# STAVES: Speedy Tensor-Aided Volterra-Based Electronic Simulator

***Abstract*—Volterra series is a powerful tool for blackbox macro-modeling of nonlinear devices. However, the exponential complexity growth in storing and evaluating higher order Volterra kernels has limited so far its employment on complex practical applications. On the other hand, tensors are a higher order generalization of matrices that can naturally and efficiently capture multi-dimensional data. Significant computational savings can often be achieved when the appropriate low-rank tensor decomposition is available. In this paper we exploit a strong link between tensors and frequency-domain Volterra kernels in modeling nonlinear systems. Based on such link we have developed a technique called speedy tensor-aided Volterra-based electronic simulator (STAVES) utilizing high-order Volterra transfer functions for highly accurate time-domain simulation of nonlinear systems. The main computational tools in our approach are the canonical tensor decomposition and the inverse discrete Fourier transform. Examples demonstrate the efficiency of the proposed method in simulating some practical nonlinear circuit structures.**

***Keywords*—*Tensor, tensor decomposition, Volterra series, nonlinear simulation, discrete Fourier transform***

## I. Introduction

The advancement in electronic design automation (EDA) has been crucial in supporting the ever-growing complexity and functionality in modern electronic products. Fast and accurate device and on-chip structure modeling and simulation are at the heart of EDA research. In particular, the analog and radio-frequency modules, though typically occupying a small part of a mixed-signal chip, are critical but difficult to analyze owing to their inherent nonlinearities. An important analytical tool for describing nonlinear systems is the Volterra theory [1], [2] which has been in use for at least more than half a century. In particular, the Volterra series has been successfully applied to modeling and simulating weakly nonlinear systems, e.g., [1], [3], [4]. Some recent work [5] developed a systematic approach to obtain the Volterra series representation from X-parameters [6]–[8] via the harmonic input method. The extracted Volterra transfer functions then allow time-domain simulations to be conducted by the frequency-domain Volterra kernels. This further facilitates the application of Volterra series on blackbox macro-modeling of nonlinear devices and systems. Although the formulation of Volterra series does not preclude itself from modeling strong nonlinearities, the exponential complexity growth in storing and evaluating higher order Volterra kernels and transfer functions results in the so-called "curse of dimensionality" that forbids practical implementation.

On the other hand, if we regard scalars, vectors and matrices as $d$th-order data structures whereby $d = 0, 1, 2$, respectively, then tensors are higher order ($d \geq 3$) counterparts [9]. In fact, many engineering problems and data can be represented intrinsically as multi-dimensional arrays. Using the tensor format to present their inherent structure can often lead to significant savings in computation and storage, thus providing a powerful tool to overcome the curse of dimensionality. The tensor representation and decompositions were proposed and studied almost a century ago [10]. However their potential power was only recognized and successfully demonstrated around the 1980s in chemometrics and psychometrics [11]. In the past decade, their use has further spread into communications, signal processing, numerical linear algebra and big data [12]–[17] etc. Nonetheless, in the EDA field and particularly in nonlinear circuit modeling and simulation, tensors still remain a relatively new technique.

The key contribution of this paper is the proposal and demonstration of the utilization of the canonical tensor decomposition for a highly compact representation of the frequency-domain Volterra kernels. Consequently, despite the order of the kernel or of the nonlinearity, a single off-line decomposition (the computationally expensive step) gives rise to several factor matrices which can then be repeatedly used with the inverse Fourier transform (IFT) (the computationally cheap step) to produce time-domain simulations subject to arbitrary inputs. This procedure is summarized in a routine we named: speedy tensor-aided Volterra-based electronic simulator (STAVES).

We remark that the integration of Volterra theory and tensors has also appeared in previous works on nonlinear simulation [18], [19] and in a different context of system identification [20]. The former works in [18], [19] use tensor decompositions with ad hoc symmetrization of tensor mode factors, while the method in [20] is the reverse process of reconstructing the rank-1 tensors (called outer products) from time-domain observations. It should be noticed that all such previously existent works are time-domain methods. This paper differs in the sense that STAVES allows a genuine use of frequency-domain Volterra transfer functions, which can be easily derived from X-parameters. Indeed, with its elegant formulation and low complexity in representing large nonlinearities, we believe STAVES constitutes a powerful supplement, if not a replacement, to existing nonlinear circuit simulation platforms. We expect further optimization of STAVES, both theoretically and numerically, will come along shortly after this first inception.

This paper is organized as follows. Section II reviews Volterra series and its kernel extraction, as well as the basics of tensors. Section III presents the STAVES algorithm. Numerical examples are given in Section IV, together with some additional remarks. Finally, Section VI draws the conclusion.

