AV: Autonomous Volterra Algorithm for Steady-State Analysis of Nonlinear Circuits

Haotian Liu, *Student Member, IEEE,* and Ngai Wong, *Member, IEEE*

***Abstract*—We present a novel algorithm, named autonomous Volterra (AV), that achieves efficient steady-state analysis of nonlinear circuits. With elegant analytic forms and availability of efficient solvers, AV constitutes a competitive steady-state algorithm besides the two mainstreams, namely, shooting Newton (SN) and harmonic balance (HB). Nonlinear systems are first captured in nonlinear differential algebraic equations (DAEs), followed by expansion into linear Volterra subsystems. A key step of steady-state analysis lies in modeling each Volterra subsystem with autonomous nonlinear inputs. The steady-state solution of these subsystems then proceeds with a series of Sylvester equation solves, completely avoiding the guesses of initial condition and time stepping as in SN, as well as the uncertain length of Fourier series as in HB. Error control in AV is also straightforward by monitoring the norms of the Sylvester equation solutions. We further demonstrate that AV is readily parallelizable with superior scalability towards large-scale problems.**

***Index Terms*—Steady-state analysis, Volterra, autonomous system, nonlinear circuit simulation, QLDAE**

I. Introduction

There is a constant quest for numerically sound and efficient nonlinear modeling techniques for simulating the behaviors of critical analog or radio-frequency (RF) blocks in modern mixed-signal chips [1]–[3]. These blocks are intrinsically nonlinear and their accurate modeling and simulation have always been a challenging task, in contrast to linear subsystems (e.g., interconnects) whereby mature linear time-invariant (LTI) or linear time-varying (LTV) techniques abound, e.g., [4]–[6]. In analog/RF simulation such as for mixers and switched-capacitor filters, traditional time-domain SPICE simulation often falls short due to the drastic difference in input signal frequencies and thereby the prohibitive time steps required for accurate (high-resolution) steady-state or transient analyses [7], [8]. The two mainstream specialized algorithms for (quasi-)periodic steady-state (PSS) analysis are, namely, shooting Newton (SN) and harmonic balance (HB) [2], [9]–[12]. (With "quasi-periodic" we mean a response is similar to a periodic function but does not have a true period, which may happen in multi-tone RF simulation, but henceforth we will simply use "periodic" to refer to both periodic and quasi-periodic cases.)

This work was supported in part by the Hong Kong Research Grants Council under Project HKU 718711E and the University Research Committee of The University of Hong Kong.

The authors are with the Department of Electrical and Electronic Engineering, The University of Hong Kong (email: {htliu, nwong}@eee.hku.hk).

The PSS problem of nonlinear circuits with a periodic input indicates that the specific solution must satisfy the periodic boundary condition in a certain time interval. To solve the nonlinear equation with a boundary condition, SN first discretizes the nonlinear equation along the time axis using, e.g., the backward-Euler method. Then, the final value at the end of one period is regarded as a function of the initial value. Therefore, a higher level Newton-Raphson method can be applied to solve the nonlinear equation for aligning the initial and final values [9], [11]. In short, SN is a time-domain numerical method that iteratively solves the time-domain transient response in one period. Assuming that the number of unknown variables is n and there are M overall time steps, the computational complexity of an ordinary transient analysis in one iteration of SN is in the order of $\mathcal{O}(Mn^2)$. If the system matrices are sparse, the power of n in the complexity can be reduced to a number between 1 and 2 by sparse matrix solvers. And the complexity of solving the update equation is $\mathcal{O}(n^2)$ per iteration if iterative solvers are employed, which can be further reduced to that of ordinary transient analysis by matrix-free method [11]. In strongly nonlinear RF circuits or wideband multi-tone simulation, a large amount of time steps are necessary in each iteration, leading to huge computational load. Furthermore, convergence problems may arise if the time constant of the circuit is much larger than the steady-state period [13]. Recently, parallel methods of SN have been developed to speed up the iterative solver of the update matrices using GPU parallelization [14], but the SN method still depends on the number of time steps since the transient analysis cannot be omitted.

Alternatively, HB is a frequency-domain method that matches the steady state of the original system with frequency harmonics. It approximates the steady-state response with a finite-length Fourier series [2], [15], [16]. Then, its coefficients are also solved by Newton-Raphson method. Employing fast Fourier transform (FFT) [10], and denoting the length of Fourier series by M, the computational cost is $\mathcal{O}(nM \log M)$ per iteration. Nonetheless, when the system is strongly nonlinear it still requires a large number of harmonics to be computed calling for expensive computation. Furthermore, the number of required harmonics is usually hard to predict beforehand.

In this paper, a new autonomous Volterra (AV) approach is proposed for the PSS analysis of nonlinear systems. On the one hand, the Volterra theory [17] provides a systematic approach to modeling nonlinear circuits (e.g., [1], [18]) whereby the Volterra series is truncated at some high-order convolution terms to approximate the behavior of (weak) nonlinearity, similar in concept to a truncated Taylor series in modeling

local behavior. However, the model size and complexity also grow with the order of Volterra kernels, thus restricting the use of the Volterra framework to weakly nonlinear models. Fortunately, the recent introduction of quadratic-linear differential algebraic equation (QLDAE) [19], [20] has enabled the capture of some common electrical (strong) nonlinearities with only low-order Volterra kernels. This is done at the expense of extra state variables to keep the strongly nonlinear functions, say sine/cosine and exponential functions etc., only in quadratic-linear format instead of cubic or higher-order terms. On the other hand, by treating the periodic input to the nonlinear system as coming from a coupled autonomous system, AV computes the steady-state response of each order of Volterra subsystem via the solution of a Sylvester equation, without any time discretization or initial value guess (as in SN) or uncertain Fourier series length (as in HB). Consequently, different from SN and HB which require solving nonlinear systems of equations via iterative schemes and thereby some initial guesses, AV represents a control-theoretic, closed-form and initial-condition-free approach to PSS analysis. Moreover, AV naturally benefits from parallelization through block diagonalization and decomposition of the Sylvester equations. This translates into essentially linear complexity with respect to the order of the Volterra kernels used, making it exceptionally scalable in large-size problems.

We remark that such autonomous "signal generator" concept (incorporating the periodic input signal as part of the linear and nonlinear closed system) is not new and has appeared in the context of linear/nonlinear moment matching and control-theoretic steady-state analysis [21]–[26]. Nonetheless, to the authors' knowledge, its connection with Volterra modeling and the arising elegant Sylvester equation-based expressions and practical, including parallelized, implementations are completely new and proposed for the first time in the literature.

The rest of this paper is organized as follows. Section II reviews the backgrounds of the autonomous linear system and QLDAE. The AV method is proposed in Section III, with a discussion on parallelized implementation. Numerical examples are given in Section IV, followed by several important remarks in Section V. Finally, Section VI concludes this paper.

II. Background and Preliminaries

A. Linear system with an autonomous input

For the ease of illustration, we study a single-input LTI state space

$$\dot{x} = Gx + Bu, \tag{1}$$

where $x \in \mathbb{R}^n$ and $u \in \mathbb{R}$ are the state vector and input signal, respectively. A sinusoidal input $u = a\cos(\omega t)$ can be modeled as the output of a 2nd-order autonomous system (i.e., a system without external inputs) which is also referred to as a signal generator in, e.g., [24]–[26]. In particular, defining $v = [v_1 \ v_2]^T \in \mathbb{R}^2$ and $\omega \in \mathbb{R}$, we set up the autonomous system

$$\dot{v} = Sv \text{ where } S = \begin{bmatrix} 0 & \omega \\ -\omega & 0 \end{bmatrix} v \ , \ v(0) = \begin{bmatrix} 1 \\ 0 \end{bmatrix}. \tag{2}$$

The system matrix S has the eigendecomposition

$$SQ = Q\Lambda \text{ where } \Lambda = \begin{bmatrix} j\omega & 0 \\ 0 & -j\omega \end{bmatrix}, \ Q = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & j \\ j & 1 \end{bmatrix}. \tag{3}$$

Recognizing Q is unitary ($Q^*Q = QQ^* = I$), the closed-form solution $v(t) = e^{St}v(0)$ to (2) is easily found to be

$$v(t) = \begin{bmatrix} \cos(\omega t) & \sin(\omega t) \\ -\sin(\omega t) & \cos(\omega t) \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} \cos(\omega t) \\ -\sin(\omega t) \end{bmatrix}. \tag{4}$$

