Efficient small-signal circuit analysis and sensitivity computations with the PVL algorithm

Roland W. Freund
AT&T Bell Laboratories
Room 2C–420
600 Mountain Avenue
Murray Hill, NJ 07974–0636, USA

Peter Feldmann
AT&T Bell Laboratories
Room 2C–217
600 Mountain Avenue
Murray Hill, NJ 07974–0636, USA

Abstract

In this paper we describe the application of the PVL algorithm to the small-signal analysis of circuits, including sensitivity computations. The PVL algorithm is based on the efficient computation of the Padé approximation of the network transfer function via the Lanczos process. The numerical stability of the algorithm permits the computation of the Padé approximation to any accuracy over a certain frequency range. We extend the algorithm to compute sensitivities of network transfer functions, their poles, and zeros, with respect to arbitrary circuit parameters, with minimal additional computational cost. We then demonstrate the implementation of our algorithm on circuit examples.

1 Introduction

The process of analyzing analog circuits with full accounting of parasitic elements, interconnect analysis at the board or chip level, and numerous other circuit simulation tasks often require the analysis of large linear networks. These networks can become extremely large, especially when circuits are automatically extracted from layout, or contain models of distributed elements, such as transmission lines, ground planes, antennas, and other

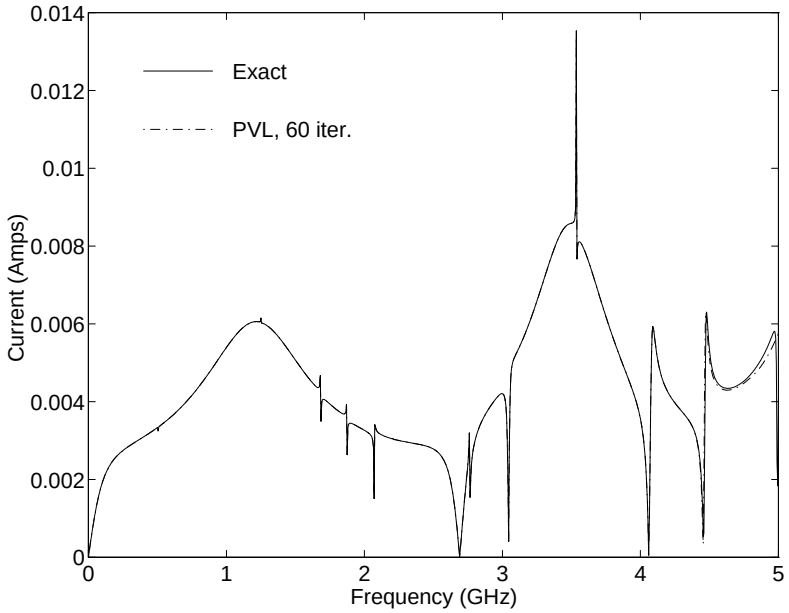

Figure 1: Results for the PEEC circuit, 60 PVL iterations

three-dimensional structures. The use of time-domain differential-equation integration or complex phasor analysis, as implemented in SPICE-like simulators, would be inefficient or even prohibitive for such large problems.

Recently, in [1], we have introduced a new algorithm for the efficient analysis of large linear networks. This algorithm, called PVL (**P**adé **V**ia **L**anczos), computes in a numerically stable way, the Padé approximation of a linear circuit response via the Lanczos process [2].

A key feature of the PVL algorithm is the fact that the accuracy of the approximation can be guaranteed for a specified frequency range. In addition, the efficiency of the algorithm is such that it allows computation of the circuit response over an entire frequency range with similar computational effort as classical phasor analysis would require for a single frequency point. Therefore, PVL can be used to replace complex phasor analysis altogether.

In Figure 1, we illustrate this point with an example taken from [1]. Here, we simulate a lumped-element equivalent circuit for a three-dimensional electromagnetic problem modeled via PEEC [3] (partial-element equivalent circuit). Using a single Padé approximant (of order 60), PVL accurately simulates the circuit response over a frequency range of 5 GHz. One would need to carry out a high-resolution complex phasor analysis involving a large number of frequency points to capture all the resonance frequencies. The response shown in Figure 1 was obtained with the PVL algorithm at a cost that is smaller than phasor analysis for a single frequency point.

In addition to efficiency and accuracy, the PVL algorithm has other features that makes it useful in the analysis of large linear circuits. One important application is the computation

of sensitivities of the impulse response or other measures related to it such as the location of is poles or zeros. In this paper we describe the application of the PVL algorithm to the small-signal analysis of circuits, including sensitivity computations.

The remainder of the paper is organized as follows. In Section 2, we establish some notation and review the small-signal analysis of circuits. In Section 3, we sketch the PVL algorithm. In Section 4, we extend the PVL algorithm to compute sensitivities of the impulse response and related quantities. In Section 5, we present results of numerical experiments with this extended version of PVL. In Section 6, we make some concluding remarks.

2 Small-signal circuit equations

All of the established circuit-equation formulation methods, such as MNA, sparse tableau, etc. [4] lead to a system of differential equations of the form

$$\mathbf{f}(\mathbf{z},t) + \frac{d}{dt}\mathbf{q}(\mathbf{z},t) = 0. \tag{1}$$

Here, $\mathbf{z} = \mathbf{z}(t)$ is the vector of circuit variables at time t, the term $\mathbf{f}(\mathbf{z},t)$ represents the contribution of nonreactive elements such as resistors, sources, etc., and the term $\frac{d}{dt}\mathbf{q}(\mathbf{z},t)$ represents the contribution of reactive elements such as capacitors and inductors.

Most of the circuit simulations of interest can be performed starting from this formulation. The small-signal analysis of circuits is an important approximation technique that reduces the simulation of a nonlinear dynamic circuit to the analysis of a nonlinear resistive circuit followed by the analysis of a linear dynamic circuit. We assume that all time-varying elements in the circuit are independent sources. This assumption allows us to simplify (1) to

$$\mathbf{f}(\mathbf{z}) + \frac{d}{dt}\mathbf{q}(\mathbf{z}) - \mathbf{e}(t) = 0. \tag{2}$$

Furthermore, we assume that the excitation term in (2), $\mathbf{e}(t)$, is "small", and thus all circuit variables $\mathbf{z}$ will be "close" to equilibrium values, $\mathbf{z}_0$, which represent the DC operating point of the circuit. The DC operating point can be computed by passivizing all time-varying elements, i.e., setting $\mathbf{e}(t) = 0$ and solving (2). With no time-varying elements, the second term in (2) vanishes, and the operating point $\mathbf{z}_0$ is found as the solution of the system of nonlinear equations

$$\mathbf{f}(\mathbf{z}_0) = 0. \tag{3}$$

By expanding the terms $\mathbf{f}(\mathbf{z})$ and $\mathbf{q}(\mathbf{z})$ in the circuit equations (2) into Taylor series about the operating point $\mathbf{z}_0$, we obtain

$$\left[\mathbf{f}(\mathbf{z}_0) + \left.\frac{\partial \mathbf{f}}{\partial \mathbf{z}}\right|_{\mathbf{z}_0} \delta\mathbf{z} + \cdots\right] + \frac{d}{dt}\left[\mathbf{q}(\mathbf{z}_0) + \left.\frac{\partial \mathbf{q}}{\partial \mathbf{z}}\right|_{\mathbf{z}_0} \delta\mathbf{z} + \cdots\right] - \mathbf{e}(t) = 0. \tag{4}$$

Since we assumed that all the deviations from the equilibrium values are "small", all but the first-order terms can be neglected in (4). Using $\mathbf{f}(\mathbf{z}_0) = 0$ and $\frac{d}{dt}\mathbf{q}(\mathbf{z}_0) = 0$, and setting

$$\mathbf{G} = \left.\frac{\partial \mathbf{f}}{\partial \mathbf{z}}\right|_{\mathbf{z}_0} \quad \text{and} \quad \mathbf{C} = \left.\frac{\partial \mathbf{q}}{\partial \mathbf{z}}\right|_{\mathbf{z}_0}, \tag{5}$$

we obtain from (4) the linear dynamic system

$$\mathbf{G}\,\delta\mathbf{z} + \mathbf{C}\,\frac{d}{dt}\delta\mathbf{z} - \mathbf{e}(t) = 0 \tag{6}$$

that needs to be solved as part of the small-signal analysis.