## II. Background

### A. Volterra series

Volterra theory has long been used in analyzing communication systems and nonlinear control [1], [2]. Volterra series can be regarded as a Taylor series with memory effects since its evaluation at a particular time point requires input information from the past. Specifically, a nonlinear time-invariant system with an input $u(t)$ and an output $y(t)$ can be expanded into a Volterra series as

$$y(t) = y_1(t) + y_2(t) + y_3(t) + \cdots,$$

where

$$y_n(t) = \int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} h_n(\tau_1, \cdots, \tau_n) \cdot u(t-\tau_1) \cdots u(t-\tau_n) d\bar{\tau}, \quad (1)$$

where $d\bar{\tau} = d\tau_1 \cdots d\tau_n$, and $h_n(\tau_1, \cdots, \tau_n)$ is the $n$th-order Volterra kernel. Due to the commutable product structure of $u(t-\tau_1)\cdots u(t-\tau_n)$ in higher order ($n \geq 2$) terms, the kernel is not unique. However it can be made unique via *symmetrization* by permuting arguments [2], e.g., for $n = 2$, $h_2 \leftarrow (h_2(t_1,t_2) + h_2(t_2,t_1))/2$. We will assume symmetric kernels from now on. Note that $y_1$ is the usual first-order convolution with a frequency-domain input-output relationship $Y_1(\omega) = H_1(\omega)U(\omega)$ where $H_1(\omega) = \int_{-\infty}^{\infty} h_1(\tau) e^{-j\omega\tau} d\tau$ is the impulse response transfer function and $U(\omega) = \int_{-\infty}^{\infty} u(\tau) e^{-j\omega\tau} d\tau$. Analogously, the (nonlinear) higher-order transfer functions (frequency-domain kernels) are defined as

$$H_n(\omega_1, \cdots, \omega_n) = \int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} h_n(\tau_1, \cdots, \tau_n) \cdot e^{-j\omega_1\tau_1} \cdots e^{-j\omega_n\tau_n} d\bar{\tau}. \quad (2)$$

Nonetheless, unlike the first-order case, there is no direct counterpart in the (multivariate) frequency domain with respect to a (univariate) input $U(\omega)$. Instead, we need to replace the single time axis in the product of $u$ in (1) by multiple axes as $u(t_1-\tau_1)\cdots u(t_n-\tau_n)$, yielding [2]

$$Y_n(\omega_1, \cdots, \omega_n) = H_n(\omega_1, \cdots, \omega_n) U(\omega_1) \cdots U(\omega_n). \quad (3)$$

The multi-dimensional IFT of (3) is a generalization of the univariate formula, given by

$$\begin{aligned} y_n(t_1, \cdots, t_n) &= \mathscr{F}_{(n)}^{-1}\left(Y_n(\omega_1, \cdots, \omega_n)\right) \\ &= \frac{1}{(2\pi)^n} \int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} Y_n(\omega_1, \cdots, \omega_n) \cdot e^{j\omega_1 t_1} \cdots e^{j\omega_n t_n} d\omega_1 \cdots d\omega_n. \end{aligned} \quad (4)$$

To restore the required $y_n(t)$ which is the $n$th-order response of the circuit subject to $u(t)$, one then evaluates along the *diagonal line* in the multi-time hyperplane, i.e.,

$$y_n(t) = y_n(t_1, \cdots, t_n)|_{t_1=t_2=\cdots=t_n=t}. \quad (5)$$

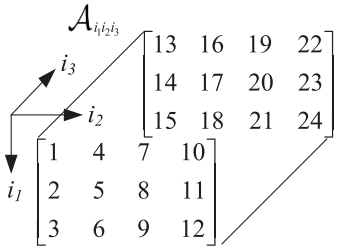


Fig. 1. An example tensor $\mathcal{A} \in \mathbb{R}^{3\times4\times2}$.

### B. Volterra kernel extraction from X-parameters

Without knowing the state equation, it is difficult to determine the higher order Volterra kernels $h_n$ or transfer functions $H_n$ in nature. Fortunately, the recently developed X-parameter method [5] provides a systematic way to obtain the Volterra transfer functions from X-parameters. X-parameters are a superset of S-parameters [6]–[8]. They describe the relationships between incident and scattered waves by using not only port-to-port but also harmonic-to-harmonic interactions under certain large signal operating points.

To obtain a complete description of the Volterra kernels $H_n(\omega_1, \omega_2, \ldots, \omega_n)$, a frequency sweep along the axes $\omega_1$, $\omega_2$, $\ldots$, $\omega_n$ in the interested region is required when generating the X-parameters. Suppose the number of frequency sweeping points along each frequency axis is $m$, the equivalent number of frequency sweeping points of $H_n$ is $nm$ thanks to the symmetry properties of the kernels. By matching the X-parameters with the corresponding Volterra kernels at each frequency point, the discrete frequency-domain Volterra kernels $H_n$ can be obtained.

Given $H_n$ as the blackbox model of the nonlinear system, its output response can be computed via (3) and (4), with complexities $O(m^n)$ and $O(nm^n \log m)$ if the fast Fourier transform (FFT) is deployed in (4). However, it is readily seen that the exponential growth of the computational complexity and memory requirement forbids its application on highly nonlinear systems.