Subsequently, $u = a\cos(\omega t)$ can be taken as the output of (2) or (4) as

$$u = Lv \text{ where } L = \begin{bmatrix} a & 0 \end{bmatrix}. \tag{5}$$

Combining (1) to (5) yields

$$\begin{bmatrix} \dot{x} \\ \dot{v} \end{bmatrix} = \begin{bmatrix} G & BL \\ 0 & S \end{bmatrix} \begin{bmatrix} x \\ v \end{bmatrix}. \tag{6}$$

Using $\lambda_i(\circ)$ to denote the eigenvalue of a matrix, if $\lambda_i(G) - \lambda_j(S) \neq 0$ ($i = 1, \cdots, n$, $j = 1, 2$), the *Sylvester equation*

$$G\Pi + BL = \Pi S, \tag{7}$$

wherein $\Pi \in \mathbb{R}^{n\times 2}$, always has a unique solution. Then, denoting the kth-order identity matrix by I_k (subscript is sometimes omitted when the dimension is obvious), we get

$$\begin{bmatrix} G & BL \\ 0 & S \end{bmatrix} \begin{bmatrix} I_n & \Pi \\ 0 & I_2 \end{bmatrix} = \begin{bmatrix} I_n & \Pi \\ 0 & I_2 \end{bmatrix} \begin{bmatrix} G & 0 \\ 0 & S \end{bmatrix} \tag{8}$$

from which the solution to (6) is easily seen to be

$$x(t) - \Pi v(t) = e^{Gt}\left(x(0) - \Pi v(0)\right). \tag{9}$$

This expression conveniently shows that $x(t)$ is dependent on two parts: one is the initial condition $(x(0) - \Pi v(0))$ scaled by the exponential e^{Gt} over time, while the other is the *steady state* $\Pi v(t)$ due to the oscillatory nature of $v(t)$. More importantly, if $x(0) = \Pi v(0)$, then $x(t)$ is directly put into PSS, namely, $x(t) = \Pi v(t)$, regardless of the eigenvalues of G (i.e., G is not necessarily stable).

B. QLDAE

The recent introduction of the QLDAE [19], [20], taking the full form

$$C\dot{x} = G_1 x + G_2 x \otimes x + D_1 xu + D_2 x \otimes xu + Bu, \tag{10}$$

has enabled *exact* representation of common (strong) nonlinearities in electrical modeling, such as exponential and sine/cosine curves etc., by exploiting their "self-referential" derivative relationships. This is best illustrated with an example. Suppose we have a "diode-type" exponential nonlinear system

$$\dot{z} = -z - (e^{4z} - 1) + u, \tag{11}$$

wherein u is a scalar input and z the scalar state variable. Under zero input and zero initial state, the system remains at the equilibrium point $z = 0$. Setting $x = [z_1 \ \ z_2]^T$, $z_1 = z$ and $z_2 = e^{4z_1} - 1$ such that $\dot{z}_2 = 4e^{4z_1}\dot{z}_1 = 4(z_2+1)(-z_1 - z_2 + u)$,

and using the shorthand $x\otimes x = x^{②}$ etc. (also used throughout this paper), we end up at the QLDAE

$$\dot{x} = G_1 x + G_2 x^{②} + D_1 x u + B u \tag{12}$$

with $G_1 = -\begin{bmatrix} 1 & 1 \\ 4 & 4 \end{bmatrix}$, $G_2 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & -4 & 0 & -4 \end{bmatrix}$, $D_1 = \begin{bmatrix} 0 & 0 \\ 0 & 4 \end{bmatrix}$ and $B = \begin{bmatrix} 1 \\ 4 \end{bmatrix}$. The reader is referred to [19], [20] for other QLDAE-compatible nonlinearities such as fractions and posynomials etc.

III. Autonomous Volterra method

In the sequel, we illustrate the proposed AV scheme for PSS analysis via the generic form in (12). This is mainly done for notational ease: the non-presence of the D_2 term in (12) is because it is NOT needed in all electrical examples we have encountered, while all derivations in the paper can easily accommodate it should it be nonzero. Also, adaptation to the general QLDAE (10) whereby a singular C matrix may arise is described in Appendix A.

A. Linear Volterra subsystems

By Volterra theory [17] the solution x to (12) is approximated with the series $x(t) = x_1(t) + x_2(t) + x_3(t) + \cdots$ where

$$x_k(t) = \int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} h_k(\tau_1, \cdots, \tau_k) \cdot u(t-\tau_1)\cdots u(t-\tau_k) d\tau_1 \cdots d\tau_k, \tag{13}$$

$h_k(\tau_1, \cdots, \tau_k)$ being the *Volterra kernel* of order k. First, we apply variational analysis [17] to extract linear Volterra subsystems, i.e., by setting $u \leftarrow \alpha u$ in (12), then $x \leftarrow \alpha x_1 + \alpha^2 x_2 + \alpha^3 x_3 + \cdots$. Equating equal powers of α,

$$\dot{x}_1 = G_1 x_1 + B u, \tag{14a}$$
$$\dot{x}_2 = G_1 x_2 + G_2 x_1 \otimes x_1 + D_1 x_1 u, \tag{14b}$$
$$\dot{x}_3 = G_1 x_3 + G_2 (x_1 \otimes x_2 + x_2 \otimes x_1) + D_1 x_2 u, \tag{14c}$$
$$\dot{x}_4 = G_1 x_4 + G_2 (x_1 \otimes x_3 + x_2 \otimes x_2 + x_3 \otimes x_1) + D_1 x_3 u, \tag{14d}$$

and so on. An insight is that these linear subsystems are cascaded, with the output from the lower-order subsystem injected nonlinearly into its higher-order neighbors, e.g., [1], [18].

B. AV method

We are now ready to devise the AV scheme. We provide treatment to a single-input tone and assume a stable G_1 to start with, whereas treatment of a singular G_1 (as in the specific parameter settings below (12)) and extension to multi-tone input are discussed later.

Using the results from Section II-A and feeding the input u in (5) into (14a), the steady-state response is immediately obtained as $x_1(t) = \Pi_1 v(t)$ where $\Pi_1 \in \mathbb{R}^{n\times 2}$ solves the Sylvester equation (here $B_1 = BL$)

$$G_1 \Pi_1 + B_1 = \Pi_1 S. \tag{15}$$

A unique Π_1 always exists since the (stable) eigenvalues of G_1 and the imaginary eigenvalues in S never add to zero. In particular, the steady-state initial condition is $x_1(0) = \Pi_1 v(0)$ with respect to $u = a\cos(\omega t)$, leading to $x_1(t) = \Pi_1 v(t)$, which in fact holds for all t. Therefore, the time argument in the steady-state solution can be omitted without ambiguity, namely, $x_1 = \Pi_1 v$.

Now because $x_1^{②} = \Pi_1^{②} v^{②}$, and u is a scalar so $x_1 u = x_1 \otimes u = \Pi_1 v \otimes L v = (\Pi_1 \otimes L) v^{②}$, (14b) can be rewritten as

$$\begin{aligned} \dot{x}_2 &= G_1 x_2 + \left(G_2 \Pi_1^{②} + D_1 (\Pi_1 \otimes L)\right) v^{②} \\ &\triangleq G_1 x_2 + B_2 v^{②}. \end{aligned} \tag{16}$$

Moreover, the time derivative of $v^{②}$ yields

$$\dot{v}^{②} = \dot{v} \otimes v + v \otimes \dot{v} = (I_2 \otimes S + S \otimes I_2) v^{②} \triangleq S_2 v^{②}. \tag{17}$$

The systems (16) and (17) result in another autonomous system as in (6), leading to a second Sylvester equation with the unknown $\Pi_2 \in \mathbb{R}^{n\times 4}$, namely,

$$G_1 \Pi_2 + B_2 = \Pi_2 S_2. \tag{18}$$

Similarly, the steady-state solution reads $x_2 = \Pi_2 v^{②}$. Next, we have $x_1 \otimes x_2 = (\Pi_1 \otimes \Pi_2) v^{③}$ and $x_2 u = (\Pi_2 \otimes L) v^{③}$ etc. Defining $B_3 = G_2(\Pi_1 \otimes \Pi_2 + \Pi_2 \otimes \Pi_1) + D_1(\Pi_2 \otimes L)$ and $S_3 = S \otimes I_4 + I_2 \otimes S \otimes I_2 + I_4 \otimes S$, the third Sylvester equation

$$G_1 \Pi_3 + B_3 = \Pi_3 S_3 \tag{19}$$

then yields the steady-state $x_3 = \Pi_3 v^{③}$.