Since the system (6) is linear, the superposition law holds, and hence all sources of excitation can be treated separately. Therefore, without loss of generality, we can assume that there is only one source of excitation, $\mathbf{e}(t) = \mathbf{b}u(t)$. Then, the small-signal linear time-invariant dynamic system (6) is described by the following system of first-order differential equations:

$$\begin{aligned} \mathbf{C}\,\frac{d}{dt}\mathbf{x} &= -\mathbf{G}\,\mathbf{x} + \mathbf{b}\,u, \\ y &= \mathbf{l}^T\mathbf{x}. \end{aligned} \tag{7}$$

Here, $\mathbf{x}$ represents the small-signal circuit variables, $\delta\mathbf{z}$, and y is the output of interest.

Our goal is to determine the impulse-response of the linear circuit with zero initial-conditions, which, in turn, can be used to compute the response to any excitation. We apply the Laplace transform to the system (7), assuming zero initial conditions. Then, from (7), we obtain

$$\begin{aligned} s\,\mathbf{C}\,\mathbf{X} &= -\mathbf{G}\,\mathbf{X} + \mathbf{b}\,U, \\ Y &= \mathbf{l}^T\mathbf{X}, \end{aligned} \tag{8}$$

where $\mathbf{X}$, U, and Y denote the Laplace transform of $\mathbf{x}$, u, and y, respectively. It follows from (8) that the Laplace-domain *impulse response*, or *transfer function*, defined as $H(s) = Y(s)/U(s)$, is given by

$$H(s) = \mathbf{l}^T\,(\mathbf{G} + s\mathbf{C})^{-1}\,\mathbf{b}. \tag{9}$$

The impulse response $H(s)$ can be accurately and efficiently approximated by means of the PVL algorithm.

3 The PVL algorithm

The PVL algorithm [1] uses the Lanczos process to compute a Padé approximation to the frequence response (9) in a numerically stable fashion. In this section, we review the PVL algorithm.

In the following, let $s_0 \in \mathbb{C}$ be an arbitrary, but fixed expansion point such that the matrix $\mathbf{G} + s_0\mathbf{C}$ is nonsingular. Here, $\mathbf{G}$ and $\mathbf{C}$ are the matrices from the small-signal linear

system (7). Our goal is to approximate the impulse response (9) in a region in the complex plane about the point s_0. Using the change of variables $s = s_0 + \sigma$ and setting

$$\mathbf{A} = -\left(\mathbf{G} + s_0\mathbf{C}\right)^{-1}\mathbf{C}, \quad \mathbf{r} = \left(\mathbf{G} + s_0\mathbf{C}\right)^{-1}\mathbf{b}, \tag{10}$$

we can rewrite (9) as follows:

$$\begin{aligned} H(s_0 + \sigma) &= \mathbf{l}^T\left(\sigma\mathbf{C} + \mathbf{G} + s_0\mathbf{C}\right)^{-1}\mathbf{b} \\ &= \mathbf{l}^T\left(\mathbf{I} - \sigma\mathbf{A}\right)^{-1}\mathbf{r}. \end{aligned} \tag{11}$$

Here $\mathbf{I}$ denotes the identity matrix.

We now sketch the PVL algorithm for accurately approximating the function (11). For a detailed description of PVL, we refer the reader to [1].

3.1 System reduction by Padé approximation

By expanding $H(s_0 + \sigma)$ into a Taylor series about $\sigma = 0$, we obtain from (11) the following representation:

$$H(s_0 + \sigma) = \mathbf{l}^T\left(\mathbf{I} + \sigma\mathbf{A} + \sigma^2\mathbf{A}^2 + \cdots\right)\mathbf{r} = \sum_{k=0}^{\infty} m_k\sigma^k, \tag{12}$$

where

$$m_k = \mathbf{l}^T\mathbf{A}^k\mathbf{r}, \quad k = 0, 1\ldots, \tag{13}$$

are the so-called *moments* of the transfer function.

Let $q \geq 1$ be an arbitrary integer. A rational function of the form

$$H_q(s_0 + \sigma) = \frac{b_0 + b_1\sigma + \cdots + b_{q-1}\sigma^{q-1}}{1 + a_1\sigma + a_2\sigma^2 + \cdots + a_q\sigma^q}, \tag{14}$$

with coefficients $a_1, a_2, \ldots, a_q, b_0, b_1, \ldots, b_{q-1} \in \mathbb{C}$, is called a qth *Padé approximant* (or *Padé approximant of order* q) to the impulse response $H(s_0 + \sigma)$ if the Taylor series of H and H_q about $\sigma = 0$ agree in at least the first $2q$ terms, i.e.,

$$\begin{aligned} H_q(s_0 + \sigma) &= \sum_{k=0}^{2q-1} m_k\sigma^k + \mathcal{O}(s^{2q}) \\ &= H(s_0 + \sigma) + \mathcal{O}(s^{2q}). \end{aligned} \tag{15}$$

Note that the condition (15) just represents $2q$ equations for the $2q$ free parameters $a_1, a_2, \ldots, a_q, b_0, b_1, \ldots, b_{q-1}$ in (14), which suggests that a qth Padé approximant H_q always exists. Indeed, it can be shown that—except for certain degenerate cases—the function H_q exists and is unique; see, e.g., [5].

It turns out that Padé approximation is a very powerful tool for the analysis of large linear circuits. This was first demonstrated by Pillage and Rohrer [6] with their asymptotic

waveform evaluation (AWE) approach that is based on the computation of the Padé approximant H_q; for a detailed description of AWE, we refer the reader to [7, 8] and the references given there.

In AWE, the Padé approximant H_q is directly generated from the moments (13). Unfortunately, this procedure is inherently numerically unstable, as we illustrated in [1]. In fact, this is the reason why, in practice, AWE can be used only for fairly moderate values of q. In [1], we proposed a different algorithm, called PVL, for computing H_q. The PVL algorithm bypasses the moments (13), and instead, PVL exploits the connection [9] between Padé approximation and the Lanczos process [2] to generate the Padé approximant H_q in a numerically stable manner.

3.2 The Lanczos algorithm

The classical Lanczos algorithm (applied to the matrix $\mathbf{A}$ and the vectors $\mathbf{l}$, $\mathbf{r}$ from (11)) can be stated as follows.

Algorithm 1 (Lanczos algorithm [2])

0) Set $\mathbf{v} = \mathbf{r}$, $\mathbf{w} = \mathbf{l}$, $\mathbf{v}_0 = \mathbf{w}_0 = 0$, and $\delta_0 = 1$.

For $n = 1, 2, \ldots, q$ do:

1) Compute $\rho_n = \|\mathbf{v}\|_2$ and $\eta_n = \|\mathbf{w}\|_2$.
   If $\rho_n = 0$ or $\eta_n = 0$, then stop.

2) Set

$$\mathbf{v}_n = \frac{\mathbf{v}}{\rho_n}, \quad \mathbf{w}_n = \frac{\mathbf{w}}{\eta_n},$$

$$\delta_n = \mathbf{w}_n^T \mathbf{v}_n, \tag{16}$$

$$\alpha_n = \frac{\mathbf{w}_n^T \mathbf{A} \mathbf{v}_n}{\delta_n}, \quad \beta_n = \eta_n \frac{\delta_n}{\delta_{n-1}}, \quad \gamma_n = \rho_n \frac{\delta_n}{\delta_{n-1}}, \tag{17}$$

$$\mathbf{v} = \mathbf{A}\mathbf{v}_n - \mathbf{v}_n \alpha_n - \mathbf{v}_{n-1} \beta_n,$$

$$\mathbf{w} = \mathbf{A}^T \mathbf{w}_n - \mathbf{w}_n \alpha_n - \mathbf{w}_{n-1} \gamma_n.$$

We remark that, in Algorithm 1, a breakdown, triggered by division by 0, will occur if one encounters $\delta_n = 0$ in (17). Furthermore, division by a nonzero yet small number $\delta_n \approx 0$ in (17) may result in numerical instabilities. However, we stress that these problems can be remedied by using a so-called look-ahead variant of the Lanczos algorithm, see [10, 11]. In fact, in our implementation of the PVL algorithm we employ the look-ahead Lanczos algorithm described in [11].

