Computation of the Periodic Steady-State Response of Nonlinear Networks by Extrapolation Methods

STIG SKELBOE, MEMBER, IEEE

Abstract—**The problem of computing the periodic steady-state response can be formulated as solving a nonlinear equation of the form $z=F(z)$ where $F(z)$ is the solution vector for the nonlinear network after one period of integration from the initial vector z. The convergence of the sequence $y_0, y_1, \cdots$ generated by $y_{r+1}=F(y_r)$ can be accelerated by extrapolation methods.**

This paper presents a unified analysis of three extrapolation methods: the scalar and vector ϵ-algorithms and the minimum polynomial extrapolation algorithm.

The main result of the paper is the theorem giving conditions for quadratic convergence of the extrapolation methods. To obtain this result the methods are studied for linear problems (where F is a linear function) and the error propagation properties are investigated.

For autonomous systems a function called G similar to F can be defined. In order to obtain quadratic convergence from the extrapolation methods, the derivatives of F and G must be Lipschitz continuous. The appendixes give sufficient conditions for the Lipschitz continuity.

A discussion of practical problems related to the implementation of the extrapolation methods is based on the convergence theorem and the error analysis.

The performance of the extrapolation methods is demonstrated and compared with other methods for steady-state analysis by four examples, two autonomous and two nonautonomous.

Extrapolation methods are very easy to implement, and they are efficient for the steady-state analysis of nonlinear circuits with few reactive elements giving rise to slowly decaying transients.

I. Introduction

A VERY large class of electrical circuits can be described by a system of N ordinary differential equations:

$$y' f(t,y), \quad y(t_0)=y_0, \qquad t\in[t_0,t_f]. \tag{1.1}$$

An important subclass of these problems has periodic solutions with known period. When (1.1) models an amplifier with a periodic forcing function, an example of this type occurs.

A periodic steady-state solution of the system (1.1) can be obtained (when it exists) by integrating from an appropriate initial value till the transients have died out. If the transients have disappeared after a few periods, this is the simplest and most efficient approach for the computation of the periodic steady-state solution.

Manuscript received May 22, 1978; revised April 25, 1979. This work was supported by the National Research Council of Canada, and the following Danish foundations: Philips Fond af 1958; Frants Allings Legat; and Statens Teknisk-Videnskabelige Forskningsraad.

The author is with Elektronikcentralen, Danish Research Centre for Applied Electronics, Horsholm, Denmark.

When the solution of (1.1) contains transients which are slowly changing relative to the period, the brute force approach for computation of the steady state outlined above will often imply the integration of hundreds of periods before the steady-state solution is reached. A number of different methods for accelerating the decay of the transients have been suggested, and in this paper the application of extrapolation will be discussed.

The steady-state problem can be formulated as an algebraic fixed point problem for a vector-valued vector function F. Let F be defined as the integration of one period, T of (1.1) such that

$$y(t_0+T)=F(y_0;t_0)$$

and in general:

$$y(t+T)=F(y(t);t), \qquad t\in[t_0,t_f-T].$$

The solution $y(t)$ of (1.1) will only be considered at the points $t_r=t_0+rT$, $r=0,1,\cdots$, and we have

$$y_{r+1}=F(y_r) \tag{1.2}$$

where the explicit t-dependence is omitted from F. In Appendix A a stricter definition of F is given.

The problem of computing the periodic steady-state response of (1.1) can be formulated as the problem of computing a fixed point z_0 of the function F, such that

$$z_0=F(z_0). \tag{1.3}$$

With the initial vector z_0 at t_0 the system (1.1) has a solution $z(t)$ where $z(t_0)=z_0$ and $z(t_0+T)=F(z_0)=z_0$ by definition of F. Thus $z(t)$ is the requested periodic steady-state solution.

The brute force approach to steady-state analysis, where (1.1) is integrated till the transients have disappeared, corresponds to the contraction principle $y_{r+1}=F(y_r)$ where $y_r\to z_0$ for $r\to\infty$.

The convergence of a sequence $y_0, y_1, \cdots$ generated by (1.2) can be accelerated by extrapolation [1]. In this paper various extrapolation techniques will be described and analyzed.

Newton's method can be used to solve (1.3) [2], [3], [9], and in [4] a generalization of the Newton iteration is presented. By this method an arbitrary number of periods of the transient can be skipped.

0098-4094/80/0300-0161$00.75

By defining a discrepancy vector $\delta(y_r)=F(y_r)-y_r$, an error measure $P(y_r)=\delta'(y_r)\delta(y_r)$ can be constructed. The error $P(y_r)$ is then minimized by a gradient method [5].