### C. Tensors

We use a calligraphic font, e.g., $\mathcal{A}$, to denote a tensor. A $d$th-order (or $d$-way) tensor, first assumed real for ease of illustration, is a multi-way array $\mathcal{A} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d}$ generalizing the matrix format to its $d$th-order counterpart [9], wherein the $n_i$'s are called the dimensions. An example third-order tensor is shown in Fig. 1.

A $d$th-order rank-1 tensor can be written as the outer product of $d$ vectors

$$\mathcal{A} = a^{(1)} \circ a^{(2)} \circ \cdots \circ a^{(d)}, \; a^{(k)} \in \mathbb{R}^{n_k}, \quad (6)$$

where $\circ$ is the outer product. Its element $\mathcal{A}_{i_1 i_2 \cdots i_d} = a^{(1)}_{i_1} a^{(2)}_{i_2} \cdots a^{(d)}_{i_d}$, where $a^{(k)}_{i_k}$ is the $i_k$th entry of vector $a^{(k)}$.

The CANDECOMP/PARAFAC (CP) decomposition[1] [9],

[1] CANDECOMP (canonical decomposition) by Carroll and Chang [11] and PARAFAC (parallel factors) by Harshman [21].

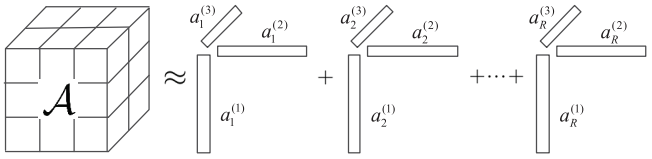


Fig. 2. A CP decomposition of a third-order tensor.

[11], [15], [21] approximates a tensor $\mathcal{A}$ by a finite sum of rank-1 tensors, which can be written by

$$\mathcal{A} \approx \sum_{j=1}^{R} a_j^{(1)} \circ a_j^{(2)} \circ \cdots \circ a_j^{(d)},\ a_j^{(k)} \in \mathbb{R}^{n_k}, \tag{7}$$

where $R \in \mathbb{Z}^+$ is called the rank of the approximation. For ease of notation, people usually express the CP (7) by $\mathcal{A} \approx [\![\mathbf{A}^{(1)}, \ldots, \mathbf{A}^{(d)}]\!]$, where the factor matrices are defined by $\mathbf{A}^{(k)} \triangleq [a_1^{(k)}, a_2^{(k)}, \ldots, a_R^{(k)}] \in \mathbb{R}^{n_k \times R}$. Fig. 2 illustrates a rank-$R$ approximation of a third-order tensor. Several methods have been developed to compute the CP decomposition, such as the alternating least squares [11], [21] method or the optimization method CPOPT [22]. For more details about different CP methods, readers may refer to [9].

## III. STAVES

In this section we present the main STAVES algorithm. First of all, we obtain the X-parameter data generated by the harmonic-balance simulation (e.g., the ADS X-parameter generation platform [23]) or measured by any modern nonlinear vector network analyzers. Next, the discrete Volterra transfer functions $H_n$ can be found using the extraction method [5] introduced in Section II-B.

It is readily seen that the discrete $H_n$ and $Y_n$ in (2) and (3) are naturally $n$th-order tensors

$$\mathcal{H}_n,\ \mathcal{Y}_n \in \mathbb{C}^{\overbrace{m \times \cdots \times m}^{n}}, \tag{8}$$

with each element $(\mathcal{H}_n)_{m_1 \cdots m_n} = H_n(\omega_{m_1}, \ldots, \omega_{m_n})$, $(\mathcal{Y}_n)_{m_1 \cdots m_n} = Y_n(\omega_{m_1}, \ldots, \omega_{m_n})$ and $m$ be the number of sampling points along each frequency axis. Consequently, (3) becomes

$$\mathcal{Y}_n = \mathcal{H}_n \odot (U \circ \cdots \circ U), \tag{9}$$

where $\odot$ denotes the Hardmard product between tensors. It is necessary to separate the real and imaginary parts of $\mathcal{H}_n$ such that we end up with two real tensors $\mathrm{Re}(\mathcal{H}_n)$ and $\mathrm{Im}(\mathcal{H}_n)$ via $\mathcal{H}_n = \mathrm{Re}(\mathcal{H}_n) + j\,\mathrm{Im}(\mathcal{H}_n)$.