The Lanczos algorithm has numerous interesting properties and applications; we refer the reader to [10, 11] and the references given there. The key property of Algorithm 1 is that it produces a very useful small-dimensional approximation, called $\mathbf{T}_q$ in the sequel, to the (usually large-dimensional) matrix $\mathbf{A}$. More precisely, $\mathbf{T}_q$ is the $q \times q$ tridiagonal matrix

$$\mathbf{T}_q = \begin{bmatrix} \alpha_1 & \beta_2 & 0 & \cdots & 0 \\ \rho_2 & \alpha_2 & \beta_3 & \ddots & \vdots \\ 0 & \rho_3 & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & \beta_q \\ 0 & \cdots & 0 & \rho_q & \alpha_q \end{bmatrix}, \tag{18}$$

where ρ_n, α_n, and β_n are the quantities generated by Algorithm 1. The matrix $\mathbf{T}_q$ is—in some sense—the best $q \times q$ approximation to $\mathbf{A}$. We stress that this approximation is very good even if q is much smaller than the order N of the $N \times N$ matrix $\mathbf{A}$. In fact, the Lanczos algorithm is mostly applied to very large matrices, and typically, $q \ll N$.

It turns out that the Lanczos tridiagonal matrix $\mathbf{T}_q$ contains all the information that is needed to generate the qth Padé approximant H_q. The rational function H_q is given by

$$H_q(s_0 + \sigma) = \mathbf{l}^T \mathbf{r} \cdot \mathbf{e}_1^T \left(\mathbf{I} - \sigma \mathbf{T}_q \right)^{-1} \mathbf{e}_1, \tag{19}$$

where $\mathbf{e}_1 = \begin{bmatrix} 1 & 0 & \cdots & 0 \end{bmatrix}^T \in \mathbb{R}^q$ is the first unit vector in $\mathbb{R}^q$. Furthermore, the *poles* of H_q, i.e., the zeros of the denominator polynomial in (14), are just the inverse eigenvalues of $\mathbf{T}_q$. Indeed, by rewriting the expression (19) of H_q in terms of the eigendecomposition $\mathbf{T}_q = \mathbf{S}_q \mathbf{\Lambda}_q \mathbf{S}_q^{-1}$ of the Lanczos matrix $\mathbf{T}_q$, we get

$$H_q(s_0 + \sigma) = \mathbf{l}^T \mathbf{r} \cdot \underbrace{\mathbf{e}_1^T \mathbf{S}_q}_{= \mu^T} \left(\mathbf{I} - \sigma \mathbf{\Lambda}_q \right)^{-1} \underbrace{\mathbf{S}_q^{-1} \mathbf{e}_1}_{= \nu} = \sum_{j=1}^{q} \frac{\mathbf{l}^T \mathbf{r} \cdot \mu_j \nu_j}{1 - \sigma \lambda_j}. \tag{20}$$

Here, $\mathbf{\Lambda}_q = \operatorname{diag}(\lambda_1, \lambda_1, \ldots, \lambda_q)$ contains the eigenvalues of $\mathbf{T}_q$, and μ_j and ν_j are the components of the vectors μ and ν. Note that, from (20), we immediately obtain the so-called *pole/residue representation* of the Padé approximant:

$$H_q(s_0 + \sigma) = k_\infty + \sum_{\substack{j=1 \\ \lambda_j \neq 0}}^{q} \frac{-\mathbf{l}^T \mathbf{r} \cdot \mu_j \nu_j / \lambda_j}{\sigma - 1/\lambda_j} \tag{21}$$

The term k_∞ in (21) may result if one of the eigenvalues of $\mathbf{T}_q$ is zero.

Finally, we remark that the *zeros* of the reduced-order model H_q, i.e., the zeros of the numerator polynomial in (14), can also be computed easily from the Lanczos matrix $\mathbf{T}_q$. In fact, it can be shown that

$$H_q(s_0 + \sigma) = \mathbf{l}^T \mathbf{r} \, \frac{\det \left(\mathbf{I} - \sigma \widetilde{\mathbf{T}}_q \right)}{\det \left(\mathbf{I} - \sigma \mathbf{T}_q \right)}, \tag{22}$$

where

$$\widetilde{\mathbf{T}}_q := \begin{bmatrix} \alpha_2 & \beta_3 & 0 & \cdots & 0 \\ \rho_3 & \alpha_3 & \ddots & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & \beta_q \\ 0 & \cdots & 0 & \rho_q & \alpha_q \end{bmatrix} \tag{23}$$

is the matrix obtained from $\mathbf{T}_q$ by deleting the first row and column. By (22), the zeros of H_q are just the inverses of the eigenvalues of $\widetilde{\mathbf{T}}_q$.

3.3 A sketch of PVL

The computational procedure based on the Padé/Lanczos connection is the PVL algorithm. It can be sketched as follows.

Algorithm 2 (Sketch of the PVL algorithm)

1) Run q steps of the Lanczos process (Algorithm 1) to obtain the tridiagonal matrix $\mathbf{T}_q$.

2) Compute an eigendecomposition

$$\mathbf{T}_q = \mathbf{S}_q \operatorname{diag}(\lambda_1, \lambda_1, \ldots, \lambda_q)\mathbf{S}_q^{-1} \tag{24}$$

of $\mathbf{T}_q$, and set

$$\mu = \mathbf{S}_q^T \mathbf{e}_1 \quad \text{and} \quad \nu = \mathbf{S}_q^{-1}\mathbf{e}_1. \tag{25}$$

3) Compute the poles and residues of H_q by setting

$$\begin{aligned} p_j &= 1/\lambda_j \quad \text{and} \quad k_j = \frac{\mathbf{l}^T\mathbf{r} \cdot \mu_j \nu_j}{\lambda_j} \quad \text{for all} \quad j = 1, 2, \ldots, q \quad \text{with} \quad \lambda_j \neq 0, \\ k_\infty &= \sum_{\substack{j=0 \\ \lambda_j = 0}}^{q} \mathbf{l}^T\mathbf{r} \cdot \mu_j \nu_j. \end{aligned} \tag{26}$$

We remark that the PVL algorithm and AWE require roughly the same amount of computational work. As in AWE, the dominating cost is the computation of the LU factorization

$$\mathbf{G} + s_0\mathbf{C} = \mathbf{L}\mathbf{U}, \tag{27}$$

which needs to be computed only once. Based on (27), the vectors $\mathbf{A}\mathbf{v}_n$ and $\mathbf{A}^T\mathbf{w}_n$ required in step 2) of Algorithm 1 are obtained as follows. First, using forward-backward substitution, we solve

$$\mathbf{L}\mathbf{U}\mathbf{z} = -\mathbf{C}\mathbf{v}_n \quad \text{and} \quad \mathbf{U}^T\mathbf{L}^T\mathbf{y} = -\mathbf{w}_n \tag{28}$$

for $\mathbf{z}$ and $\mathbf{y}$, and then, we set

$$\mathbf{A}\mathbf{v}_n = \mathbf{z} \quad \text{and} \quad \mathbf{A}^T\mathbf{w}_n = \mathbf{C}^T\mathbf{y}. \tag{29}$$

Hence, the PVL algorithm involves $2q$ forward-backward substitutions to generate the qth Padé approximant, which is the same as in AWE.

4 Computing sensitivities with PVL

In [12], the AWE algorithm was extended to compute sensitivities of poles and zeros of the impulse response. In this section, we show how the PVL algorithm can be used to generate in a numerically stable manner sensitivities of the impulse response and related quantities. We note that the advantages of PVL over AWE are preserved in sensitivity computations, and our method is not restricted to low-order Padé approximants.

Our goal is the calculation of sensitivities of circuit quantities with respect to some parameter $p \in \mathbb{R}$. In the following, we always use $'$ to denote derivatives with respect to p. Moreover, we assume that the matrices $\mathbf{C}$, $\mathbf{G}$ and the vectors $\mathbf{b}$, $\mathbf{l}$ in (7) are differentiable with respect to p. This guarantees that the quantities $\mathbf{A}$, $\mathbf{l}$, and $\mathbf{r}$, which define the impulse response $H(s_0 + \sigma)$ in (11), are also differentiable, and we denote by

$$\mathbf{A}' = \frac{\partial \mathbf{A}}{\partial p}, \quad \mathbf{l}' = \frac{\partial \mathbf{l}}{\partial p}, \quad \text{and} \quad \mathbf{r}' = \frac{\partial \mathbf{r}}{\partial p} \tag{30}$$

their derivatives.