An entirely different approach to steady-state analysis is presented in [6]. It is based on Fourier expansion of the steady-state solution and computation of the Fourier coefficients by optimization. In [7] another method for steady-state analysis based on a Fourier expansion of the solution is described. In [8] it is described how extrapolation can be applied to Fourier coefficients computed at staggered time windows.

Appendix A gives sufficient conditions on the function f from (1.1) for F to have a Lipschitz continuous derivative $\underline{F}'$ so that the following expansion is valid:[1]

$$x_{r+1}-y_{r+1}=\underline{F}'(y_r)(x_r-y_r)+0(\|x_r-y_r\|^2) \quad (1.4)$$

where $x_r=x(t_0+rT)$, $y_r=y(t_0+rT)$, and $x(t),y(t)$ are solutions of the differential equation (1.1).

From (1.4) we find

$$y_r-z_0=\underline{F}'(z_0)(y_{r-1}-z_0)+0(\|y_{r-1}-z_0\|^2) \quad (1.5)$$

which implies that the convergence of the contraction principle (1.2) is very slow if $\|\underline{F}'(z_0)\|$ is close to 1.

By a reformulation of (1.5) we find

$$y_r-z_0=(\underline{I}-\underline{F}'(z_0))^{-1}(y_{r-1}-y_r)+0(\|y_{r-1}-z_0\|^2) \quad (1.6)$$

which shows that $\|y_r-z_0\|\gg\|y_{r-1}-y_r\|$ if $\underline{F}'(z_0)$ has eigenvalues close to 1. This is exactly the case for a problem with slowly decaying transients. The relations (1.5) and (1.6) show some of the most important problems in steady-state state analysis.

Section IV presents the Extrapolation Algorithm for the computation of z_0, the value of the steady-state solution $z(t)$ at t_0. In step n a vector $x^{(n+1)}$ is computed by extrapolation from the sequence $y_0,y_1,\cdots,y_{2m}$:

$$y_0=x^{(n)}$$
$$y_r=F(y_{r-1}), \qquad \text{for } r=1,2,\cdots,2m$$
$$x^{(n+1)}=\text{EXTRAP}\,(y_0,y_1,\cdots,y_{2m})$$

where EXTRAP means the scalar or vector ϵ-algorithm or the minimum polynomial method.[2]

From (1.5) we see that $y_1,y_2,\cdots$ is computed from a linear recursion apart from a perturbation $d_r=0(\|y_{r-1}-z_0\|^2)$. In Section II it is shown that the extrapolation methods being considered are exact:

$$z_0=\text{EXTRAP}\,(w_0,w_1,\cdots,w_{2m})$$

if $w_1,w_2,\cdots$ are generated by the linear recursion

$$w_r=z_0+\underline{A}(w_{r-1}-z_0). \quad (1.7)$$

This result is given in Corollary 2.1 and Theorem 2.4 for the ϵ-algorithms and the minimum polynomial method, respectively. The parameter m is found to be the number of nonzero eigenvalues of $\underline{A}$.

[1]The symbol $0(\|x-y\|^2)$ denotes a vector d where $\|d\|<K\|x-y\|^2$ for some $K>0$ and all $x,y\in D\subset R^N$.

[2]Actually, the minimum polynomial method is only applied to the sequence $y_0,y_1,\cdots,y_{m+1}$.

In Section III the propagation of errors through the extrapolation algorithms is studied, and it is shown that

$$\text{EXTRAP}\,(w_0+a_0,w_1+a_1,\cdots,w_{2m}+a_{2m})=z_0+0(\max_r\|a_r\|) \quad (1.8)$$

where the sequence $w_0,w_1,\cdots$ is generated by (1.7), and where $a_0,a_1,\cdots$ are "small" errors of the corresponding w-elements.

By comparing the nonlinear recursion formula (1.5) with the linear one (1.7) for $\underline{A}=\underline{F}'(z_0)$, the error propagation relation in (1.8) gives the main convergence result of this paper:

$$\text{EXTRAP}\,(y_0,y_1,\cdots,y_{2m})=z_0+0(\|y_0-z_0\|^2)$$

or

$$x^{(n+1)}-z_0=0(\|x^{(n)}-z_0\|^2).$$

This result establishes the quadratic convergence of the Extrapolation Algorithm given in Section IV.

In Section IV m is redefined to mean the number of eigenvalues of $\underline{F}'(z_0)$ "close" to 1, and it is discussed how the effect of the remaining nonzero eigenvalues can be removed by a low-pass filtering approach.

From the preceding discussion it is clear that the Lipschitz continuity of $\underline{F}'$ leading to (1.4) is necessary for quadratic convergence of the Extrapolation Algorithm. Therefore the Appendixes A and B give sufficient conditions on nonautonomous and autonomous systems for $\underline{F}'$ and $\underline{G}'$, respectively, to be Lipschitz continuous.

