided into two steps, the first to construct the basis functions and the second to filter the resulting basis functions with the kernels. The second step was similar for the different algorithms, but in the first step some complexity reductions were possible and proposed.

It was shown that the memory polynomial model is the cheapest model to construct while the Volterra and Kautz-Volterra models are the most expensive. Also the Volterra with dynamic deviation model showed more initial complexity per coefficient than the Volterra model due to having more complex basis functions and less reusability.

Acknowledgment

This research has been carried out in the GigaHertz Centre in a joint research project financed by the Swedish Governmental Agency of Innovation Systems (VINNOVA), Chalmers University of Technology, Ericsson AB, Infineon Technologies, and NXP Semiconductors.

References

[1] J. C. Pedro and S. A. Maas, “A comparative overview of microwave and wireless power-amplifier behavioral modeling approaches,” *IEEE Trans. Microw. Theory Tech.*, vol. 53, no. 4, pp. 1150–1163, 2005.

[2] M. Isaksson, D. Wisell, and D. Ronnow, “A comparative analysis of behavioral models for RF power amplifiers,” *IEEE Trans. Microw. Theory Tech.*, vol. 54, no. 1, pp. 348–359, 2006.

[3] V. Mathews and G. L. Sicuranza, *Polynomial Signal Processing*, ser. Wiley Series in Telecommunications and Signal Processing. New York: Wiley, 2000.

[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, 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 Tech.*, vol. 54, no. 12, pp. 4323–4332, 2006.

[6] M. Isaksson and D. Ronnow, “A Kautz-Volterra behavioral model for RF power amplifiers,” in *Proc. IEEE MTT-S Int. Micro. Symp. Dig.*, 2006, pp. 485–488.