The key to STAVES is to decompose high-order $\mathrm{Re}(\mathcal{H}_n)$ and $\mathrm{Im}(\mathcal{H}_n)$ by CP for $n > 2$. We can always use certain ranks $R_{\text{real},n}$ and $R_{\text{imag},n}$ to approximate $\mathrm{Re}(\mathcal{H}_n)$ and $\mathrm{Im}(\mathcal{H}_n)$ by

$$\begin{aligned}
\mathrm{Re}(\mathcal{H}_n) &\approx [\![\mathbf{RH}_n^{(1)}, \ldots, \mathbf{RH}_n^{(n)}]\!] = \sum_{j=1}^{R_{\text{real},n}} h_{r,j}^{(1)} \circ \cdots \circ h_{r,j}^{(n)}, \\
\mathrm{Im}(\mathcal{H}_n) &\approx [\![\mathbf{IH}_n^{(1)}, \ldots, \mathbf{IH}_n^{(n)}]\!] = \sum_{j=1}^{R_{\text{imag},n}} h_{i,j}^{(1)} \circ \cdots \circ h_{i,j}^{(n)},
\end{aligned} \tag{10}$$

where $h_{r,j}^{(k)}, h_{i,j}^{(k)} \in \mathbb{R}^m$, $\mathbf{RH}_n^{(k)} = [h_{r,1}^{(k)}, \ldots, h_{r,R_{\text{real},n}}^{(k)}] \in \mathbb{R}^{m \times R_{\text{real},n}}$ and $\mathbf{IH}_n^{(k)} = [h_{i,1}^{(k)}, \ldots, h_{i,R_{\text{imag},n}}^{(k)}] \in \mathbb{R}^{m \times R_{\text{imag},n}}$ are the factor matrices of $\mathrm{Re}(\mathcal{H}_n)$ and $\mathrm{Im}(\mathcal{H}_n)$, respectively. Using the CP structure (10), the Hardmard product in (9) simply becomes

$$\begin{aligned}
\mathcal{Y}_n = {} & [\![\mathbf{RH}_n^{(1)} \odot U, \ldots, \mathbf{RH}_n^{(n)} \odot U]\!] \\
& + j[\![\mathbf{IH}_n^{(1)} \odot U, \ldots, \mathbf{IH}_n^{(n)} \odot U]\!].
\end{aligned} \tag{11}$$

Furthermore, the multi-dimensional IFT in (4) reads

$$\begin{aligned}
\mathcal{F}_{(n)}^{-1}(\mathcal{Y}_n) = {} & \mathcal{F}_{(n)}^{-1}([\![\mathbf{RH}_n^{(1)} \odot U, \ldots, \mathbf{RH}_n^{(n)} \odot U]\!]) \\
& + j\mathcal{F}_{(n)}^{-1}([\![\mathbf{IH}_n^{(1)} \odot U, \ldots, \mathbf{IH}_n^{(n)} \odot U]\!]) \\
= {} & [\![\mathcal{F}_{(1)}^{-1}(\mathbf{RH}_n^{(1)} \odot U), \ldots, \mathcal{F}_{(1)}^{-1}(\mathbf{RH}_n^{(n)} \odot U)]\!] \\
& + j[\![\mathcal{F}_{(1)}^{-1}(\mathbf{IH}_n^{(1)} \odot U), \ldots, \mathcal{F}_{(1)}^{-1}(\mathbf{IH}_n^{(1)} \odot U)]\!],
\end{aligned} \tag{12}$$

where $\mathcal{F}_{(n)}^{-1}$ denotes the $n$-dimensional inverse discrete Fourier transform (IDFT) and $\mathcal{F}_{(1)}^{-1}(\mathbf{H})$ represents the single-mode IDFT on *each column* of the matrix $\mathbf{H}$.

Moreover, the diagonal elements of $y_n(t_1, \ldots, t_n)$ in (5) (i.e., setting $t_1 = \cdots = t_n = t$) can be easily read off by reconstructing only the diagonal elements of (12) and the elements are given by

$$\begin{aligned}
(y_n)_k = {} & \sum_{j=1}^{R_{\text{real},n}} \prod_{l=1}^{n} \left(\mathcal{F}_{(1)}^{-1}(h_{r,j}^{(l)} \odot U)\right)_k \\
& + j \sum_{j=1}^{R_{\text{imag},n}} \prod_{l=1}^{n} \left(\mathcal{F}_{(1)}^{-1}(h_{i,j}^{(l)} \odot U)\right)_k,
\end{aligned} \tag{13}$$

where $(\cdot)_k$ retrieves the $k$th element of the vector, for $k = 1, \ldots, m$. STAVES is summarized in Algorithm 1.

In STAVES, it is not necessary to perform the multi-dimensional IDFT, neither to rebuild the whole tensor with $m^n$ entries. The most expensive step in STAVES will be calculating the CP decompositions for $\mathrm{Re}(\mathcal{H}_n)$ and $\mathrm{Im}(\mathcal{H}_n)$. However, it is worth pointing out that these two decompositions (steps 5-6 in Algorithm 1) can be pre-computed once "offline" and then reused for different on-going simulations. For a specific $u \in \mathbb{R}^m$, the complexity of evaluating $y_n$ by steps 7-9 in Algorithm 1 is dominated by the single-mode IDFT of $n(R_{\text{real},n} + R_{\text{imag},n})$ vectors with length $m$. Therefore the complexity will be in $O((R_{\text{real},n} + R_{\text{imag},n})nm \log m)$, in contrast with the original complexity $O(nm^n \log m)$ for (4).