4.1 An extended Lanczos algorithm

Recall from (19) that the PVL algorithm computes the Padé approximation $H_q(s_0 + \sigma)$ to the impulse response $H(s_0 + \sigma)$ via the Lanczos tridiagonal matrix $\mathbf{T}_q$ given in (18). It order to obtain sensitivities related to H_q, we therefore need the derivative, $\mathbf{T}_q'$, of the matrix $\mathbf{T}_q$. First, we note that—since $\mathbf{A}$, $\mathbf{l}$, and $\mathbf{r}$ are differentiable—all quantities generated by the Lanczos Algorithm 1 are also differentiable, and thus, by (18), we have

$$\mathbf{T}_q' = \begin{bmatrix} \alpha_1' & \beta_2' & 0 & \cdots & 0 \\ \rho_2' & \alpha_2' & \beta_3' & \ddots & \vdots \\ 0 & \rho_3' & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & \beta_q' \\ 0 & \cdots & 0 & \rho_q' & \alpha_q' \end{bmatrix}. \tag{31}$$

Moreover, the entries ρ_n', α_n', and β_n' of $\mathbf{T}_n'$ can be computed by means of an "extended" version of the Lanczos algorithm that is obtained by differentiating all equations in Algorithm 1 and adding the differentiated relations to Algorithm 1; the resulting extended Lanczos process is stated as Algorithm 3 below. For example, differentiating (16) gives

$$\delta_n' = \mathbf{w}_n^T \mathbf{v}_n' + (\mathbf{w}_n')^T \mathbf{v}_n, \tag{32}$$

and differentiating the first relation in (17) gives

$$\alpha_n' = \frac{(\mathbf{w}_n')^T \mathbf{A}\mathbf{v}_n + \mathbf{w}_n^T (\mathbf{A}\mathbf{v}_n)'}{\delta_n} - \mathbf{w}_n^T \mathbf{A}\mathbf{v}_n \frac{\delta_n'}{\delta_n^2}. \tag{33}$$

Note that, by means of (32) and the formula for α_n in (17), we can rewrite (33) in the form

$$\begin{aligned} \alpha'_n &= \frac{(\mathbf{w}'_n)^T \mathbf{A}\mathbf{v}_n + \mathbf{w}_n^T (\mathbf{A}\mathbf{v}_n)' - \alpha_n \left(\mathbf{w}_n^T \mathbf{v}'_n + (\mathbf{w}'_n)^T \mathbf{v}_n\right)}{\delta_n} \\ &= \frac{\mathbf{w}_n^T \left((\mathbf{A}\mathbf{v}_n)' - \mathbf{v}'_n \alpha_n\right) + (\mathbf{w}'_n)^T (\mathbf{A}\mathbf{v}_n - \mathbf{v}_n \alpha_n)}{\delta_n}, \end{aligned} \tag{34}$$

which is just the equation (36) in the following extended Lanczos algorithm.

Algorithm 3 (Extended Lanczos algorithm, including sensitivity computations)

0) Set $\mathbf{v} = \mathbf{r}$, $\mathbf{w} = \mathbf{l}$, $\mathbf{v}' = \mathbf{r}'$, $\mathbf{w}' = \mathbf{l}'$, $\mathbf{v}_0 = \mathbf{w}_0 = \mathbf{v}'_0 = \mathbf{w}'_0 = 0$, $\delta_0 = 1$, and $\delta'_0 = 0$.

For $n = 1, 2, \ldots, q$ do:

1) Compute $\rho_n = \|\mathbf{v}\|_2$ and $\eta_n = \|\mathbf{w}\|_2$.
   If $\rho_n = 0$ or $\eta_n = 0$, then stop.

2) Set

$$\begin{aligned} \mathbf{v}_n &= \frac{\mathbf{v}}{\rho_n}, \quad \mathbf{w}_n = \frac{\mathbf{w}}{\eta_n}, \\ \rho'_n &= \mathbf{v}_n^T \mathbf{v}', \quad \eta'_n = \mathbf{w}_n^T \mathbf{w}', \\ \mathbf{v}'_n &= \frac{\mathbf{v}' - \mathbf{v}_n \rho'_n}{\rho_n}, \quad \mathbf{w}'_n = \frac{\mathbf{w}' - \mathbf{w}_n \eta'_n}{\eta_n}, \\ \delta_n &= \mathbf{w}_n^T \mathbf{v}_n, \quad \delta'_n = \mathbf{w}_n^T \mathbf{v}'_n + \mathbf{v}_n^T \mathbf{w}'_n, \\ \alpha_n &= \frac{\mathbf{w}_n^T \mathbf{A}\mathbf{v}_n}{\delta_n}, \quad \beta_n = \eta_n \frac{\delta_n}{\delta_{n-1}}, \quad \gamma_n = \rho_n \frac{\delta_n}{\delta_{n-1}}, \\ \mathbf{v} &= \mathbf{A}\mathbf{v}_n - \mathbf{v}_n \alpha_n - \mathbf{v}_{n-1} \beta_n, \\ \mathbf{w} &= \mathbf{A}^T \mathbf{w}_n - \mathbf{w}_n \alpha_n - \mathbf{w}_{n-1} \gamma_n, \\ \beta'_n &= \eta'_n \frac{\delta_n}{\delta_{n-1}} + \beta_n \left(\frac{\delta'_n}{\delta_n} - \frac{\delta'_{n-1}}{\delta_{n-1}}\right), \quad \gamma'_n = \rho'_n \frac{\delta_n}{\delta_{n-1}} + \gamma_n \left(\frac{\delta'_n}{\delta_n} - \frac{\delta'_{n-1}}{\delta_{n-1}}\right), \\ \mathbf{t}_1 &= (\mathbf{A}\mathbf{v}_n)' - \mathbf{v}'_n \alpha_n, \quad \mathbf{t}_2 = \left(\mathbf{A}^T \mathbf{w}_n\right)' - \mathbf{w}'_n \alpha_n, \qquad (35) \\ \alpha'_n &= \frac{\mathbf{w}_n^T \mathbf{t}_1 + (\mathbf{w}'_n)^T (\mathbf{A}\mathbf{v}_n - \mathbf{v}_n \alpha_n)}{\delta_n}, \qquad (36) \\ \mathbf{v}' &= \mathbf{t}_1 - \mathbf{v}_n \alpha'_n - \mathbf{v}_{n-1} \beta'_n - \mathbf{v}'_{n-1} \beta_n, \\ \mathbf{w}' &= \mathbf{t}_2 - \mathbf{w}_n \alpha'_n - \mathbf{w}_{n-1} \gamma'_n - \mathbf{w}'_{n-1} \gamma_n. \end{aligned}$$

4.2 Sensitivities of the impulse response

For sensitivity computation in PVL, we run q steps of the extended Lanczos Algorithm 3, instead of Algorithm 1. As a result, we obtain both the Lanczos tridiagonal matrix $\mathbf{T}_q$ and its derivative $\mathbf{T}'_q$. Using these two matrices, we can compute all sensitivities of interest.