Sections V and VI discuss problems related to the implementation of extrapolation methods, while Section VII gives examples and comparisons of the performance of extrapolation methods.

II. Extrapolation

Consider a sequence $y_0,y_1,\cdots$, generated by the recursion formula

$$y_r=\underline{A}y_{r-1}+b, \qquad r=1,2,\cdots. \quad (2.1)$$

When $\underline{B}=\underline{I}-\underline{A}$ is nonsingular a fixed point z_0:

$$z_0=\underline{A}z_0+b \quad (2.2)$$

can be computed from

$$\underline{B}z_0=b. \quad (2.3)$$

In this case the solution of the difference equation (2.1) can be given in closed form [10] as

$$y_r=\underline{A}^r(y_0-\underline{B}^{-1}b)+\underline{B}^{-1}b$$
$$=\underline{A}^r(y_0-z_0)+z_0. \quad (2.4)$$

Let y_0-z_0 be expanded after the principal vectors of $\underline{A}$, [11]:

$$y_0-z_0=\sum_{r=1}^{N}\alpha_r d_r. \quad (2.5)$$

The principal vectors $d_1,d_2,\cdots,d_N$ make up the columns of a matrix $\underline{D}$ so that $\underline{D}^{-1}\underline{A}\underline{D}$ is the Jordan canonical form which is a block diagonal matrix that consists of

blocks of the form [11]:

$\underline{C}_1(\lambda_s)=[\lambda_s]$ and

$$\underline{C}_p(\lambda_r)=\begin{bmatrix}\lambda_r & 1 & 0 & \cdots & 0 & 0\\ 0 & \lambda_r & 1 & \cdots & 0 & 0\\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots\\ 0 & 0 & 0 & \cdots & \lambda_r & 1\\ 0 & 0 & 0 & \cdots & 0 & \lambda_r\end{bmatrix}, \qquad p>1.$$

Each $p\times p$ block $\underline{C}_p(\lambda_r)$ is associated with one eigenvector and corresponds to an eigenvalue λ_r of multiplicity p. A principal vector $\boldsymbol{d}_s$ corresponding to a simple eigenvalue λ_s is an eigenvector of $\underline{A}$.

The minimum polynomial of the eigenvector $\boldsymbol{d}_s$ with respect to $\underline{A}$ is $x-\lambda_s$ as $(\underline{A}-\lambda_s\underline{I})\boldsymbol{d}_s=0$. The minimum polynomials of the principal vectors $\boldsymbol{d}_{r+1}, \boldsymbol{d}_{r+2},\cdots,\boldsymbol{d}_{r+p}$ corresponding to $\underline{C}_p(\lambda_r)$ are $(x-\lambda_r),(x-\lambda_r)^2,\cdots,(x-\lambda_r)^p$, respectively, and $\boldsymbol{d}_{r+1}$ is accordingly the eigenvector associated with λ_r (see [11, p. 43]). A principal vector $\boldsymbol{d}_{r+s}$ with minimum polynomial $(x-\lambda_r)^s$ is called a principal vector of grade s. Notice that the Jordan matrix can have several simple Jordan submatrices $\underline{C}_{p_1}(\lambda_r),\underline{C}_{p_2}(\lambda_r),\cdots$ for the same eigenvalue λ_r of multiplicity $p_1+p_2+\cdots$ but associated with linearly independent eigenvectors.

The minimum polynomial of y_0-z_0 in (2.5) with respect to $\underline{A}$ can now be written as

$$P_0(x)=\prod_{r\in I_0}(x-\lambda_r)^{p(r)}=\sum_{r=0}^{m_0}c_r x^r. \tag{2.6}$$

I_0 is a set of indexes specifying the different eigenvalues associated with the principal vectors of the expansion (2.5) (nonzero α_r-coefficients). The index $p(r)$ is the maximum degree of a minimum polynomial in λ_r of the principal vectors of the expansion (2.5). The degree of $P_0(x)$ is $m_0=\sum_{r\in I_0}p(r)$.

If $\underline{A}$ is singular it has a possibly multiple eigenvalue at zero. Define the index set $I_1=I_0\backslash\{r_0\}$ where $\lambda_{r_0}=0$, $q=p(r_0)$, and $m_1=m_0-q$. Assume that the multiplicity of λ_{r_0} is p. Then the exponent q of the corresponding factor of the minimum polynomial of y_0-z_0 may assume any value between 0 and $p, 0\leqslant q\leqslant p$, depending on the structure of the Jordan matrix and the expansion of y_0-z_0 in (2.5).

The minimum polynomial of $\underline{A}^s(y_0-z_0), s\geqslant q$ with respect to $\underline{A}$ is then

$$P_1(x)=\prod_{r\in I_1}(x-\lambda_r)^{p(r)}=\sum_{r=0}^{m_1}c_r x^r. \tag{2.7}$$

We can now prove the following.