In addition, STAVES requires the storage of only the factor matrices in memory, with space complexity $O((R_{\text{real},n} +$

**Algorithm 1** STAVES Algorithm

**Input:** $\mathcal{H}_1, \mathcal{H}_2, \ldots, \mathcal{H}_n, u$
**Output:** $y$
1: $U = \mathcal{F}_{(1)}(u)$;
2: $y_1 = \mathcal{F}_{(1)}^{-1}(\mathcal{H}_1 \odot U)$;
3: $y_2 = \mathcal{F}_{(2)}^{-1}(\mathcal{H}_2 \odot (U \circ U))$;
4: **for** $i = 3$ **to** $n$ **do**
5: $[\![\mathbf{RH}_i^{(1)}, \ldots, \mathbf{RH}_i^{(i)}]\!] = \mathrm{CP}(\mathrm{Re}(\mathcal{H}_i))$;
6: $[\![\mathbf{IH}_i^{(1)}, \ldots, \mathbf{IH}_i^{(i)}]\!] = \mathrm{CP}(\mathrm{Im}(\mathcal{H}_i))$;
7: $\mathbf{Y}_r = \mathcal{F}_{(1)}^{-1}(\mathbf{RH}_i^{(1)} \odot U) \odot \cdots \odot \mathcal{F}_{(1)}^{-1}(\mathbf{RH}_i^{(i)} \odot U)$;
8: $\mathbf{Y}_i = \mathcal{F}_{(1)}^{-1}(\mathbf{IH}_i^{(1)} \odot U) \odot \cdots \odot \mathcal{F}_{(1)}^{-1}(\mathbf{IH}_i^{(i)} \odot U)$;
9: $y_i = \mathrm{sum}(\mathbf{Y}_r, 2) + j \cdot \mathrm{sum}(\mathbf{Y}_i, 2)$;
{sum(·,2) represents the sum over each column vector of a matrix.}
10: **end for**
11: $y = y_1 + y_2 + \cdots + y_n$;

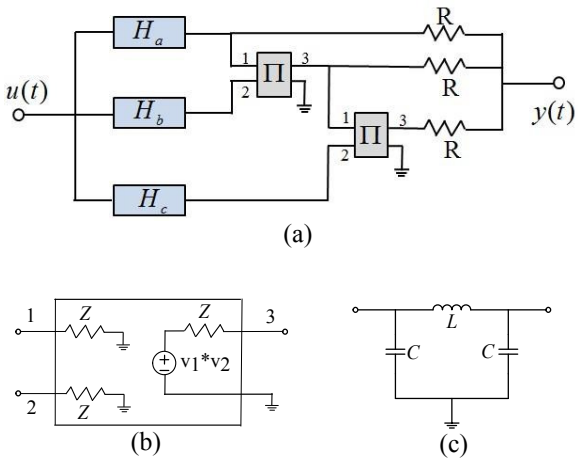


Fig. 3. (a) System diagram of a third-order mixer circuit. The symbol Π denotes a mixer. (b) The equivalent circuit of the mixer. $Z = R = 50\,\Omega$. (c) The circuit schematic diagram of the low-pass filters $H_a$, $H_b$ and $H_c$, with $L = 42.52\,\mathrm{nH}$ and $C = 8.5\,\mathrm{pF}$.

$R_{\mathrm{imag},n})nm)$, while $O(m^n)$ memory is required for the conventional approach.

## IV. Numerical Examples

We apply STAVES to several practical circuit design examples in order to verify its validity and demonstrate its efficiency compared to the traditional approach via the multi-dimensional IDFT. The algorithm is implemented in MATLAB and run on a desktop with an Intel i5 3.3GHz CPU and 16GB RAM. The CP decomposition in STAVES is computed by the CPOPT algorithm provided in the Tensor Toolbox [15], [22].

### A. Example 1: A third-order nonlinear mixer system

The first example is a third-order nonlinear system with two mixers. Fig. 3(a) shows the system diagram. The circuit structure of the mixers is in Fig. 3(b), with V-I relationship at port 3 given by $i_3(t) = [v_1(t)v_2(t) - v_3(t)]/Z$, where $v_j$ and $i_j$ are the voltage and current at port $j$ respectively. $H_a$, $H_b$ and $H_c$ are identical linear low-pass filters whose scheme shown in Fig. 3(c).

TABLE I. Frequency sweep scheme for the 3-tone X-parameters generator

| Frequency | Start | Step | Stop |
|---|---|---|---|
| $\omega_1$ | 7 MHz | 120 MHz | 2.047 GHz |
| $\omega_2$ | 41 MHz | 120 MHz | 2.081 GHz |
| $\omega_3$ | 87 MHz | 120 MHz | 2.127 GHz |

Volterra transfer functions of the nonlinear system are extracted from the X-parameters by using the method presented in II-B. The 3-tone X-parameters of the nonlinear system are generated by applying the ADS X-parameters generator [23]. The frequency sweep scheme of each axis is given in Table I. The number of sampling points $m$ is 201, therefore we obtain the transfer functions $\mathcal{H}_1 \in \mathbb{C}^{201}$, $\mathcal{H}_2 \in \mathbb{C}^{201\times 201}$, $\mathcal{H}_3 \in \mathbb{C}^{201\times 201\times 201}$, etc.