First, consider the sensitivity of the approximate impulse response H_q. By differentiating the representation (19) of H_q, we obtain

$$\begin{aligned} H'_q(s_0+\sigma) &= \frac{\partial}{\partial p} H_q(s_0+\sigma) \\ &= \mathbf{e}_1^T \left(\mathbf{I}-\sigma\mathbf{T}_q\right)^{-1} \left[\left(\mathbf{l}^T\mathbf{r}' + \mathbf{r}^T\mathbf{l}'\right)\, \mathbf{e}_1 \,+\, \sigma\cdot\mathbf{l}^T\mathbf{r}\cdot\mathbf{T}'_q \left(\mathbf{I}-\sigma\mathbf{T}_q\right)^{-1}\mathbf{e}_1\right]. \end{aligned} \tag{37}$$

Recall from Algorithm 2 that, in PVL, we also compute the eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_q$ and an associated eigenvector matrix $\mathbf{S}_q$ such that

$$\mathbf{T}_q = \mathbf{S}_q \,\mathrm{diag}(\lambda_1, \lambda_1, \ldots, \lambda_q)\mathbf{S}_q^{-1}. \tag{38}$$

Furthermore, in PVL we have the vectors

$$\mu = \begin{bmatrix} \mu_1 \\ \mu_2 \\ \vdots \\ \mu_q \end{bmatrix} = \mathbf{S}_q^T\mathbf{e}_1 \quad \text{and} \quad \nu = \begin{bmatrix} \nu_1 \\ \nu_2 \\ \vdots \\ \nu_q \end{bmatrix} = \mathbf{S}_q^{-1}\mathbf{e}_1. \tag{39}$$

available. Using (38) and (39), we obtain from (37) the following computationally more economic formula:

$$H'_q(s_0+\sigma) = \sum_{j=1}^{q} \frac{\mu_j}{1-\sigma\lambda_j} \left[\left(\mathbf{l}^T\mathbf{r}' + \mathbf{r}^T\mathbf{l}'\right)\nu_j \,+\, \sigma\cdot\mathbf{l}^T\mathbf{r}\cdot\sum_{k=1}^{q}\frac{\nu_k c_{jk}}{1-\sigma\lambda_k}\right], \tag{40}$$

where the coefficients c_{jk} are the entries of the matrix

$$\mathbf{C} = [\,c_{jk}\,]_{j,k=1,2,\ldots,q} = \mathbf{S}_q^{-1}\,\mathbf{T}'_q\,\mathbf{S}_q. \tag{41}$$

Recall from (26) that the poles p_j of H_q are given as $p_j = 1/\lambda_j$, where λ_j, $j = 1, 2, \ldots, q$, are just the eigenvalues of $\mathbf{T}_q$. Therefore, the sensitivity p'_j of the jth pole p_j is given by

$$p'_j = -\frac{\lambda'_j}{\lambda_j^2}. \tag{42}$$

Thus, it only remains to show how to obtain the sensitivity λ'_j of the jth eigenvalue λ_j of $\mathbf{T}_q$. To this end, we use standard perturbation theory for eigenvalues; see, e.g., [13, 14]. First, for simplicity, we assume that λ_j is a simple eigenvalue of $\mathbf{T}_q$. Then from the results in [13] it follows that

$$\lambda'_j = \mathbf{y}_j^T\,\mathbf{T}'_q\,\mathbf{s}_j, \tag{43}$$

where $\mathbf{s}_j$ is the jth column of the eigenvector matrix $\mathbf{S}_q$ in (38) and $\mathbf{y}_j^T$ is the jth row of $\mathbf{S}_q^{-1}$. The case that λ_j is a multiple eigenvalue can be treated similarly, see [14]. The eigenvalue sensitivities are then given as the eigenvalues of the matrix

$$\left(\mathbf{Y}^{(j)}\right)^T \mathbf{T}_q' \, \mathbf{S}^{(j)}. \tag{44}$$

Here, $\mathbf{S}^{(j)}$ is the submatrix of the matrix $\mathbf{S}_q$ in (38) that contains all vectors associated with the multiple eigenvalue λ_j, and $\left(\mathbf{Y}^{(j)}\right)^T$ is the corresponding submatrix of $\mathbf{S}_q^{-1}$.

Finally, we note that sensitivities of the zeros of H_q can be computed in an analogous manner, by using the fact that the zeros are the inverse eigenvalues of the matrix $\widetilde{\mathbf{T}}_q$ given in (23).

4.3 Computing derivatives of circuit quantities

Algorithm 3 requires the computation of the derivatives of $\mathbf{A}$, $\mathbf{l}$, and $\mathbf{r}$. Recall that

$$\mathbf{A} = -\left(\mathbf{G} + s_0\mathbf{C}\right)^{-1}\mathbf{C}, \quad \mathbf{r} = \left(\mathbf{G} + s_0\mathbf{C}\right)^{-1}\mathbf{b}, \tag{45}$$

and $\mathbf{b}$ and $\mathbf{l}$ are vectors that depend on the nature of excitation and of the output of interest, respectively.

We need, therefore, to compute matrices $\mathbf{G}'$, $\mathbf{C}'$, and the vectors $\mathbf{b}'$ and $\mathbf{l}'$. Again, we denote by p the parameter with respect to which we want to compute sensitivities. Then, from (5), we get

$$\mathbf{G}' = \frac{\partial}{\partial p}\left[\frac{\partial \mathbf{f}}{\partial \mathbf{z}}(\mathbf{z}_0(p), p)\right] = \frac{\partial^2 \mathbf{f}}{\partial \mathbf{z}^2}\,\frac{\partial \mathbf{z}_0}{\partial p} + \frac{\partial^2 \mathbf{f}}{\partial p\,\partial \mathbf{z}} \tag{46}$$

and

$$\mathbf{C}' = \frac{\partial^2 \mathbf{q}}{\partial \mathbf{z}^2}\,\frac{\partial \mathbf{z}_0}{\partial p} + \frac{\partial^2 \mathbf{q}}{\partial p\,\partial \mathbf{z}}. \tag{47}$$

The vector $\partial \mathbf{z}_0/\partial p$ represents the sensitivity of the operating point with respect to the parameter p and can be obtained by solving the linear system that results from the differentiation of the operating point equations

$$\mathbf{f}(\mathbf{z}_0(p), p) = 0, \tag{48}$$

$$\frac{\partial \mathbf{f}}{\partial \mathbf{z}}\,\frac{\partial \mathbf{z}_0}{\partial p} + \frac{\partial \mathbf{f}}{\partial p} = 0. \tag{49}$$

In general the computation of sensitivities requires the knowledge of second-order derivatives of the device modeling equations. Since in most cases second-order derivatives are not provided by models, they must be computed either through automatic differentiation [15, 16], or by finite differences.

In some important special cases, second-order derivative information is easier to obtain. For example, when the parameter affects only the reactive part of the circuit equations the

Poles	Pole Sensitivities
$-8.68942 \cdot 10^1$	$8.66 \cdot 10^1$
$-6.46523 \cdot 10^5$	$1.433 \cdot 10^3$
$-2.05993 \cdot 10^7 \pm 6.07465 \cdot 10^6$	$-1.32 \cdot 10^5 \mp 1.861 \cdot 10^5$
$-3.50809 \cdot 10^7 \pm 2.76351 \cdot 10^7$	$6.7 \cdot 10^4 \pm 1.558 \cdot 10^5$
$-7.7302 \cdot 10^7$	$1.323 \cdot 10^5$
$-8.17495 \cdot 10^7$	$9.888 \cdot 10^3$
$-9.58123 \cdot 10^7 \pm 7.59525 \cdot 10^7$	$6.907 \cdot 10^5 \quad \pm 2.67 \cdot 10^7$

Table 1: Poles and Sensitivities for the 741 OpAmp, 18 PVL iterations

operating point is not dependent on the parameter p, and therefore the first term in (48) and (49) vanishes. Moreover, when the circuit is linear, then $\mathbf{G}'$ and $\mathbf{C}'$ can be easily obtained using stencils [4].

The vectors $\mathbf{b}$ and $\mathbf{l}$ depend in most cases solely on the topology of the circuit, and therefore their derivatives with respect to circuit parameters are zero. However, sometimes, the parameter p can affect the excitation of a circuit and cause $\mathbf{b}'$ to be nonzero, or an output function may result in a nonzero $\mathbf{l}'$.

Finally, we show how to obtain the vectors $(\mathbf{A}\mathbf{v}_n)'$ and $(\mathbf{A}^T\mathbf{w}_n)'$ in equation (35) of Algorithm 3. Recall that the vector $\mathbf{u} = \mathbf{A}\mathbf{v}_n$ is obtained by solving the linear system

$$(\mathbf{G} + s_0\mathbf{C})\,\mathbf{u} = -\mathbf{C}\mathbf{v}_n. \tag{50}$$

By differentiating (50), we see that the vector $\mathbf{u}' = (\mathbf{A}\mathbf{v}_n)'$ can be computed by solving the linear system

$$(\mathbf{G} + s_0\mathbf{C})\,\mathbf{u}' = -\mathbf{C}'\mathbf{v}_n - \mathbf{C}\mathbf{v}_n' - (\mathbf{G}' + s_0\mathbf{C}')\,\mathbf{u}. \tag{51}$$

A similar procedure is used to obtain $(\mathbf{A}^T\mathbf{w}_n)'$. Observe that both linear systems (50) and (51) have the same coefficient matrix $\mathbf{G} + s_0\mathbf{C}$, and therefore, no additional LU-factorization is required for sensitivity computations.