By now the pattern should become clear whereby the steady-state responses $x_k = \Pi_k v^{ⓚ}$ ($\Pi_k \in \mathbb{R}^{n\times 2^k}$, $k = 1, 2, \cdots$) are computed in turn. The aggregated PSS response with respect to u is then

$$\begin{aligned} x(t) &= x_1(t) + x_2(t) + x_3(t) + \cdots \\ &= \Pi_1 v(t) + \Pi_2 v^{②}(t) + \Pi_3 v^{③}(t) + \cdots, \end{aligned} \tag{20}$$

with the initial condition $x(0) = x_1(0) + x_2(0) + x_3(0) + \cdots$, exactly the same desired solution sought by SN algorithm.

Of course, one would question the required number of Volterra subsystems or, in other words, the number of Sylvester equations one needs to solve. Luckily, noting $\left\|v^{ⓚ}(t)\right\| = 1$, such determination is rather straightforward as $\|x_k(t)\| = \left\|\Pi_k v^{ⓚ}(t)\right\| \le \|\Pi_k\|$. That is, the AV algorithm converges upon $\|\Pi_k\|$ falling below a preset tolerance.

C. Singular system matrix

An often overlooked but critical numerical issue in QLDAE is the possibly singular G_1 in (10) or (12). For instance, in our diode-type example in Section II-B, the eigenvalues of G_1 are 0 and -5. This creates problems in solving Sylvester equations corresponding to even-order Volterra responses like x_2, x_4 etc. This is because zero eigenvalues (DC modes) exist in G_1 and S_2, S_4 etc., making the Sylvester equation such as (18) unsolvable (such scenario would not happen for odd-order S_3, S_5 etc. which do not contain zero eigenvalues). This singularity problem is not unique to AV but also causes non-invertible matrix in the finite-difference solution to (10) or (12) required in SN or regular ODE solution. The QLDAE papers [19], [20] attempt to solve this in the frequency-domain by choosing a slightly displaced expansion point about DC,

but this fix is apparently ad hoc and only works in the model order reduction context [19], [20], rather than the exact solves of (10), (12) or (14).

To tackle this singularity issue, we recognize that for stable physical systems, the DC input tone due to the QLDAE representation is automatically filtered and never appears as the actual input to the next-stage Volterra subsystems. Recalling the system in (12), and with respect to (14b) and (16), $B_2 v^{②}(t)$ serves as the input to the linear subsystem of x_2. A key insight is to recognize this term as another linear state-space system, and to note that a linear autonomous system with a certain (vector) initial condition is equivalent to another linear system with zero initial condition but with the input matrix set to that initial condition subject to an impulse input.

Specifically, $B_2 v^{②}(t)$ of (12) can be described by the compact state-space notation (in below α, α' etc. are real constants)

$$\left[\begin{array}{c|c} S_2 & v^{②}(0) \\ \hline B_2 & 0 \end{array}\right] = \left[\begin{array}{cccc|c} 0 & \omega & \omega & 0 & 1 \\ -\omega & 0 & 0 & \omega & 0 \\ -\omega & 0 & 0 & \omega & 0 \\ 0 & -\omega & -\omega & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 \\ \alpha & \beta & \gamma & -\alpha & \end{array}\right]. \quad (21)$$

Then, a similarity transform by

$$T_2 = \frac{1}{2}\begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & -1 & 0 \\ -1 & 0 & 0 & 1 \end{bmatrix}, T_2^{-1} = 2T_2^T = \begin{bmatrix} 1 & 0 & 0 & -1 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & -1 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix}, \quad (22)$$

i.e., with $S_2 \leftarrow T_2^{-1} S_2 T_2$ etc., results in an obvious model reduction as the DC components do not have any contribution:

$$\begin{aligned} \left[\begin{array}{c|c} T_2^{-1}S_2T_2 & T_2^{-1}v^{②}(0) \\ \hline B_2T_2 & 0 \end{array}\right] &= \left[\begin{array}{cccc|c} 0 & 2\omega & 0 & 0 & 1 \\ -2\omega & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ \hline 0 & 0 & 0 & 0 & 0 \\ \alpha' & \beta' & \gamma' & 0 & \end{array}\right] \\ &= \left[\begin{array}{cc|c} 0 & 2\omega & 1 \\ -2\omega & 0 & 0 \\ \hline 0 & 0 & 0 \\ \alpha' & \beta' & \end{array}\right] \\ &\triangleq \left[\begin{array}{c|c} \tilde{S}_2 & \tilde{v}^{②}(0) \\ \hline \tilde{B}_2 & 0 \end{array}\right]. \end{aligned} \quad (23)$$

Here T_2 can be derived from the eigenvectors of S in (3) and the structure of S_2 in (17) (see more details in Section III-D). Subsequently, (16) has a "reduced" equivalent form

$$\begin{aligned} \dot{x}_2(t) &= G_1 x_2(t) + B_2 T_2 e^{(T_2^{-1}S_2T_2)t} T_2^{-1} v^{②}(0) \\ &= G_1 x_2(t) + \tilde{B}_2 e^{\tilde{S}_2 t} \tilde{v}^{②}(0). \end{aligned} \quad (24)$$

Equation (24) can be cast into a pair of counterpart equations to (16) and (17), namely,

$$\dot{x}_2 = G_1 x_2 + \tilde{B}_2 \tilde{v}^{②}, \ \dot{\tilde{v}}^{②} = \tilde{S}_2 \tilde{v}^{②}. \quad (25)$$

Apparently, (18) is truncated into

$$G_1 \tilde{\Pi}_2 + \tilde{B}_2 = \tilde{\Pi}_2 \tilde{S}_2. \quad (26)$$

which is now solvable since $\tilde{S}_2$ no longer contains DC modes.

Finally, to make the solution of x_2 compatible for the solution of x_3 and beyond , we have to re-express x_2 as a function of $v^{②}(t)$ instead of $\tilde{v}^{②}(t)$. But this is straightforward since with some care it can be shown that

$$\begin{aligned} x_2(t) &= \tilde{\Pi}_2 \tilde{v}^{②}(t) \\ &= \tilde{\Pi}_2 T_2^{-1}(1:2,:) v^{②}(t) \\ &= \tilde{\Pi}_2 \begin{bmatrix} 1 & 0 & 0 & -1 \\ 0 & 1 & 1 & 0 \end{bmatrix} v^{②}(t) \triangleq \Pi_2 v^{②}(t), \end{aligned} \quad (27)$$

wherein the Matlab-type notation $T_2^{-1}(1:2,:)$ means the first two rows of T_2^{-1}. This x_2 then allows similar derivation of x_3 and so on as in (19) and thereafter. Similar removal of uncontrollable/unobservable DC modes, called minimal realization in control language, follows for x_4, x_6 etc. We summarize the pseudo code of AV in Algorithm 1.

Algorithm 1 AV PSS Algorithm

Input: G_1, G_2, D_1, B, L
Output: Steady-state response x

$j \leftarrow 1$;
$B_1 = BL$;
Solve $G_1\Pi_1 + B_1 = \Pi_1 S$ for Π_1;
$x_1 \leftarrow \Pi_1 v$;
while $\|\Pi_j\|/\|\Pi_1\| >$ tol **do**
  $S_{j+1} \leftarrow I_2 \otimes S_j + S \otimes I_2^{①}$;
  $B_{j+1} \leftarrow G_2 \sum_{m+n=j+1} \Pi_m \otimes \Pi_n + D_1(\Pi_j \otimes L)$;
  $j \leftarrow j+1$;
  if j is even and G_1 is singular **then**
    $S_j \leftarrow T_j^{-1} S_j T_j$;
    $B_j \leftarrow B_j T_j$;
    Remove the unobservable and uncontrollable modes;
    Solve $G_1\Pi_j + B_j = \Pi_j S_j$ for Π_j;
    $\Pi_j \leftarrow \Pi_j T_j^{-1}$;
  else
    Solve $G_1\Pi_j + B_j = \Pi_j S_j$ for Π_j;
  end if
  $x_j \leftarrow \Pi_j v^{①}$;
end while
$x \leftarrow \sum_{i=1}^{j} x_i$