We focus on computing the third-order response $y_3$, as the procedures of calculating $y_1$ and $y_2$ are the same in both approaches. Equations (3)-(5) are used by the traditional approach to generate the reference result $y_{3,\mathrm{ref}}$, while the output of our STAVES is denoted by $y_{3,\mathrm{STAVES}}$. The relative error of STAVES is calculated by the normalized mean-square error of its time-domain waveform as $\|y_{3,\mathrm{STAVES}} - y_{3,\mathrm{ref}}\|_2 / \|y_{3,\mathrm{ref}}\|_2$. Meanwhile, the speedup of STAVES is defined as $t_{\mathrm{ref}}/t_{\mathrm{STAVES}}$, where $t_{\mathrm{ref}}$ is the run time for executing (3)-(5) and $t_{\mathrm{STAVES}}$ is the run time for STAVES to compute steps 7-9 in Algorithm 1.

The system is fed by a rectangular pulse input with raise and fall time $t_r = t_f = 1\,\mathrm{ns}$, width $t_w = 5\,\mathrm{ns}$, and magnitude $V_0 = 1\,\mathrm{V}$. We sweep the tensor ranks $R_{\mathrm{real},3} = R_{\mathrm{imag},3} = R$ for $R = 1, \ldots, 100$ to illustrate the performance of STAVES. Fig. 4(a) demonstrates the time-domain responses of $y_3$ when different ranks are applied, comparing with the reference curve. A fairly good match is observed when the rank $R \geq 20$. On the other hand, the relative error and speedup of STAVES for different ranks are plotted in Fig. 4(b) and 4(c), respectively. From these figures, a certain trade-off between the accuracy and efficiency of STAVES can be discovered, though, a 60x speedup is still achievable for $R$ around 100 with a 0.6% error.

### B. Example 2: A low-noise amplifier

The second numerical example is a low-noise amplifier. The schematic is obtained from the *X-parameter generation tutorial* of the example directory of ADS [23]. Again, the 3-tone X-parameters are generated with the same frequency sweep scheme displayed in Table I and the Volterra transfer functions are extracted. In this case, $m$ is also 201 and we use the same pulse excitation as the previous one.

This time, we sweep the tensor ranks $R_{\mathrm{real},3} = R_{\mathrm{imag},3} = R$ for $R = 1, \ldots, 200$. The time-domain waveforms of the first 6 ns are plotted in Fig. 5(a). It is shown that STAVES accurately captures the third-order response when $R \geq 40$. The error plotted in Fig. 5(b) indicates that a larger rank is needed to achieve a similar accuracy compared to the previous circuit,

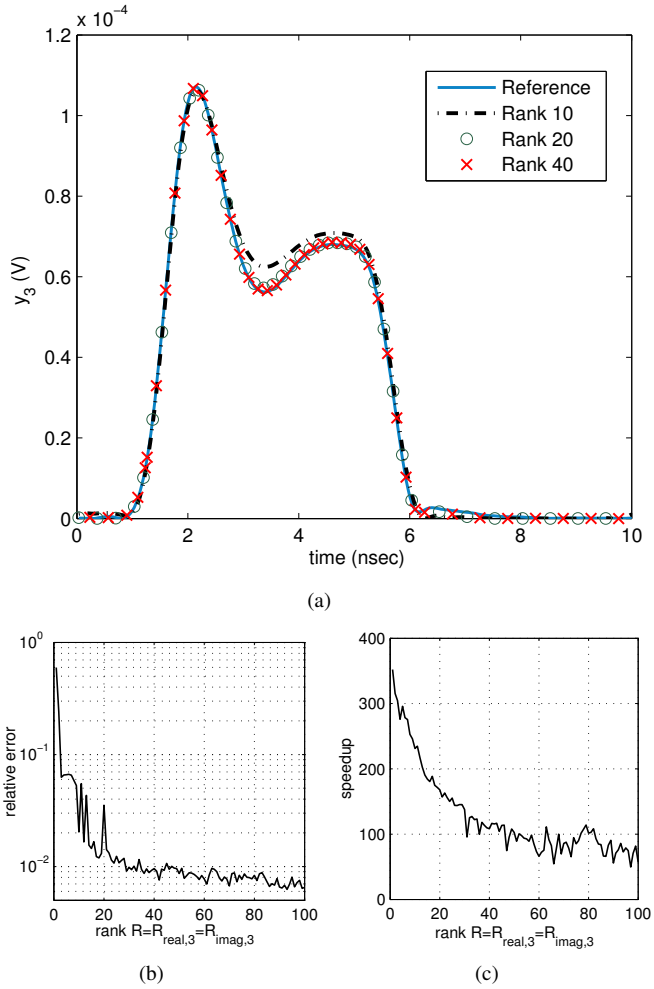


Fig. 4. (a) Time-domain results of STAVES with different rank approximations for the mixer system. (b) Relative errors of STAVES with different ranks. (c) Speedups brought by STAVE with different ranks.