Theorem 2.1

Let $y_r, r=0,1,\cdots$ be generated by (2.1). Then y_r fulfill the difference equation

$$\sum_{r=0}^{m_0}c_r y_{n+r}=z_0\sum_{r=0}^{m_0}c_r, \qquad n=0,1,\cdots \tag{2.8}$$

and

$$\sum_{r=0}^{m_1}c_r y_{n+r}=z_0\sum_{r=0}^{m_1}c_r, \qquad n=q,q+1,\cdots \tag{2.9}$$

$$\begin{array}{ccccccc} & & \epsilon_0^{(0)} & & & & \\ & \epsilon_{-1}^{(1)} & & \epsilon_1^{(0)} & & & \\ & & \epsilon_0^{(1)} & & \epsilon_2^{(0)} & & \\ & \epsilon_{-1}^{(2)} & & \epsilon_1^{(1)} & & \epsilon_3^{(0)} & \\ & & \epsilon_0^{(2)} & & \epsilon_2^{(1)} & & \\ & \epsilon_{-1}^{(3)} & & \epsilon_1^{(2)} & & & \\ & & \epsilon_0^{(3)} & & & & \end{array}$$

Fig. 1.

where m_0, m_1, and the coefficients $c_0, c_1,\cdots,c_{m_0}$ are defined above.

Proof: From (2.4) we find

$$\sum_{r=0}^{m_0}c_r y_{n+r}=\sum_{r=0}^{m_0}c_r\underline{A}^{n+r}(y_0-z_0)+\sum_{r=0}^{m_0}c_r z_0, \qquad n=0,1,\cdots.$$

Since $P_0(x)$ (2.6) is the minimum polynomial of y_0-z_0 with respect to $\underline{A}$, (2.8) follows immediately.

Analogously we find

$$\sum_{r=0}^{m_1}c_r y_{n+r}=\sum_{r=0}^{m_1}c_r\underline{A}^{n+r-q}\left[\underline{A}^q(y_0-z_0)\right]+\sum_{r=0}^{m_1}c_r z_0, \qquad n=q,q+1,\cdots$$

and (2.9) follows immediately from (2.7). ■

See also [12] which discusses the case where q is equal to the multiplicity of the zero eigenvalue.

ϵ-Algorithms

The ϵ-algorithm [13], [14] applied to the sequence $y_0, y_1,\cdots$ is defined as follows:

$$\epsilon_{-1}^{(r)}=0, \qquad r=1,2,\cdots$$

$$\epsilon_0^{(r)}=y_r, \qquad r=0,1,\cdots$$

$$\epsilon_{s+1}^{(r)}=\epsilon_{s-1}^{(r+1)}+\left(\epsilon_s^{(r+1)}-\epsilon_s^{(r)}\right)^{-1}, \qquad r=0,1,\cdots, \text{ and } s=0,1,\cdots. \tag{2.10}$$

The ϵ-values can be placed in a two-dimensional array, where formula (2.10) relates the elements within a lozenge. (See Fig. 1.)

There are different possibilities for defining the inverse of a vector in (2.10). The primitive inverse of a vector $\boldsymbol{v}$ is taken component-wise:

$$\boldsymbol{v}^{-1}=\left(v_1^{-1},v_2^{-1},\cdots,v_N^{-1}\right). \tag{2.11}$$

For this inverse the ϵ-algorithm is called scalar.

The Samelson inverse [14] is defined by

$$\boldsymbol{v}^{-1}=\boldsymbol{v}/\boldsymbol{v}^t\boldsymbol{v}=\boldsymbol{v}/\|\boldsymbol{v}\|_2^2. \tag{2.12}$$

where $\boldsymbol{v}^t$ means the transpose of $\boldsymbol{v}$. For this inverse the algorithm (2.10) is called the vector ϵ-algorithm. In the following sections the 2-norm will be used in connection with the vector ϵ-algorithm.

In [15] the following theorem is proved.

Theorem 2.2

If the ϵ-algorithm (2.10) is applied to the scalar sequence $y_0, y_1, \cdots$ fulfilling the difference equation:

$$\sum_{r=0}^{m} c_r y_{n+r} = z_0 \sum_{r=0}^{m} c_r, \qquad n=0,1,\cdots \tag{2.13}$$

then

$$\epsilon_{2m}^{(r)} = z_0, \qquad r=0,1,\cdots, \qquad \text{for } \sum_{r=0}^{m} c_r \neq 0$$

and

$$\epsilon_{2m}^{(r)} = 0, \qquad r=0,1,\cdots, \qquad \text{for } \sum_{r=0}^{m} c_r = 0. \quad \blacksquare$$

For the vector ϵ-algorithm a similar theorem is proved in [16].