D. Parallelizing AV

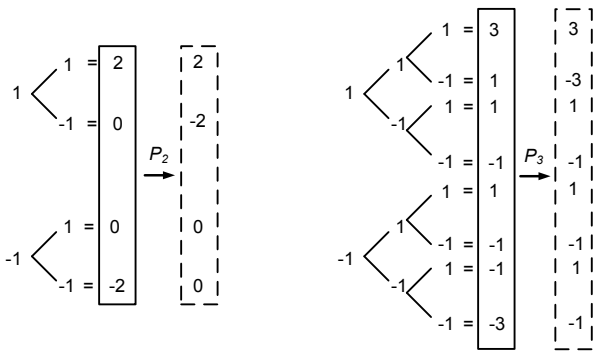
We further discuss the parallelization of AV algorithm. Obviously, the major cost of the algorithm is in solving those Sylvester equations of increasingly high orders. Luckily, one never needs to directly solve the full Sylvester equation. Each S_k is highly structured with the following recursive formula (whereby we take $S_1 = S$),

$$S_{k+1} = I_2 \otimes S_k + S \otimes I_2^{⑥}. \quad (28)$$

From (3), noting S is similar to Λ, it is readily seen that Λ (and the Λ_k similarly defined) also satisfy the same recursive relationship and

$$S_k Q^{⑥} = Q^{⑥} \Lambda_k. \quad (29)$$

Here Λ_k is a diagonal matrix holding the purely imaginary eigenvalues of S_k best illustrated with tree diagram examples in Fig. 1. As shown in the figure, the eigenvalues can be



$$P_2 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{bmatrix} \qquad P_3 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Fig. 1. (Left column) The diagonal matrix Λ_2 holds 2^2 eigenvalues which are integer multiples of $j\omega$ in the order in the solid box. The dashed box denotes the permutated order from high to low frequencies after operation by P_2. (Right column) The case for Λ_3 and P_3.

shuffled, via a (unitary) permutation matrix P_k containing only 0 and 1, into descending order of frequency pairs along the diagonal. A further similarity transform on each frequency pair [cf. (3)] then results in a block diagonal matrix with 2^{k-1} blocks, each of size 2×2. In short, such block diagonalization and decomposition process entails a (unitary) similarity transform, denoted Q_k below, on S_k:

$$\underbrace{\left((I_k \otimes Q)P_k^T(Q^{\textcircled{k}})^*\right)}_{Q_k^*} S_k \underbrace{\left(Q^{\textcircled{k}}P_k(I_k \otimes Q^*)\right)}_{Q_k}. \qquad (30)$$

Noting that all similarity transform matrices involve only permutation and Kronecker products of Q, a real and sparse ternary matrix (containing only -1, 0 and 1 with a proper scaling) can then be readily derived from the signs of the real/imaginary part of Q_k in (30). Indeed, the T_2 constructed this way has been shown in (22). As a further example, T_3 is found to be

$$T_3 = \frac{1}{4}\begin{bmatrix} 0 & -1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & -1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & -1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 0 & -1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 & -1 \\ 0 & 1 & 1 & 0 & -1 & 0 & 1 & 0 \\ 0 & 1 & -1 & 0 & 1 & 0 & 1 & 0 \\ -1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix},$$

its inverse is $T_3^{-1} = 4T_3^T$ and $T_3^{-1}S_3T_3$ reads

$$T_3^{-1}S_3T_3 = j\omega \begin{bmatrix} 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 \\ -3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \end{bmatrix}.$$

Such structure is ready for parallel computing: (19) can be transformed into

$$G_1\left(\Pi_3 T_3\right) + B_3 T_3 = \left(\Pi_3 T_3\right)\left(T_3^{-1} S_3 T_3\right) \qquad (31)$$

which corresponds to four independent Sylvester equations conformal to the 2×2 blocks in $T_3^{-1}S_3T_3$. These Sylvester equations, each solving two columns in $\Pi_3 T_3$, can then be distributed concurrently to different processor cores whose solutions are reconstructed into Π_3. An important point to note is that in practice, the T_k's needed for block diagonalization are pre-computed and stored, so they do not incur any computational overhead. Subsequently, for a single-tone input, the kth Sylvester equation is parallelizable into 2^{k-1} independent 2-column Sylvester equations. This property allows excellent scalability of AV algorithm on modern parallel computing architecture especially for high-order problems.

In our implementation, a parallel version of AV is coded and run on graphics processing unit (GPU) via the NVIDIA CUDA [27] platform. Since the size of each decomposed Sylvester equation is relatively small (viz. only two-column solves), the Kronecker product expansion is employed for its solution. For example, (15) is put into

$$(I_2 \otimes G_1 - S^T \otimes I_n)\text{vec}(\Pi_1) = -\text{vec}(B_1), \qquad (32)$$

where vec$(\circ)$ denotes the vectorization operator formed by stacking the columns of its argument into a single column vector. This linear problem is then directly solved by LU decomposition with pivoting on GPU cores. In general, iterative solvers such as CG and GMRES show better efficiency than LU. Nonetheless, LU is better suited for the GPU architecture. In CUDA, each stream processor has a number of duplicated cores that execute the same instruction concurrently but with different data, named single instruction multiple data (SIMD). The advantage of LU is that it allows solving each problem with the same instruction on each core without branches, while the instructions of iterative solvers depend on the convergence of each individual problem. Subsequently, iterative linear solvers may result in low parallelization efficiency as the cores on one processor can only execute the same instruction simultaneously. We remark that there exist algorithms for parallelized CG and GMRES, e.g., [28], [29], but the parallelization lies only in the matrix-vector multiplication in the original CG or GMRES routines. In other words, they are still solving one algebraic equation on the multi-core platform, whereas the proposed parallelized AV exploits decomposition into smaller sub-problems and their simultaneous solution.

An example of the parallelized GPU solver is shown in Fig. 2 wherein NVIDIA GeForce GTX 580 with 3GB RAM

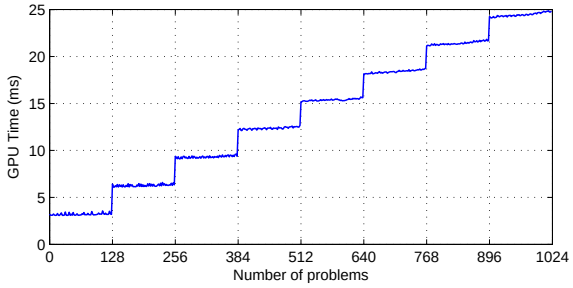


Fig. 2. The GPU time of the parallelized solver vs number of equations in the inverter chain example (NVIDIA GeForce GTX 580, 3GB RAM).

is used. Different numbers of equations, corresponding to the example in Section IV-C, are distributed to the parallelized LU matrix solver. The time of data transfer is not included in the results. The staircase shape of the curve indicates that the maximum number of equations this particular GPU can solve concurrently is 128.

Suppose there are $\mathcal{O}(n_s)$ nonzero entries in the sparse matrix G_1, the complexity of a sparse LU decomposition and forward/backward substitutions will be $\mathcal{O}(n_s^\alpha)(1 < \alpha < 2)$. Thus the total complexity of the algorithm will be $\mathcal{O}(2^k n_s^\alpha)$ $(1 < \alpha < 2)$, where k is the number of Volterra subsystems needed. If the number of equations 2^{k-1} in kth-order is smaller than the maximum number the GPU can solve concurrently, the total complexity can be reduced to $\mathcal{O}(kn_s^\alpha)$ via parallelization. In practice, the Volterra responses usually decay fast so that k is almost always within ten (see numerical examples).

IV. Numerical Examples

We compare AV against the conventional SN and HB. We have used uniform step size backward-Euler method for the transient solver in single-/multi-tone SN, and FFT-based single-/multi-tone HB. All codes have been coded in Matlab and CUDA. The performance of parallelized AV is also contrasted with its sequential implementation. Finally, a classic intermodulation test of AV against HB is provided at the end of this section.

A. Illustrative examples

We start by computing the PSS responses of two simple examples for illustrative purpose. The first example is the exponential diode-curve equation (11) discussed in Section II, with an input $u(t) = 2\cos(20\pi t)$. The AV method is applied to its singular QLDAE system matrix (12), while SN and HB methods are applied to the original system (11). The convergence tolerance of the Newton-Raphson iterations within SN and HB is set to be around 10^{-7} in all following examples. Specifically, AV computes the PSS response which is then compared with the result of ODE solver which simulates the system long enough to eliminate all transient effects. All experiments are carried out on a desktop with Intel i5 750@2.67GHz, 16GB RAM and Windows 7. We define the error as the average relative error over a time period. And the runtimes of the experiments are defined as the overall CPU (and GPU) runtime counted by Matlab (and CUDA built-in timers), which include the time of data transfer between the host and GPU memories.