5 An illustrative example

In this section, we present numerical results for two typical examples.

Our first example is the 741 operational amplifier shown in Figure 2. The same example was analyzed in [12]. The magnitude and phase response of the operational amplifier obtained with PVL in 18 iterations are shown in Figures 3 and 4, respectively. They match with 5 digits of accuracy the response predicted by complex phasor analysis.

Table 1 shows the poles that converged after 18 PVL iterations, together with their normalized sensitivities with respect to the compensation capacitor (30pF).

A graphical representation of pole sensitivities is shown in Figure 5.

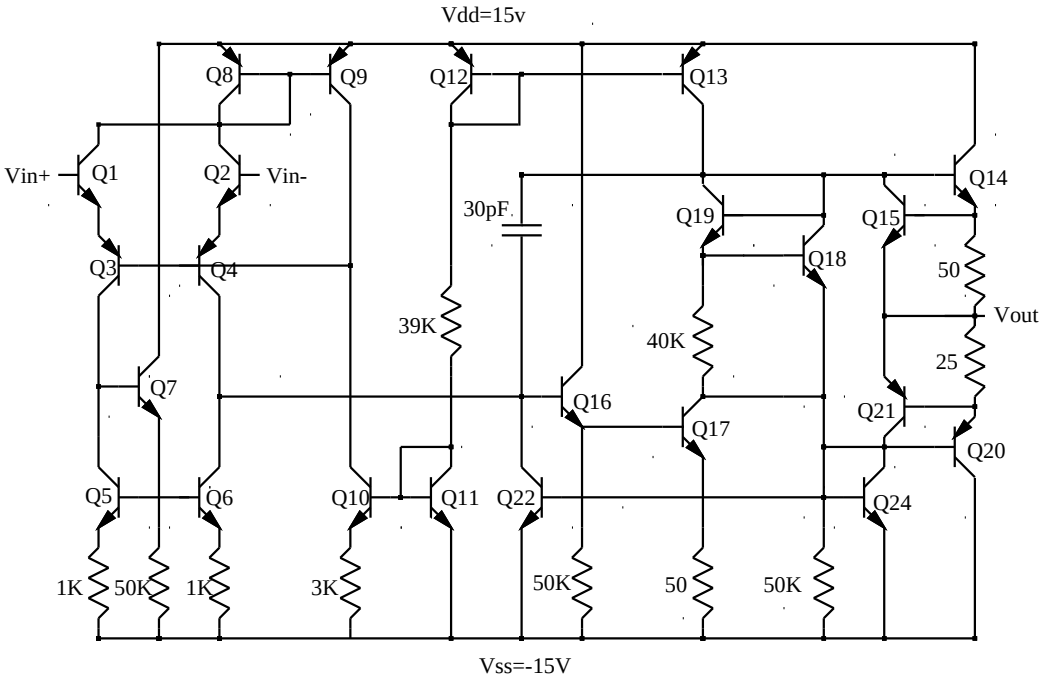

Figure 2: Schema of the 741 Operational Amplifier

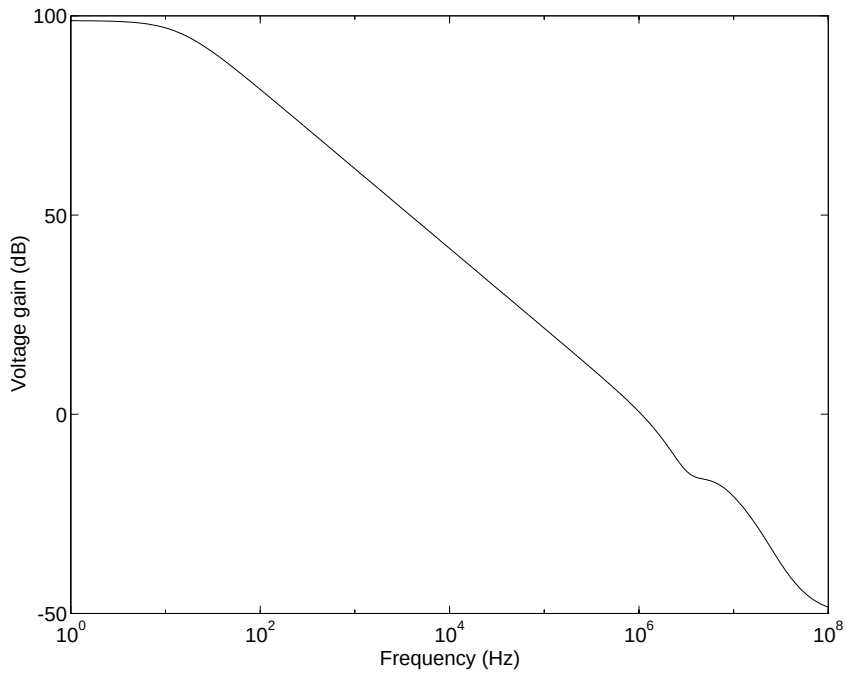

Figure 3: 741 Operational Amplifier magnitude response

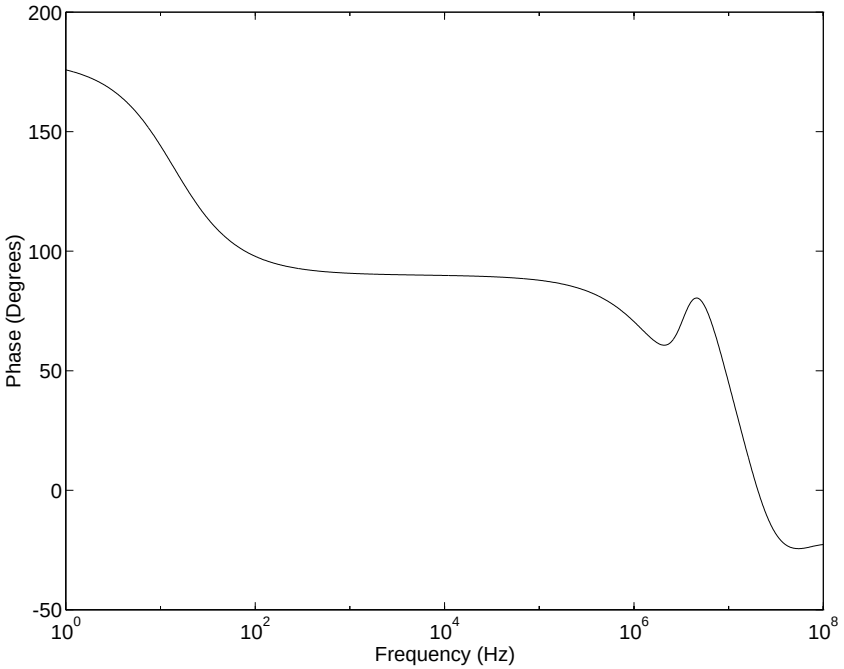

Figure 4: 741 Operational Amplifier phase response

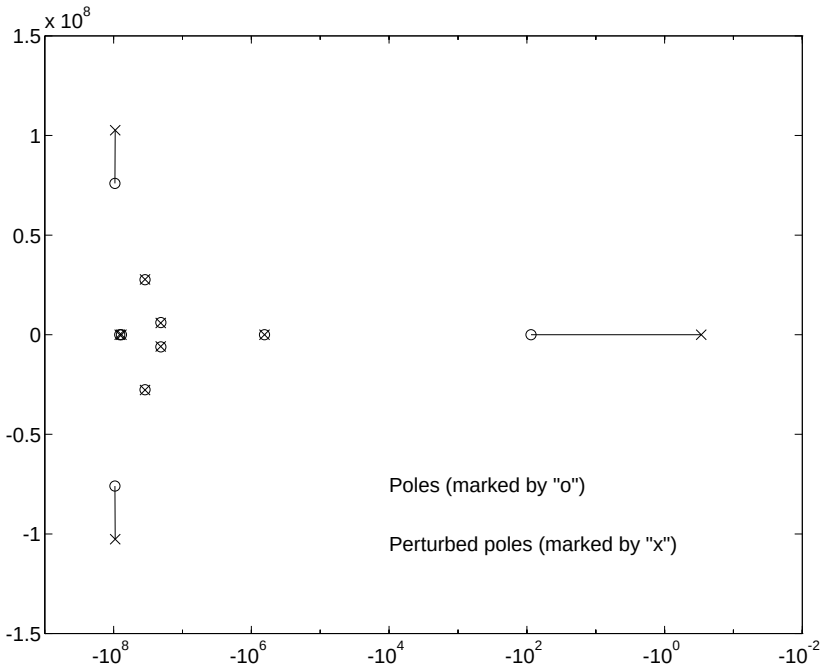

Figure 5: 741 Operational Amplifier pole sensitivities

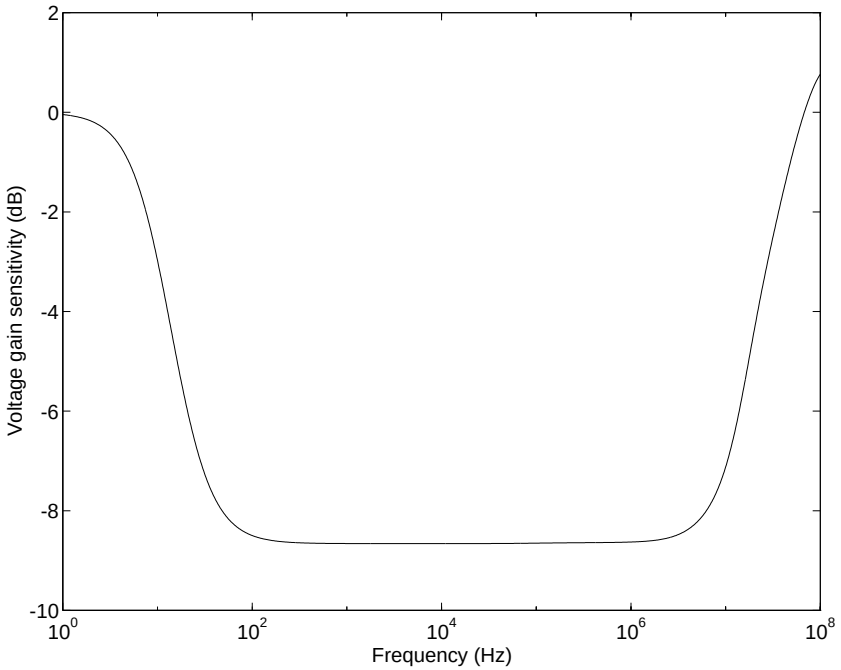

Figure 6: 741 Operational Amplifier magnitude sensitivities

Figures 6 and 7 show the normalized sensitivities of the magnitude and phase of the amplifier impulse response computed directly with the PVL algorithm. The responses match very well results obtained by perturbation over the entire frequency range. In contrast, the results published in [12] show discrepancies at higher frequencies.

Note that in this example the sensitivity parameter affects only a linear element in the reactive part of the circuit equations, therefore computation of second order derivatives was not required.

Our second example is a simulation of a low-noise amplifier designed for a radio-frequency application and implemented in an advanced BiCMOS process. The netlist, extracted from the actual layout with all the parasitics included, consists of 51 MOSFET devices, 26 bipolar transistors, 35 resistors, 6 inductors, and 381 capacitors. The circuit must operate over a range of temperatures and contains a sophisticated temperature stabilization scheme. Therefore, the sensitivity of the frequency response with respect to temperature is of considerable interest.