Theorem 2.3

If the ϵ-algorithm (2.10) with the vector inverse (2.12) is applied to the vector sequence $y_0, y_1, \cdots$ fulfilling the difference equation

$$\sum_{r=0}^{m} c_r y_{n+r} = z_0 \sum_{r=0}^{m} c_r, \qquad n=0,1,\cdots \tag{2.14}$$

then

$$\epsilon_{2m}^{(r)} = z_0, \qquad r=0,1,\cdots, \qquad \text{for } \sum_{r=0}^{m} c_r \neq 0$$

and

$$\epsilon_{2m}^{(r)} = 0, \qquad r=0,1,\cdots, \qquad \text{for } \sum_{r=0}^{m} c_r = 0. \quad \blacksquare$$

The Theorems 2.1, 2.2, and 2.3 lead to the following.

Corollary 2.1

Let the sequence $y_0, y_1 \cdots$ be generated by (2.1) where $\underline{I} - \underline{A}$ is nonsingular. If the ϵ-algorithm (scalar or vector) is applied to $y_0, y_1, \cdots$, then $\epsilon_{2m_0}^{(r)} = z_0, r=0,1,\cdots$. If the ϵ-algorithm (scalar or vector) is applied to $y_q, y_{q+1}, \cdots$, then $\epsilon_{2m_1}^{(r)} = z_0, r=0,1,\cdots$.

Proof: Since the sequence $y_0, y_1, \cdots$ is generated by (2.1) it fulfils (2.8), and $y_q, y_{q+1}, \cdots$ fulfils (2.9) according to Theorem 2.1. The matrix $\underline{I} - \underline{A}$ is assumed nonsingular which implies that all of the eigenvalues of $\underline{A}$ are different from 1. From (2.6) and (2.7) we then have $\sum_{r=0}^{m_0} c_r \neq 0$ and $\sum_{r=0}^{m_1} c_r \neq 0$. As each component of the vector sequences fulfils a relation of the type (2.13), Theorem 2.2 gives the wanted result for the scalar ϵ-algorithm. Theorem 2.3 immediately gives the result for the vector ϵ-algorithm.

Minimum polynomial extrapolation

The difference equation

$$\sum_{r=0}^{m} c_r y_{n+r} = z_0 \sum_{r=0}^{m} c_r, \qquad n=0,1,\cdots$$

can be transformed as follows for $n = q, q+1, \cdots$:

$$\sum_{r=0}^{m} c_r (y_{n+r} - y_q) = (z_0 - y_q) \sum_{r=0}^{m} c_r$$

$$\Leftrightarrow \sum_{s=q}^{n-1} \left(\sum_{r=0}^{m} c_r \right) \Delta y_s + \sum_{s=n}^{m+n-1} \left(\sum_{r=s-n+1}^{m} c_r \right) \Delta y_s = \Delta z_0 \sum_{r=0}^{m} c_r \tag{2.15}$$

where $\Delta y_r = y_{r+1} - y_r$ and $\Delta z_0 = z_0 - y_q$. For $n = q$ formula (2.15) reduces to

$$\sum_{s=0}^{m-1} \left(\sum_{r=s+1}^{m} c_r \right) \Delta y_{s+q} = \Delta z_0 \sum_{r=0}^{m} c_r \tag{2.16}$$

which is the extrapolation formula described in [17] and [18]. For this formula we can prove the following.

Theorem 2.4

Let the sequence $y_0, y_1, \cdots$ be generated by (2.1) where $\underline{I} - \underline{A}$ is nonsingular. Then z_0 can be found from

$$\sum_{s=0}^{m-1} \left(\sum_{r=s+1}^{m} c_r \right) \Delta y_{s+q} = \Delta z_0 \sum_{r=0}^{m} c_r \tag{2.17}$$

as $z_0 = \Delta z_0 + y_q$ where $m = m_0$ for $q=0$ and $m = m_1$ for q sufficiently large.

The coefficients $c_0, c_1, \cdots, c_m$ can be computed as the coefficients of the minimum polynomial of $y_{q+1} - y_q$ with respect to $\underline{A}$.

Proof: The first part of the theorem follows directly from the derivation of (2.16).