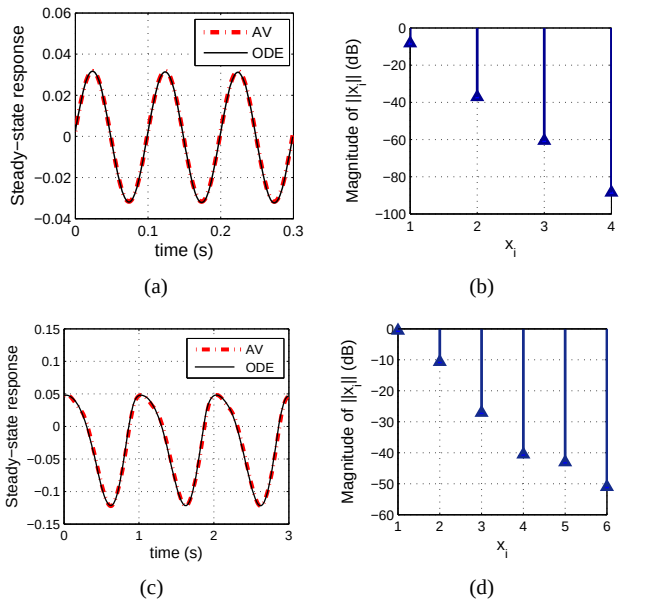


Fig. 3. (a) Time-domain waveforms of the exponential nonlinear example and (b) magnitude of x_i. (c) Time-domain waveforms of the rational nonlinear example and (d) magnitude of x_i.

TABLE I
The Exponential Example

	time/freq pts	itr	CPU time (ms)	error (%)
AV	N/A	N/A	12	1.7
SN	200	4	43	1.4
HB	200	5	15	0.1

In this example, AV computes the response up to x_4. Figs. 3(a) and 3(b) show the AV time-domain responses as well as the decay of x_i's magnitude. Table I lists the performance and relative error of each method. The second column shows the number of time steps and Fourier series length in SN and HB, respectively, and the third column lists the number of top-level iterations for them to converge. It is seen that the proposed AV method exhibits similar efficiency to SN and HB with comparable error margins.

The second example is a rational nonlinear equation

$$\dot{z} = -z + \frac{z}{z+k} + u. \tag{33}$$

Setting $z_1 = z$, $z_2 = \frac{z_1}{z_1+k}$, then $0 = -z_1 + kz_2 + z_1 z_2$. Defining $x = [z_1 \;\; z_2]^T$, we get its QLDAE

$$C\dot{x} = G_1 x + G_2 x^{②} + D_1 xu + Bu, \tag{34}$$

with the specific parameters

$$C = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix},\; G_1 = \begin{bmatrix} -1 & 1 \\ -1 & k \end{bmatrix},\; D_1 = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix},$$
$$G_2 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix},\; B = \begin{bmatrix} 1 \\ 0 \end{bmatrix}.$$

Assuming its input $u(t) = \cos(2\pi t)$ and $k = -0.1$, AV is applied to the QLDAE (33) producing the PSS results

TABLE II
THE RATIONAL EXAMPLE

	time/freq pts	itr	CPU time (ms)	error (%)
AV	N/A	N/A	7	1.8
SN	100	4	21	4.6
HB	51	6	17	0.35

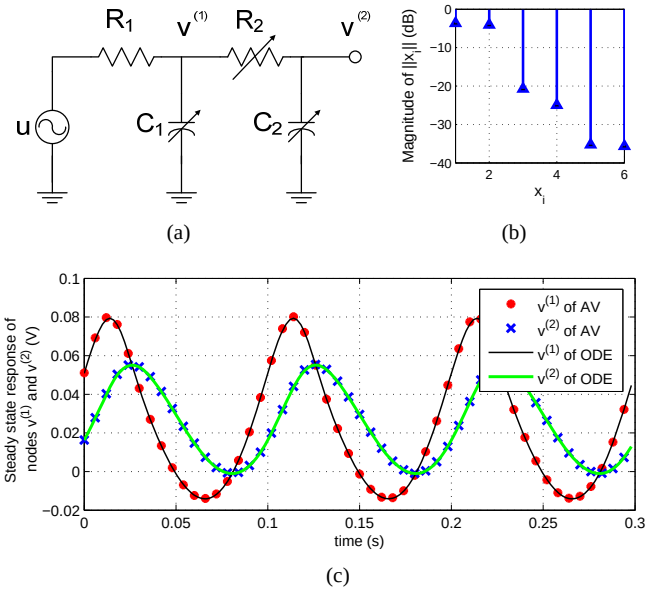


Fig. 4. (a) A nonlinear RC circuit. (b) Magnitude of x_i. (c) PSS waveforms of AV compared with ODE reference.

in Figs. 3(c) and 3(d). Again, Table II shows comparable performance of AV with SN and HB.

B. Nonlinear RC circuit

Fig. 4(a) is a communication RC circuit with second-order nonlinear resistors and capacitors. The nonlinearities of the resistors and capacitors are given by $i_{R_2}(v) = (20k\Omega)^{-1}v + (80k\Omega V)^{-1}v^2$ and $C_1(v) = C_2(v) = (1-2v)\mu$F, respectively. The frequency of the AC input voltage is $f = 10$Hz. Again, we compare the results of AV, SN and HB with ODE PSS response. The waveforms of AV method are depicted in Fig. 4(c). It is shown in Table III that though only up to 6th-order subsystems have been used, the accuracy of AV is comparable to SN and HB but with the fastest runtime.

C. Inverter chain

The circuit in Fig. 5(a) is adapted from [30]. The small-signal model of the MOSFET is shown in Fig. 5(b). We assume in the circuit, $R_S = R_L = 1k\Omega$, $C_g = 20\mu$F, $C_L = 1\mu$F, $g_1 = 10^{-3}\Omega^{-1}$, $g_2 = 10^{-3}\Omega^{-1}V^{-1}$ and the input source $V_{in}(t) = 2\cos(20\pi t)$. The number of stages, N, is 50. The PSS waveforms of the first two nodes $v^{(1)}, v^{(2)}$ and the output node $v^{(N)}$ are shown in Figs. 6(a) and 6(b), respectively, against their ODE solutions. Up to 6th-order of subsystems have been used in AV and their decaying magnitudes are displayed in Fig. 6(c).

Next, we increase the stages N to highlight the efficiency of the AV algorithm. We set N to be $10, 50, 100, 500$ and 1000,

TABLE III
THE NONLINEAR RC CIRCUIT EXAMPLE

	time/freq pts	itr	CPU time (ms)	error (%)
AV	N/A	N/A	8	0.6
SN	50	3	93	1.2
HB	51	3	17	0.08

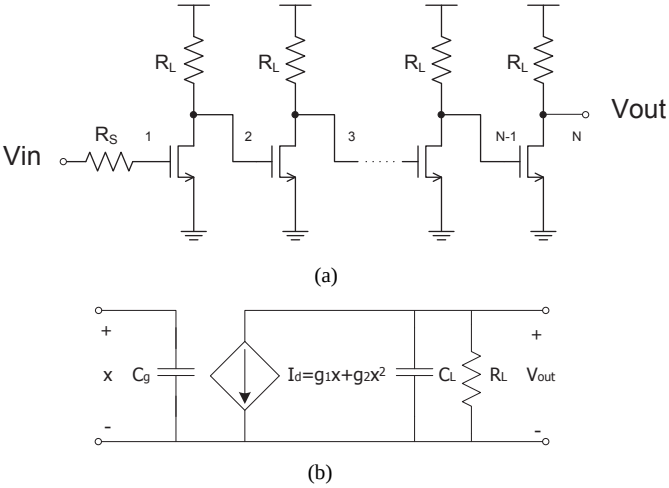


Fig. 5. (a) An inverter chain circuit. (b) Small-signal model of a MOSFET.

respectively. The comparison of the CPU times is plotted in Fig. 6(d). It can be seen that that AV has a similar scalability to SN and HB in terms of the time complexity.

D. Double-balanced mixer

Extension to multi-tone inputs or multi-input is trivial in the AV scheme. We illustrate this with the double-balanced mixer circuit in Fig. 7 [31], where $V_{\rm rf}(= V_{\rm rf+} - V_{\rm rf-})$ and $V_{\rm lo}(= V_{\rm lo+} - V_{\rm lo-})$ are the RF and local oscillating (LO) inputs, respectively, and $V_{\rm if}(= V_{\rm if+} - V_{\rm if-})$ is the intermediate-frequency (IF) output. We use the same MOSFET model as in the inverter chain example. The autonomous system matrix S in (2) is simply modified as

$$S = \begin{bmatrix} 0 & \omega_{\rm rf} & 0 & 0 \\ -\omega_{\rm rf} & 0 & 0 & 0 \\ 0 & 0 & 0 & \omega_{\rm lo} \\ 0 & 0 & -\omega_{\rm lo} & 0 \end{bmatrix} \tag{35}$$

where $\omega_{\rm rf}$ and $\omega_{\rm lo}$ are the RF and LO radian frequencies, respectively. In multi-input multi-output (MIMO) cases, the number of columns of the system matrices B, D_1 and D_2 in QLDAE (10) are expanded for each input, followed by direct application of the AV algorithm.