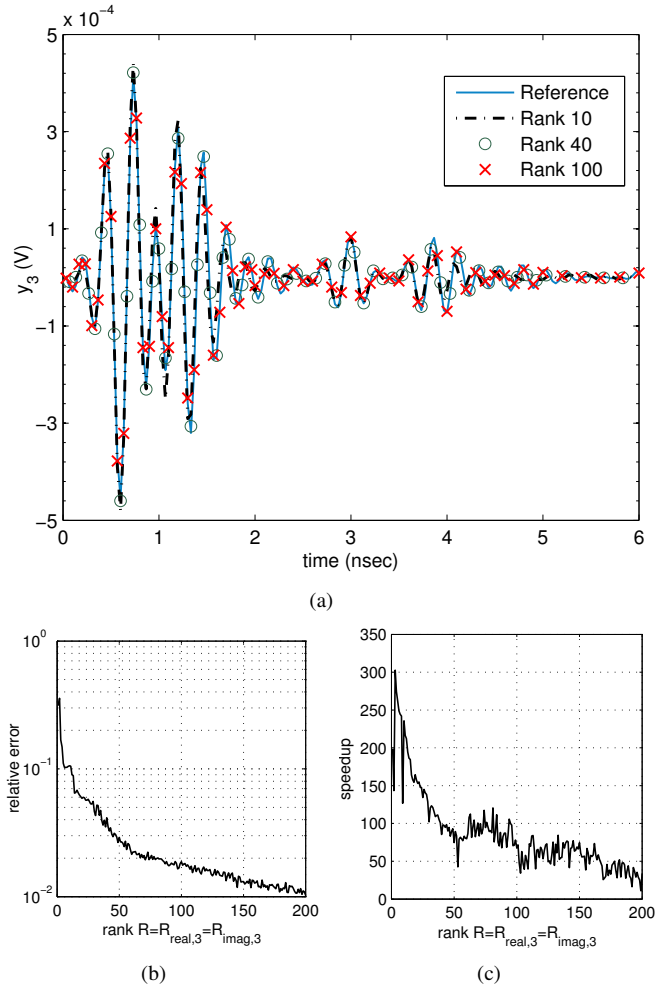


Fig. 5. (a) Time-domain results of STAVES with different rank approximations for the low noise amplifier. (b) Relative errors of STAVES with different ranks. (c) Speedups brought by STAVE with different ranks.

due to the fast changing rate of $y_3$ in this example. Meanwhile, Fig. 5(c) shows that simulating an $R = 200$ model of STAVES is still 10x faster than using the traditional approach. It further confirms that STAVES can always accelerate the simulation if the ranks are less than $O(m)$.

## V. Remarks

Accurate low-rank CP approximations of $\mathrm{Re}(\mathcal{H}_n)$ and $\mathrm{Im}(\mathcal{H}_n)$ are critical to the performance of STAVES. However, to the best of our knowledge, there is no feasible way to determine the actual rank of a specific tensor, nor is there an empirical method to prescribe the rank before the CP decomposition. Nonetheless, as it is shown in Section IV, practical examples demonstrate that ranks $R_{\mathrm{real},n}$ and $R_{\mathrm{imag},n}$ less than $O(m)$ are usually enough for fast and accurate simulations.

A major merit of STAVES is that the factor matrices for a particular device or on-chip structure are computed only once, and can then be repeatedly used for all types of input, while requiring only a linear complexity cost. That is, the most expensive step is performed only once, with all subsequent simulations being very cheap.

Moreover, a device or on-chip structure can be fully characterized by the mode factors associated with each order of its Volterra kernel without any knowledge of the actual schematic or topology. This allows accurate third party simulation while protecting IP of the circuit layout.

## VI. Conclusion

This paper has described a natural connection between tensor arithmetic and Volterra theory. Such connection allows fast and accurate nonlinear circuit simulation. The key innovation was to cast the frequency-domain Volterra kernels into tensors and to obtain their rank-1 factors. High-order convolutions in the time domain are then drastically reduced to single-mode inverse discrete Fourier transform (IDFT) despite the order of the kernel. The expensive tensor decomposition is done only once whereas all subsequent simulations require

only cheap single-mode IDFT for which highly optimized routines abound. This algorithm, called speedy tensor-aided Volterra-based electronic simulator (STAVES), has been verified through design examples achieving remarkable speedups compared to the traditional routine.

## REFERENCES


[1] E. Bedrosian and S. O. Rice, "The output properties of Volterra systems (nonlinear systems with memory) driven by harmonic and Gaussian inputs," *Proc. IEEE*, vol. 59, no. 12, pp. 1688–1707, Dec. 1971.