We study the frequency-domain characteristics of the amplifier gain—with and without temperature stabilization—and their sensitivities with respect to temperature. The two variants of the circuit were linearized around their DC operating points, and the resulting small-signal equivalent networks were analyzed, using the extended PVL algorithm.

A Padé approximant of order $q = 60$ models correctly the behavior of the amplifier from 10Hz to 10GHz. Figure 8 shows the magnitude of the circuit frequency-response, as produced by the algorithm, with the temperature stabilization circuitry turned successively

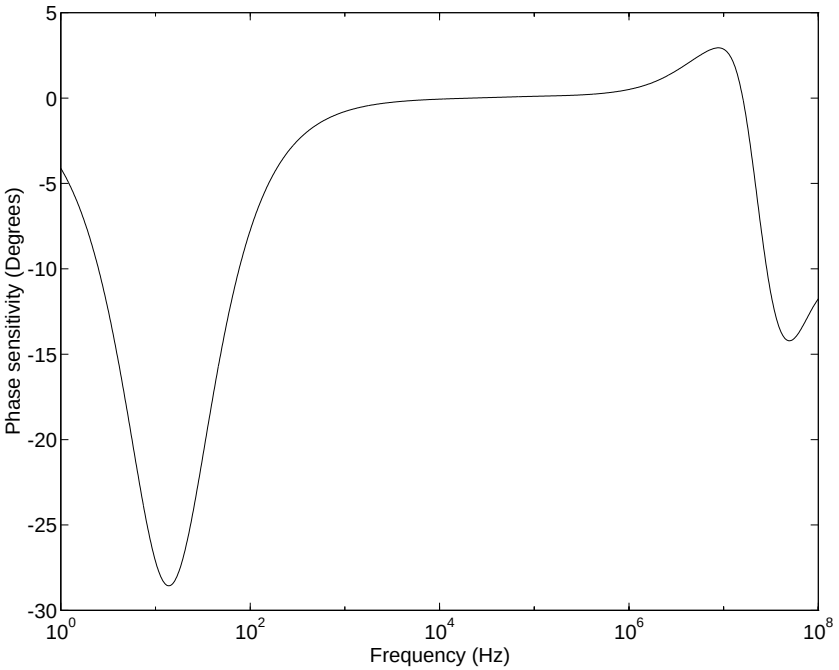

Figure 7: 741 Operational Amplifier phase sensitivities

on and off. Separately, we compared the Padé approximations to responses computed by complex phasor analysis and they were indistinguishable.

Figure 9 shows the normalized sensitivities (in dB) of the frequency-response magnitudes for the two circuit variants. The plots confirm that the temperature stabilization performs as expected, significantly reducing the sensitivity. Finally, Figure 10 shows only the normalized sensitivity of the temperature stabilized circuit at a more appropriate scale. Note that in this case the sensitivity must also account for the change in the operating point due to temperature. Since, in our present implementation, we cannot compute second-order derivatives, finite differences were used.

6 Conclusions

This paper argues that, due to its robustness and efficiency, PVL should become the algorithm of choice for the small-signal analysis of electrical circuits. The advantages of the PVL algorithm are not limited to superior accuracy and efficiency, but include new capabilities difficult to implement with existing methods. In this paper we extend the PVL algorithm to compute the sensitivity of network transfer functions and of its poles and zeros.