In the double-balanced mixer example, we assume the frequencies of $V_{\rm rf}$ and $V_{\rm lo}$ are 2GHz and 10MHz, respectively. Up to 5th-order subsystems are computed in AV. The AV PSS solution is contrasted with the ODE solution in Fig. 8. To fairly present the responses of all three methods under multi-tone excitation, we compared multidimensional FFT-based multi-tone HB and multi-time partial differential equation (MPDE)-based method [8] for multi-tone SN against AV. We choose (20, 20)-harmonic in each frequency axis for multi-tone HB

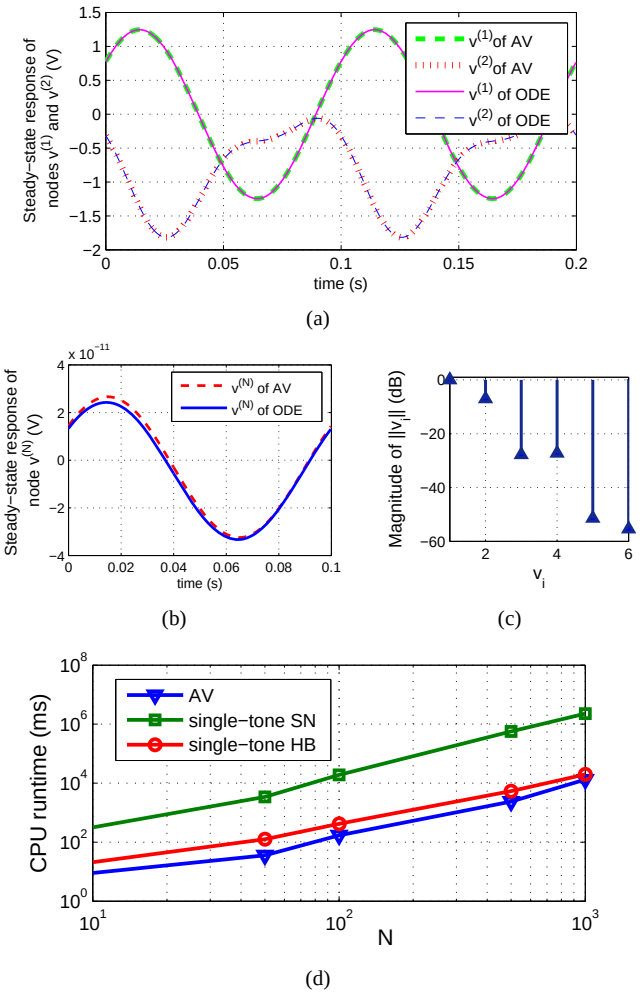


Fig. 6. (a) Time-domain waveforms $v^{(1)}, v^{(2)}, N = 50$ of AV and ODE reference. (b) Time-domain waveform $v^{(N)}, N = 50$ of AV and ODE reference. (c) Magnitude of x_i. (d) CPU times of AV, single-tone SN and HB vs. N.

TABLE IV
THE DOUBLE-BALANCED MIXER EXAMPLE

	time/freq pts	itr	CPU time (ms)	error (%)
AV	N/A	N/A	863	0.2
Multi-tone SN	(20, 20)	4	2180	4.9
Multi-tone HB	(20, 20)	4	762	0.13

and (20, 20)-time step in each time axis for MPDE. The comparison of different PSS methods are given in Table IV which shows that in this widely separated two-tone example, AV demonstrates better efficiency than multi-tone SN and similar efficiency as multi-tone HB.

E. Parallel performance

Now we test the parallelized AV algorithm on these examples to compare with the standard AV solver. For fair comparison across platforms, our decomposition-based CUDA parallel AV is compared with the standard AV using the Sylvester equation solver by Matlab parallel toolbox which is also working on CUDA platform. Except for solving the Sylvester equation, the remaining steps of the standard AV

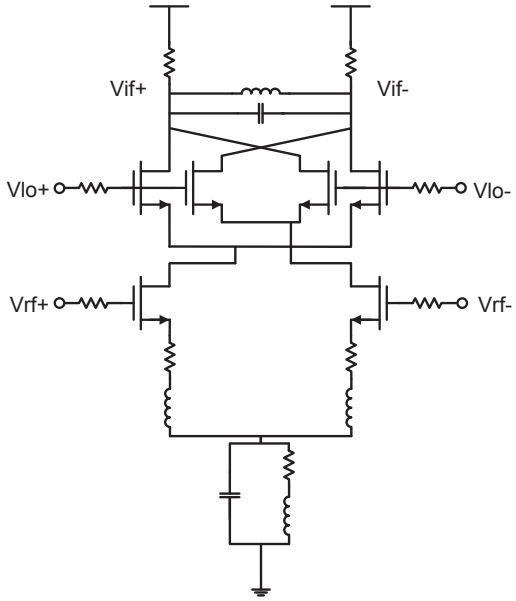


Fig. 7. A double-balanced mixer.

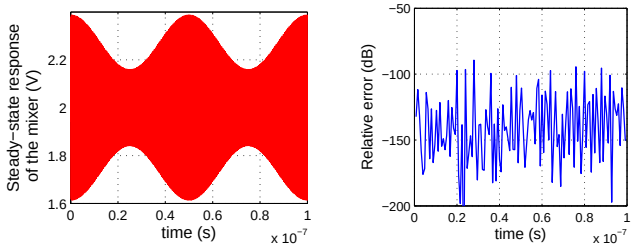


Fig. 8. (a) Time-domain waveform of $V_{\rm if}$. (b) Relative error of the AV solution.

and parallelized AV are implemented in Matlab. The runtime statistics are listed in Table V. The third row in Table V enumerates the total number of smaller equations after the decomposition in Section III-D. It can be found in the table that solving a large number of equations concurrently will gain a great speedup over the single full-scale solution even working on the same parallel platform. In the exponential, rational and RC examples, it is seen that the efficiency of the parallel algorithm is similar to the full-scale solver. This is because when the problem size is small and the order of subsystem is low, the standard Sylvester equation solver is good enough while the parallel realization suffers from the overhead of data transfer between the host and GPU memories, as well as the similarity transforms before/after the Sylvester equation partition.

To demonstrate the efficiency of the parallel AV realization, we make use of the (multi-tone-input) mixer example and compare parallel AV with multi-tone SN and HB. For fairness the latter two are also executed with standard (sequential) and GPU-parallelized matrix solver, respectively. The runtimes of their sequential and parallel implementations are listed in Table VI. It is shown in the table that though parallel multi-tone SN and HB can be accelerated by the GPU-based matrix

TABLE V
RUNTIMES OF STANDARD AND PARALLELIZED AV
(INTEL I5 750@2.67GHZ, 16GB HOST RAM, NVIDIA GEFORCE GTX 580, 3GB DEVICE RAM)

	exponential	rational	RC	inverter	mixer
order of subsystems	4	6	6	11	5
equations	15	63	63	4095	682
standard AV (ms)	15	13	15	7830	569
parallel AV (ms)	10	10	9	872	161
speedup	1.5	1.3	1.7	9.0	3.5

TABLE VI
STANDARD AND PARALLEL METHODS ON THE DOUBLE-BALANCED MIXER EXAMPLE

	time/freq pts	seq. (ms)	paral. (ms)	speedup
AV	N/A	863	161	5.36
Multi-tone SN	(20, 20)	2180	998	2.18
Multi-tone HB	(20, 20)	762	525	1.45

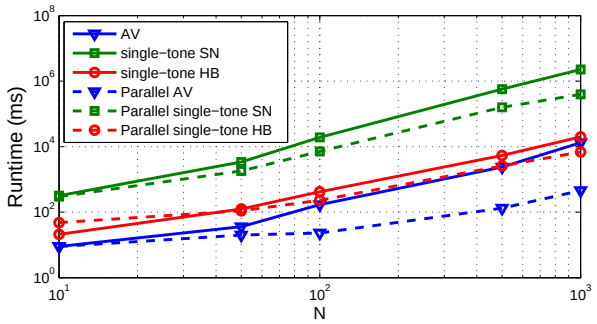


Fig. 9. Runtimes of the sequential/parallel AV, single-tone SN and HB vs. N.

solver, the parallel AV still demonstrates an obvious advantage in speedup.