[2] W. Rugh, *Nonlinear System Theory – The Volterra-Wiener Approach*. Baltimore, MD: Johns Hopkins Univ. Press, 1981.

[3] J. R. Phillips, "Projection-based approaches for model reduction of weakly nonlinear, time-varying systems," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 22, no. 2, pp. 171–187, Feb. 2003.

[4] P. Li and L. Pileggi, "Compact reduced-order modeling of weakly nonlinear analog and RF circuits," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 23, no. 2, pp. 184–203, Feb. 2005.

[5] X. Y. Z. Xiong, L. J. Jiang, J. Shutt-Aine, and W. C. Chew, "Blackbox macro-modeling of the nonlinearity based on Volterra series representation of X-parameters," in *IEEE Conf. Electrical Performance of Electronic Packaging and Systems (EPEPS)*, Oct. 2014, pp. 85–88.

[6] D. Root, J. Verspecht, D. Sharrit, J. Wood, and A. Cognata, "Broadband poly-harmonic distortion (PHD) behavioral models from fast automated simulations and large-signal vectorial network measurements," *Microwave Theory and Techniques, IEEE Transactions on*, vol. 53, no. 11, pp. 3656–3664, Nov 2005.

[7] J. Schutt-Aine, P. Milosevic, and W. Beyene, "Modeling and simulation of high speed I/O links using X parameters," in *IEEE Conf. Electrical Performance of Electronic Packaging and Systems (EPEPS)*, Oct 2010, pp. 29–32.

[8] D. E. Root, J. Verspecht, J. Horn, and M. Marcu, *X-Parameters: Characterization, Modeling, and Design of Nonlinear RF and Microwave Components*. Cambridge University Press, 2013.

[9] T. Kolda and B. Bader, "Tensor decompositions and applications," *SIAM Review*, vol. 51, no. 3, pp. 455–500, 2009.

[10] F. L. Hitchcock, "The expression of a tensor or a polyadic as a sum of products," *Journal of Mathematics and Physics*, vol. 6, pp. 164–189, 1927.

[11] J. D. Carroll and J. J. Chang, "Analysis of individual differences in multidimensional scaling via an n-way generalization of "Eckart-Young" decomposition," *Psychometrika*, vol. 35, no. 3, pp. 283–319, 1970.

[12] L. D. Lathauwer, B. D. Moor, and J. Vandewalle, "A multilinear singular value decomposition," *SIAM Journal on Matrix Analysis and Applications*, vol. 21, no. 4, pp. 1253–1278, 2000.

[13] ——, "On the best rank-1 and rank-($R_1$, $R_2$,..., $R_n$) approximation of higher-order tensors," *SIAM Journal on Matrix Analysis and Applications*, vol. 21, no. 4, pp. 1324–1342, 2000.

[14] E. Kofidis and P. A. Regalia, "On the best rank-1 approximation of higher-order supersymmetric tensors," *SIAM Journal on Matrix Analysis and Applications*, vol. 23, no. 3, pp. 863–884, 2002.

[15] B. W. Bader and T. G. Kolda, "Algorithm 862: MATLAB tensor classes for fast algorithm prototyping," *ACM Transactions on Mathematical Software*, vol. 32, no. 4, pp. 635–653, Dec. 2006.

[16] J. M. Landsberg, *Tensors: Geometry and Applications*. American Mathematical Society, 2012.

[17] N. Vervliet, O. Debals, L. Sorber, and L. D. Lathauwer, "Breaking the curse of dimensionality using decompositions of incomplete tensors: Tensor-based scientific computing in big data analysis," *IEEE Signal Process. Mag.*, vol. 31, no. 5, pp. 71–79, Sep. 2014.

[18] R. Boyer, R. Badeau, and G. Favier, "Fast orthogonal decomposition of Volterra cubic kernels using oblique unfolding," in *IEEE Intl. Conf. Acoustics, Speech and Signal Processing (ICASSP)*, May 2011, pp. 4080–4083.

[19] G. Favier and T. Bouilloc, "Parametric complexity reduction of Volterra models using tensor decompositions," in *17th European Signal Processing Conference (EUSIPCO)*, Aug 2009.

[20] G. Favier, A. Y. Kibangou, and T. Bouilloc, "Nonlinear system modeling and identification using Volterra-PARAFAC models," *Int. J. Adapt. Control Signal Process.*, vol. 26, no. 1, pp. 30–53, Jan. 2012.

[21] R. A. Harshman, "Foundations of the PARAFAC procedure: Models and conditions for an "explanatory" multi-modal factor analysis," *UCLA Working Papers in Phonetics*, vol. 16, no. 1, p. 84, 1970.

[22] E. Acar, D. M. Dunlavy, and T. G. Kolda, "A scalable optimization approach for fitting canonical tensor decompositions," *Journal of Chemometrics*, vol. 25, no. 2, pp. 67–86, February 2011.

[23] Agilent Technologies, *Agilent Advanced Design System*, 1983–2011.