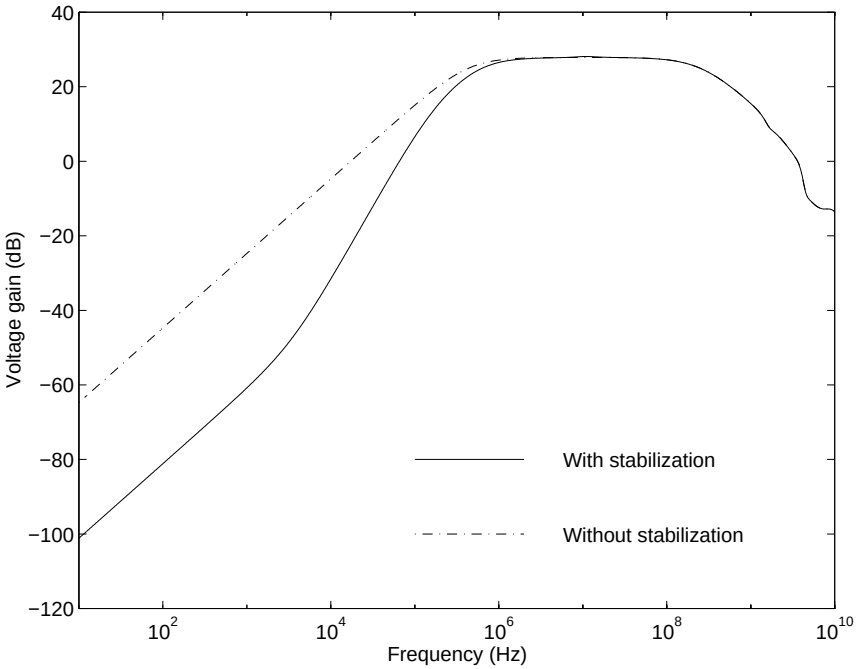

Figure 8: Low-Noise Amplifier magnitude responses

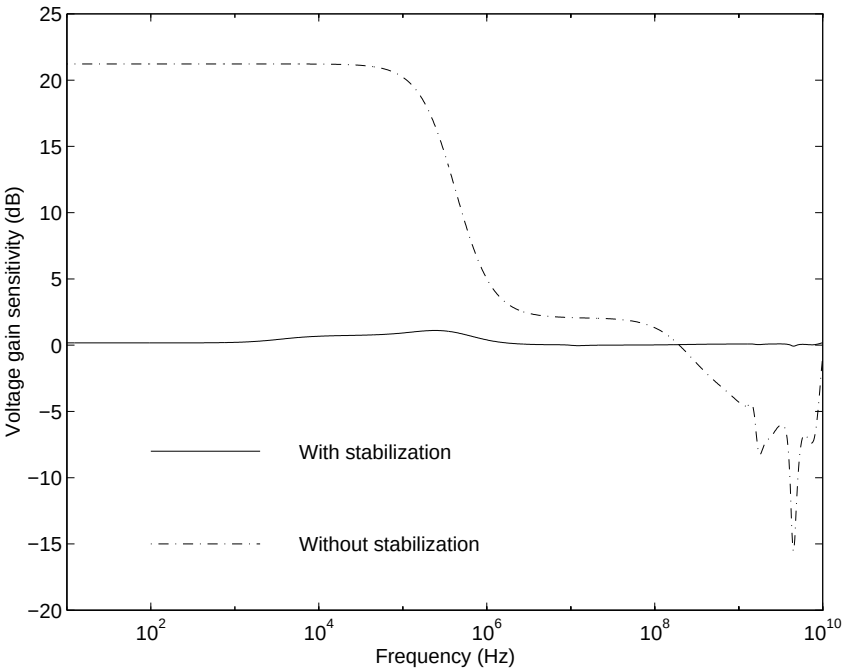

Figure 9: Low-Noise Amplifier magnitude sensitivities

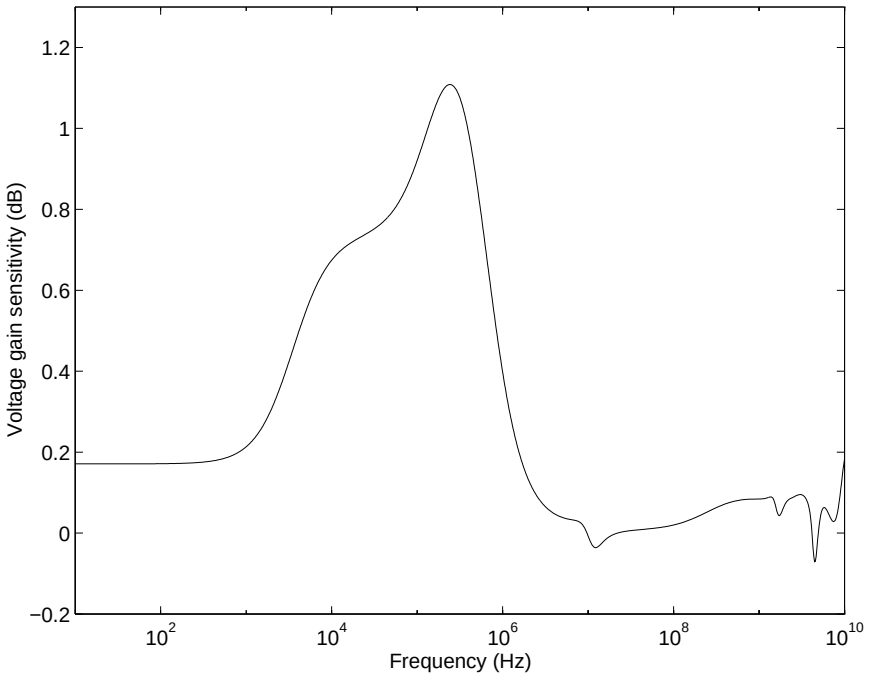

Figure 10: Low-Noise Amplifier magnitude sensitivity (stabilized)

References

[1] P. Feldmann and R.W. Freund. Efficient Linear Circuit Analysis by Padé Approximation via the Lanczos Process. Numerical Analysis Manuscript 94–01, AT&T Bell Laboratories, Murray Hill, NJ, February, 1994.

[2] C. Lanczos. An Iteration Method for the Solution of the Eigenvalue Problem of Linear Differential and Integral Operators. J. Res. Nat. Bur. Standards, Vol. 45, pp. 255–282, 1950.

[3] A.E. Ruehli. Equivalent Circuit Models for Three-Dimensional Multiconductor Systems. IEEE Trans. Microwave Theory and Tech., Vol. MTT-22, pp. 216–221, March, 1974.

[4] J. Vlach and K. Singhal. Computer Methods for Circuit Analysis and Design. Van Nostrand Reinhold, New York, N.Y., 1983.

[5] G.A. Baker, Jr. and P. Graves-Morris. Padé Approximants, Part I: Basic Theory. Addison-Wesley, Reading, MA, 1981.

[6] L.T. Pillage and R.A. Rohrer. Asymptotic Waveform Evaluation for Timing Analysis. IEEE Trans. Computer-Aided Design, Vol. CAD-9, pp. 352–366, April, 1990.

[7] V. Raghavan, R.A. Rohrer, L.T. Pillage, J.Y. Lee, J.E. Bracken, and M.M. Alaybeyi. AWE–Inspired. Proceedings of the IEEE 1993 Custom Integrated Circuits Conference, May, 1993.

[8] E. Chiprout and M.S. Nakhla. Asymptotic Waveform Evaluation and Moment Matching for Interconnect Analysis. Kluwer Academic Publishers, Norwell, MA, 1994.

[9] W.B. Gragg. Matrix Interpretations and Applications of the Continued Fraction Algorithm. Rocky Mountain J. Math., Vol. 4, pp. 213–225, 1974.

[10] R.W. Freund. The Look-Ahead Lanczos Process for Large Nonsymmetric Matrices and Related Algorithms. Linear Algebra for Large Scale and Real-Time Applications, pp. 137–163. Kluwer Academic Publishers, Dordrecht, The Netherlands, 1993.

[11] R.W. Freund, M.H. Gutknecht, and N.M. Nachtigal. An Implementation of the Look-Ahead Lanczos Algorithm for Non-Hermitian Matrices. SIAM J. Sci. Comput., Vol. 14, pp. 137–158, 1993.

[12] J.Y. Lee, X. Huang, and R.A. Rohrer. Pole and Zero Sensitivity Calculation in Asymptotic Waveform Evaluation. IEEE Trans. Computer-Aided Design, Vol. CAD-11, pp. 586–597, May, 1992.

[13] J. Stoer and R. Bulirsch. Introduction to Numerical Analysis, Second Edition. Springer-Verlag, New York, N.Y., 1993.

[14] T. Kato. A Short Introduction to Perturbation Theory for Linear Operators. Springer-Verlag, New York, N.Y., 1982.

[15] A. Griewank and G.F. Corliss (Eds.). Automatic Differentiation of Algorithms: Theory, Implementation, and Application. SIAM, Philadelphia, PA, 1991.

[16] P. Feldmann, R. Melville, and S. Moinian. Automatic Differentiation in Circuit Simulation and Device Modeling. Technical Digest of the 1992 IEEE/ACM International Conference on Computer-Aided Design, pp. 248–253, November, 1992.