Finally, we apply parallel AV on the (single-tone-input) inverter chain system to see how the solution of larger scale PSS problems benefits from the inherent block diagonalization in AV. As before, we also contrast this against both sequential and parallel single-tone SN and HB. The results are shown in Fig. 9. It is obviously seen that the parallel versions of AV, SN and HB are more efficient than their sequential methods. In particular, the speedup of parallel AV is more significant than the other two. This is because in parallel SN and HB, only GPU-based matrix solver is exploited whereas parallel AV further utilizes the partitioned Sylvester equations whose subproblems are always two-column matrix equations irrespective of order of the Volterra subsystems.

F. Intermodulation test

We end this section with a classic two-tone intermodulation test to demonstrate that AV can achieve a better dynamic range than the HB we used, which is also comparable to the performance of commercial solvers. In the test, two tones are applied to a nonlinear system. With one tone fixed, the level of one of the intermodulation products is plotted versus the magnitude of the other input tone. Then the linear slope range is defined to be the dynamic range.

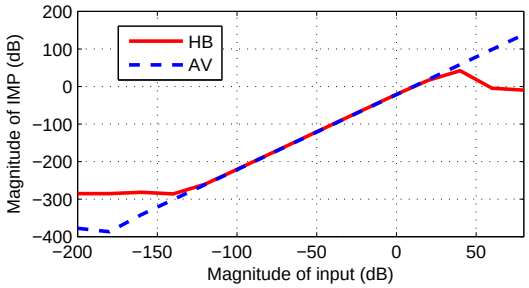


Fig. 10. Two-tone intermodulation product test for HB and AV.

We present the test by a simple example here, namely, $\dot{x} = -x - x^2 + u$. The two input tones are $\omega_1 = 2.9$ and $\omega_2 = 3.1$. Then, we fix the magnitude of the tone ω_2 and observe the intermodulation product at the frequency $2\omega_1 - \omega_2$ by varying the magnitude of the first tone ω_1. The results of the test are shown in Fig. 10. It can be seen that both HB and AV achieve a dynamic range better than 100dB, which are comparable to commercial simulators. The bending at the lower end of the HB curve is probably caused by numerical error when the input is very small, whereas that at the upper end is likely caused by higher order intermodulation terms aliased into $2\omega_1 - \omega_2$. In AV, we are testing the coefficient of the $2\omega_1 - \omega_2$ term directly, which always has the 2-to-1 slope.

V. ADDITIONAL REMARKS

1) It should be pointed out that PSS analysis via closed-form Volterra transfer functions (in contrast to the linear subsystem viewpoint in this paper) is *not new* and has been discussed in, e.g., [17], [32]. Nonetheless, such transfer function approach simply *breaks* for any singular G_1, e.g., in (12), as it renders the even-order transfer functions, $H_2(s_1, s_2)$, $H_4(s_1, s_2, s_3, s_4)$ etc., undefined at DC or zero frequency. A more subtle reason for the doomed usage of transfer functions (viz. *external* I/O representation) is the implicit assumption of zero initial condition in their general expressions. As a quick example, the Laplace transfer function of (14a) is generally written as $(sI - G_1)^{-1}B$ whereas the full representation should be $(sI - G_1)^{-1}(B + x_1(0))$, but then $x_1(0)$ is the unknown to be found in steady-state exercise. The formulas become even more complicated for higher-order Volterra systems wherein the effect of initial condition enters nonlinearly. In contrast, the linear subsystem perspective (viz. *internal* node representation) adopted in the proposed AV method naturally handles

finite initial conditions as well as their (nonlinear) Kronecker products.

2) Unlike SN, a major merit in AV is that given an arbitrary nonlinear system, no guesses of shooting initial condition and simulation time step size are required. Indeed, every iteration in AV, namely, the setup and solution of a Sylvester equation, has deterministic matrix-algebraic formulation and solution.
3) Typical HB solves a presumed length of Fourier coefficients from the nonlinear frequency-domain equations. While typical SN requires monitoring the output response and adaptively sizing the simulation time steps to reduce computation. In contrast, due to the purely matrix-algebraic setup in AV, such series length guessing and adaptive time stepping are completely avoided. By measuring the norms of the Sylvester equation solutions, a simple error estimator also exists in AV to control the accuracy of the steady-state solution. Moreover, with parallel AV, the time complexity of AV is essentially linearly proportional to the order of the Volterra kernels being used, if it is not limited by the fabrics of the GPU card.
4) Apart from sine tones, other input waveforms (e.g., those in the format of output signals of a linear system fed with one or more sine tones) can also be constructed/approximated via this autonomous approach. Furthermore, as in SN and HB, the Sylvester equations in AV are readily transformed into a structured and sparse linear system of equations whereby efficient iterative solvers and parallelization can be utilized.
5) The current limitation of AV is that it can only accommodate DAEs with uniform and analytical functions, but not piecewise continuous functions, while HB and SN can deal with piecewise functions or even functions without analytical forms. Although such issue can be alleviated by a more accurate (higher order) polynomial fitting of the piecewise curve followed by quadratic-linearization into again a QLDAE, we admit this restriction limits the application scope of AV such as in practical transistor modeling in different operating regions. Therefore AV is not comparable to SN or HB in complicated model cases at this stage. On-going work is being done to lift this limitation. Also, the accuracy of AV cannot be compared to HB with a high order of harmonics. However, for large-scale problems as seen from the numerical section, AV remains the most feasible and scalable choice due to its parallelization readiness.

VI. Conclusion

This paper has presented a novel idea of performing PSS analysis of nonlinear circuits based on the elegant leverage of nonlinear DAEs and linear Volterra subsystems. Compared to the widely used SN and HB, AV employs simple matrix-algebraic formulation and does not require any initial guess and time/frequency-domain computation. The numerical issue in the possibly singular system matrix is addressed through standard state-space operations, and extension to multi-tone and other input waveforms may readily be accommodated through this autonomous platform. Parallelization of AV for scalable implementation has also been enabled through exploiting matrix structures. Numerical examples have demonstrated that AV exhibits fast speed with comparable accuracy to SN and HB, and has excellent scalability towards large-scale problems due to its readiness for parallelization.

References

[1] 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.

[2] O. Nastov, R. Telichevesky, K. Kundert, and J. White, "Fundamentals of fast simulation algorithms for RF circuits," *Proc. IEEE*, vol. 95, no. 3, pp. 600–621, Mar. 2007.

[3] R. A. Rutenbar, G. G. E. Gielen, and J. Roychowdhury, "Hierarchical modeling, optimization, and synthesis for system-level analog and RF designs," *Proc. IEEE*, vol. 95, no. 3, pp. 640–669, Mar. 2007.

[4] A. Odabasioglu, M. Celik, and L. T. Pileggi, "PRIMA: Passive reduced-order interconnect macromodeling algorithm," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 17, no. 8, pp. 645–654, Aug. 1998.

[5] J. R. Phillips, L. Daniel, and L. M. Silveira, "Guaranteed passive balancing transformations for model order reduction," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 22, no. 8, pp. 1027–1041, Aug. 2003.

[6] A. C. Antoulas, *Approximation of Large-Scale Dynamical Systems*. Philadelphia: SIAM, 2005.

[7] K. Mayaram, D. C. Lee, S. Moinian, D. A. Rich, and J. Roychowdhury, "Computer-aided circuit analysis tools for RFIC simulation: algorithms, features, and limitations," *IEEE Trans. Circuits Syst. II*, vol. 47, no. 4, pp. 274–286, Apr. 2000.

[8] J. Roychowdhury, "Analyzing circuits with widely separated time scales using numerical PDE methods," *IEEE Trans. Circuits Syst. I*, vol. 48, no. 5, pp. 578–594, May 2001.

[9] T. J. A. Jr. and T. N. Trick, "Steady-state analysis of nonlinear circuits with periodic inputs," *Proc. IEEE*, vol. 60, no. 1, pp. 108–114, Jan. 1972.

[10] K. Kundert and A. Sangiovanni-Vincentelli, "Simulation of nonlinear circuits in the frequency domain," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 5, no. 4, pp. 521–535, october 1986.

[11] R. Telichevesky, K. Kundert, and J. White, "Efficient steady-state analysis based on matrix-free Krylov-subspace methods," in *Design Automation, 1995. DAC '95. 32nd Conference on*, 1995, pp. 480–484.

[12] J. H. Haywood and Y. L. Chow, "Intermodulation distortion analysis using a frequency-domain harmonic balance technique," *IEEE Trans. Microw. Theory Tech.*, vol. 36, no. 8, pp. 1251–1257, Aug. 1988.

[13] K. Kundert, "Simulation methods for RF integrated circuits," in *Computer-Aided Design, 1997. Digest of Technical Papers., 1997 IEEE/ACM International Conference on*, nov 1997, pp. 752 –765.