From (2.4) it follows that

$$y_{q+1} - y_q = (\underline{A} - \underline{I}) \underline{A}^q (y_0 - z_0). \tag{2.18}$$

Assume that $P(x) = \sum_{r=0}^{m} c_r x^r$ is the minimum polynomial of $y_{q+1} - y_q$ with respect to $\underline{A}$.

$$\sum_{r=0}^{m} c_r \underline{A}^r (\underline{A} - \underline{I}) \underline{A}^q (y_0 - z_0) = 0$$

$$\Leftrightarrow \sum_{r=0}^{m} c_r \underline{A}^r [\underline{A}^q (y_0 - z_0)] = 0.$$

This implies that $P(x)$ is an anihilating polynomial of $\underline{A}^q (y_0 - z_0)$ with respect to $\underline{A}$. From (2.5) and (2.18) it is seen that $y_{q+1} - y_q$ can be expanded after the same principal vectors as $\underline{A}^q (y_0 - z_0)$ plus possibly some additional vectors of lower grade for multiple eigenvalues. Therefore, $P(x)$ must be the minimum polynomial.

The zeros of $P(x)$ are eigenvalues of $\underline{A}$. Since $\underline{A} - \underline{I}$ is nonsingular, $P(1) = \sum_{r=0}^{m} c_r \neq 0$. This completes the proof of Theorem 2.4. $\blacksquare$

From (2.1) we have that $y_{q+2} - y_{q+1} = \underline{A}(y_{q+1} - y_q)$, etc. The minimum polynomial of $\Delta y_q = y_{q+1} - y_q$ with respect to $\underline{A}$ can, therefore, be computed from the sequence $\Delta y_q, \Delta y_{q+1} \cdots$ which is easily derived from the sequence $y_q, y_{q+1}, \cdots$ generated by (2.1). The computational details are given in Section VI.

III. Error Analysis

ϵ-Algorithms

Let each entry of the ϵ-table of Fig. 1 have the error $\boldsymbol{a}_s^{(r)}$. The values $\boldsymbol{\epsilon}_s^{(r)}+\boldsymbol{a}_s^{(r)}$ are the numbers, including errors, encountered in practical computations, and $\boldsymbol{\epsilon}_s^{(r)}$ are the exact values.

Theorem 3.1

For sufficiently small initial errors $\boldsymbol{a}_0^{(r)}, r=j, j+1, \cdots, j+k$, the error $\boldsymbol{a}_k^{(j)}$ of the computed value $(\boldsymbol{\epsilon}_k^{(j)}+\boldsymbol{a}_k^{(j)})$ of the ϵ-algorithm depends on the initial errors as

$$\boldsymbol{a}_k^{(j)}=0(\|\boldsymbol{a}_0\|) \tag{3.1}$$

where $\|\boldsymbol{a}_0\|=\max \|\boldsymbol{a}_0^{(r)}\|, r=j, j+1, \cdots, j+k$. The result is valid for the scalar ϵ-algorithm applied componentwise and for the vector ϵ-algorithm. In both cases it is assumed that $(\boldsymbol{\epsilon}_s^{(r+1)}-\boldsymbol{\epsilon}_s^{(r)})^{-1}$ exists for all steps of the algorithm leading to $\boldsymbol{\epsilon}_k^{(j)}$.

Proof: The scalar ϵ-algorithm is applied componentwise, and for a single component at a typical stage we have

$$\boldsymbol{\epsilon}_{s+1}^{(r)}+\boldsymbol{a}_{s+1}^{(r)}=\boldsymbol{\epsilon}_{s-1}^{(r+1)}+\boldsymbol{a}_{s-1}^{(r+1)}+\left(\boldsymbol{\epsilon}_s^{(r+1)}+\boldsymbol{a}_s^{(r+1)}-\left(\boldsymbol{\epsilon}_s^{(r)}+\boldsymbol{a}_s^{(r)}\right)\right)^{-1}. \tag{3.2}$$

For $\Delta\boldsymbol{\epsilon}_s^{(r)}=\boldsymbol{\epsilon}_s^{(r+1)}-\boldsymbol{\epsilon}_s^{(r)}$ and $\Delta\boldsymbol{a}_s^{(r)}=\boldsymbol{a}_s^{(r+1)}-\boldsymbol{a}_s^{(r)}$ we find for $|\Delta\boldsymbol{a}_s^{(r)}|<|\Delta\boldsymbol{\epsilon}_s^{(r)}|$ that

$$\boldsymbol{a}_{s+1}^{(r)}=\boldsymbol{a}_{s-1}^{(r+1)}-\Delta\boldsymbol{a}_s^{(r)}/(\Delta\boldsymbol{\epsilon}_s^{(r)})^2+0\left((\Delta\boldsymbol{a}_s^{(r)})^2\right). \tag{3.3}$$

When all the steps of the ϵ-algorithm leading to $\boldsymbol{\epsilon}_k^{(j)}$ exist, i.e., all $\Delta\boldsymbol{\epsilon}_s^{(r)}\neq 0$, and $|\Delta\boldsymbol{a}_s^{(r)}|<|\Delta\boldsymbol{\epsilon}_s^{(r)}|$, then (3.3) implies that

$$\boldsymbol{a}_k^{(j)}=0(a_0) \tag{3.4}$$

where $a_0=\max\ |a_0^{(r)}|, r=j, j+1, \cdots, j+k$. Compare this analysis with [19] and [10].

For the vector ϵ-algorithm an expression similar to (3.3) can be developed. From

$$\boldsymbol{\epsilon}_{s+1}^{(r)}+\boldsymbol{a}_{s+1}^{(r)}=\boldsymbol{\epsilon}_{s-1}^{(r+1)}+\boldsymbol{a}_{s-1}^{(r+1)}+\frac{\Delta\boldsymbol{\epsilon}_s^{(r)}+\Delta\boldsymbol{a}_s^{(r)}}{\|\Delta\boldsymbol{\epsilon}_s^{(r)}\|^2}\frac{\|\Delta\boldsymbol{\epsilon}_s^{(r)}\|^2}{\|\Delta\boldsymbol{\epsilon}_s^{(r)}+\Delta\boldsymbol{a}_s^{(r)}\|^2}$$

the error is found to be

$$\boldsymbol{a}_{s+1}^{(r)}=\boldsymbol{a}_{s-1}^{(r+1)}+\left(\alpha\Delta\boldsymbol{\epsilon}_s^{(r)}+(1+\alpha)\Delta\boldsymbol{a}_s^{(r)}\right)/\|\Delta\boldsymbol{\epsilon}_s^{(r)}\|^2$$

where

$$1/(1+\alpha)=\|\Delta\boldsymbol{\epsilon}_s^{(r)}+\Delta\boldsymbol{a}_s^{(r)}\|^2/\|\Delta\boldsymbol{\epsilon}_s^{(r)}\|^2 =1+\left(\|\Delta\boldsymbol{a}_s^{(r)}\|^2+2\Delta\boldsymbol{\epsilon}_s^{(r)t}\Delta\boldsymbol{a}_s^{(r)}\right)/\|\Delta\boldsymbol{\epsilon}_s^{(r)}\|^2. \tag{3.5}$$

For

$$\|\Delta\boldsymbol{a}_s^{(r)}\|<\|\Delta\boldsymbol{\epsilon}_s^{(r)}\| \tag{3.6}$$

the expression (3.5) is different from zero and

$$\alpha=-2\Delta\boldsymbol{\epsilon}_s^{(r)t}\Delta\boldsymbol{a}_s^{(r)}/\|\Delta\boldsymbol{\epsilon}_s^{(r)}\|^2+0(\|\Delta\boldsymbol{a}_s^{(r)}\|^2) \tag{3.7}$$

the error can, therefore, be expressed as

$$\boldsymbol{a}_{s+1}^{(r)}=\boldsymbol{a}_{s-1}^{(r+1)}+\Delta\boldsymbol{a}_s^{(r)}/\|\Delta\boldsymbol{\epsilon}_s^{(r)}\|^2 -\Delta\boldsymbol{\epsilon}_s^{(r)}\left(2\Delta\boldsymbol{\epsilon}_s^{(r)t}\Delta\boldsymbol{a}_s^{(r)}/\|\Delta\boldsymbol{\epsilon}_s^{(r)}\|^4\right)+0(\|\Delta\boldsymbol{a}_s^{(r)}\|^2). \tag{3.8}$$

For the scalar case the error (3.8) degenerates into (3.3). When all the steps leading to $\boldsymbol{\epsilon}_k^{(j)}$ exist, i.e., all $\|\Delta\boldsymbol{\epsilon}_s^{(r)}\|\neq 0$, and $\|\Delta\boldsymbol{a}_s^{(r)}\|<\|\Delta\boldsymbol{\epsilon}_s^{(r)}\|$, then (3.8) implies that

$$\boldsymbol{a}_k^{(j)}=0(\|\boldsymbol{a}_0\|)$$

where $\|\boldsymbol{a}_0\|=\max\ \|\boldsymbol{a}_0^{(r)}\|, r=j, j+1, \cdots, j+k$. This result in connection with (3.4) completes the proof of the theorem.

The second term of (3.3) may damp or amplify the error depending on the actual values of $\Delta\boldsymbol{a}_s^{(r)}$ and $\Delta\boldsymbol{\epsilon}_s^{(r)}$. In [20] the error properties of the ϵ-algorithm are studied for the sequence $y_r, r=0, 1, \cdots$:

$$y_r\sim z_0+\sum_{s=1}^{\infty} d_s\lambda_s^r, \qquad 1>\lambda_1>\lambda_2>\cdots>0. \tag{3.9}$$

The asymptotically "equal to" sign $\sim$ means that y_r converges uniformly to the expression on the right-hand side for $r\to\infty$.