[14] X.-X. Liu, H. Yu, J. Relles, and S.-D. Tan, "A structured parallel periodic Arnoldi shooting algorithm for RF-PSS analysis based on GPU platforms," in *Design Automation Conference (ASP-DAC), 2011 16th Asia and South Pacific*, Jan. 2011, pp. 13–18.

[15] Y. Thodesen and K. Kundert, "Parametric harmonic balance," in *Microwave Symposium Digest, 1996, IEEE MTT-S International*, vol. 3, jun 1996, pp. 1361 –1364 vol.3.

[16] F. P. Hart, D. G. Stephenson, C.-R. Chang, K. Gharaibeh, R. G. Johnson, and M. B. Steer, "Mathematical foundations of frequency-domain modeling of nonlinear circuits and systems using the arithmetic operator method," *Intl. J. RF and Microwave Computer-Aided Engineering*, vol. 13, no. 6, pp. 473–495, 2003.

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

[18] 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.

[19] C. Gu, "QLMOR: a new projection-based approach for nonlinear model order reduction," in *Proc. Int. Conf. Computer Aided Design*, Nov. 2009, pp. 389–396.

[20] ——, "QLMOR: a projection-based nonlinear model order reduction approach using quadratic-linear representation of nonlinear systems," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 30, no. 9, pp. 1307–1320, Sep. 2011.
[21] A. Isidori and C. I. Byrnes, "Output regulation of nonlinear systems," *IEEE Trans. Autom. Control*, vol. 35, no. 2, pp. 131–140, Feb. 1990.
[22] C. I. Byrnes, D. S. Gilliam, A. Isidori, and J. Ramsey, *On the Steady-State Behavior of Forced Nonlinear Systems*, ser. Lecture Notes in Control and Information Sciences. Springer, 2003, vol. 295/2003.
[23] A. Astolfi, "A new look at model reduction by moment matching for linear systems," in *Proc. IEEE Conf. Decision and Control*, Dec. 2007, pp. 4361–4366.
[24] ——, "Model reduction by moment matching for nonlinear systems," in *Proc. IEEE Conf. Decision and Control*, Dec. 2008, pp. 4873–4878.
[25] ——, "Model reduction by moment matching, steady-state response and projections," in *Proc. IEEE Conf. Decision and Control*, Dec. 2010, pp. 5344–5349.
[26] ——, "Model reduction by moment matching for linear and nonlinear systems," *IEEE Trans. Autom. Control*, vol. 55, no. 10, pp. 2321–2336, Oct. 2010.
[27] CUDA (Compute Unified Device Architecture). [Online]. Available: http://www.nvidia.com/object/cuda_home_new.html
[28] M. Wang, H. Klie, M. Parashar, and H. Sudan, "Solving sparse linear systems on NVIDIA Tesla GPUs," in *Proc. Int. Conf. Computational Science: Part I*, May 2009, pp. 864–873.
[29] N. Bell and M. Garland, "Implementing sparse matrix-vector multiplication on throughput-oriented processors," in *Proc. Conf. High Performance Computing, Networking, Storage and Analysis*, Nov. 2009, pp. 1–11.
[30] T. Voβ, R. Pulch, E. Maten, and A. Guennouni, "Trajectory piecewise linear approach for nonlinear differential-algebraic equations in circuit simulation," in *Scientific Computing in Electrical Engineering*, ser. Mathematics in Industry. Springer Berlin Heidelberg, 2007, vol. 11, pp. 167–173.
[31] W. Dong and P. Li, "A parallel harmonic-balance approach to steady-state and envelope-following simulation of driven and autonomous circuits," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 28, no. 4, pp. 490 –501, april 2009.
[32] E. Bedrosian and S. O. Rice, "The output properties of Volterra systems (nonlinear systems with memory) driven by harmonic and Gaussian inputs," *Proc. IEE*, vol. 59, no. 12, pp. 1688–1707, Dec. 1971.

APPENDIX A
SINGULAR C MATRIX IN QLDAE

First we give a proof of when the system matrix C in QLDAE becomes singular. In [19], [20], specifically, it is assumed that each element of the original nonlinear function can be regarded as a linear summation of several nonlinear elementary functions $g_i(x)$,

$$\dot{x} = a_{i,0}x + a_{i,1}g_1(x) + \quad \cdots + a_{i,k}g_k(x) + b_i u, \qquad (36)$$

which can represent most nonlinear circuit problems. The quadratic-linearization procedure of (36) can be summarized as the following two steps:

1) Build a polynomial equation of (36) by adding extra state variables;
2) Convert the polynomial equation into QLDAE.

For rational nonlinear elementary functions $g_i(x)$, e.g., $\frac{x}{x+k}$ or x^3, the way to polynomialize and quadratic-linearize them into QLDAE form is by adding polynomial algebraic equations. It can be proved that the process of such quadratic-linearization will render the C matrix in (10) singular. Specifically, to convert (36) into QLDAE form, the nonlinear function $g_i(x)$ is replaced by a new variable y_i which satisfies $y_i = g_i(x)$. Then, a new polynomial equation $0 = y_i - g_i(x)$ is appended to the set of state equations. Now, y_i is only a function of x rather than of $\dot{x}$. Therefore, in the C matrix in (10), the row corresponding to y_i must be all zeros, which results in a singular C after the quadratic-linearization.

The decomposition of the C matrix can be used to deal with such singularity problem. First, the C matrix in (10) can always be decomposed into a nonsingular part and a zero part after the quadratic-linearization.

$$C = \begin{bmatrix} C_n & 0 \\ 0 & 0 \end{bmatrix},$$

where C_n is the nonsingular component of C. From the discussion of the AV method in Section III-B, the jth-order Volterra subsystem can be represented by the linear system

$$C\dot{x}_j = G_1 x_j + B_j v^{\textcircled{j}}. \qquad (37)$$

The state matrices and variables in (37) can then be decomposed conformally to C, namely,

$$\begin{bmatrix} C_n & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \dot{x}'_j \\ \dot{x}''_j \end{bmatrix} = \begin{bmatrix} G_{11} & G_{12} \\ G_{21} & G_{22} \end{bmatrix} \begin{bmatrix} x'_j \\ x''_j \end{bmatrix} + \begin{bmatrix} B'_j \\ B''_j \end{bmatrix} v^{\textcircled{j}}. \qquad (38)$$

If each $g_i(x)$ in (36) is well-defined and stable at the equilibrium, which is the case in most practical circuits, then it can be easily shown that G_{22} is nonsingular such that x''_j is related to x'_j as

$$\dot{x}'_j = G' x'_j + B' v^{\textcircled{j}}, \qquad (39)$$

where $G' = C_n^{-1}(G_{11} - G_{12}G_{22}^{-1}G_{21})$ and $B' = C_n^{-1}(B'_j - G_{12}G_{22}^{-1}B''_j)$. Next, setting up the autonomous system with x'_j, the steady-state solution of x'_j can be obtained by solving the following Sylvester equation with the unknown Π'_j,

$$G'\Pi'_j + B' = \Pi'_j S_j. \qquad (40)$$

Therefore, $x'_j = \Pi'_j v^{\textcircled{j}}$. Finally, expressing x''_j by x'_j we get the final jth-order steady-state solution x_j,

$$x_j = \begin{bmatrix} x'_j \\ x''_j \end{bmatrix} = \begin{bmatrix} \Pi'_j \\ -G_{22}^{-1}G_{21}\Pi'_j - G_{22}^{-1}B''_j \end{bmatrix} v^{\textcircled{j}} \triangleq \Pi_j v^{\textcircled{j}}. \qquad (41)$$

Haotian Liu (S'11) received the B.S. degree in microelectronic engineering from Tsinghua University, Beijing, China, in 2010. He is currently working toward a Ph.D. degree in electrical and electronic engineering at the University of Hong Kong, Hong Kong.

He was a research intern at Sun (China) Engineering and Research Institute, Beijing, China, in 2010. His research interests include very-large-scale integration (VLSI) simulation, model order reduction, parallel computation and control theory.

Ngai Wong (S'98M'02) received the B.E. (with first class honors) and Ph.D. degrees in electrical and electronic engineering from the University of Hong Kong, Pokfulam, Hong Kong, in 1999 and 2003, respectively.

He was an Intern with Motorola, Inc., Kowloon, Hong Kong, from 1997 to 1998, specializing in product testing. He was a Visiting Scholar with Purdue University, West Lafayette, IN, in 2003. Currently, he is an Associate Professor with the University of Hong Kong. His current research interests include linear and nonlinear circuit modeling and simulation, model order reduction, passivity test and enforcement, and numerical algorithms in electronic design automation.