The following asymptotic expression for $a_{2s}^{(r)}$ is given when $a_{2s-2}^{(r)}$ is considered the only error ($a_{2s-1}^{(r)}$ originates from $a_{2s-2}^{(r)}$):

$$a_{2s}^{(r)}\sim-\left(\frac{\lambda_s}{1-\lambda_s}\right)^2 a_{2s-2}^{(r)} \tag{3.10}$$

for $r\to\infty$. If λ_s is close to 1, the expression predicts a severe error amplification which is also observed in practice. Formula (3.10) is also valid for the sequence (3.9) when $-1<\lambda_1<\lambda_2<\cdots<0$, and in this case the magnitudes of the errors are damped for increasing s.

Numerical experiments indicate that an expression similar to (3.10), namely,

$$a_{2s}^{(r)}\sim-\left|\frac{\lambda_s}{1-\lambda_s}\right|^2 a_{2s-2}^{(r)}$$

is valid for a real-valued series (3.9) with complex conjugate λ-pairs, $1>|\lambda_1|>|\lambda_3|>\cdots>0$. Even for $|\lambda_s|$ very close to 1 (or equal to 1) the errors do not amplify severely if $|1-\lambda_s|$ is well bounded away from zero.

Minimum polynomial extrapolation

Let the sequence $\boldsymbol{y}_0, \boldsymbol{y}_1, \cdots$ be generated by the formula (2.1), and let another sequence $\boldsymbol{y}_0^*, \boldsymbol{y}_1^*, \cdots$ be generated by

$$\boldsymbol{y}_r^*=\underline{\boldsymbol{A}}\boldsymbol{y}_{r-1}^*+\boldsymbol{b}+\boldsymbol{e}_r, \qquad r=1, 2, \cdots$$

where $\boldsymbol{e}_r$ is considered an error term.

Let $P(x)=\sum_{r=0}^m c_r x^r$ be the minimum polynomial of $\Delta\boldsymbol{y}_q$ with respect to $\underline{\boldsymbol{A}}$. Then

$$\underline{\boldsymbol{D}}\boldsymbol{c}=\Delta\boldsymbol{y}_q$$

where $c_0=-1, \boldsymbol{c}=(c_1, c_2, \cdots, c_m)^t$, and $\underline{\boldsymbol{D}}=(\Delta\boldsymbol{y}_{q+1}\ \ \Delta\boldsymbol{y}_{q+2}\ \cdots\ \Delta\boldsymbol{y}_{q+m})$. $\underline{\boldsymbol{D}}$ is an $N\times m$ matrix where $m\leqslant N$. For $\boldsymbol{e}_0=\boldsymbol{y}_0^*-\boldsymbol{y}_0$ and $\boldsymbol{a}_r=\boldsymbol{y}_r^*-\boldsymbol{y}_r$ we find

$$\Delta\boldsymbol{y}_r^*=\Delta\boldsymbol{y}_r+\Delta\boldsymbol{a}_r, \qquad r=0, 1, \cdots \tag{3.11}$$

where

$$\Delta\boldsymbol{a}_r=\underline{\boldsymbol{A}}^{r+1}\boldsymbol{e}_0+\sum_{s=0}^{r}\underline{\boldsymbol{A}}^{r-s}\Delta\boldsymbol{e}_s.$$

Notice that $\boldsymbol{a}_r$ is identical to the error $\boldsymbol{a}_r^{(0)}$ defined in the previous section.

It is assumed that $a_0 \to 0, a_1 \to 0, \cdots \Leftrightarrow e_0 \to 0, e_1 \to 0, \cdots$ and that the order of convergence is the same for the two sequences.

Define the error matrix $\underline{E}$ by

$$\underline{E} = (\Delta a_{q+1} \quad \Delta a_{q+2} \quad \cdots \quad \Delta a_{q+m})$$

and $\underline{D}^* = \underline{D} + \underline{E}$. The coefficients $c_1^*, c_2^*, \cdots, c_m^*$ of an almost anihilating polynomial of Δy_q^* with respect to $\underline{A}$ can be found by least-squares technique such that

$$\rho^* = \min_{c^*} \|\underline{D}^* c^* - \Delta y_q^*\|_2. \tag{3.12}$$

Theorem 3.2

The extrapolated value z_0^* found from $z_0^* = \Delta z_0^* + y_q^*$ where

$$\sum_{s=0}^{m-1} \left(\sum_{r=s+1}^{m} c_r^* \right) \Delta y_{s+q}^* = \Delta z_0^* \sum_{r=0}^{m} c_r^* \tag{3.13}$$

depends on the initial errors $a_r, r = q, q+1, \cdots, q+m+1$:

$$z_0^* - z_0 = 0(\|a\|) \tag{3.14}$$

where $\|a\| = \max \|a_r\|, r = q, q+1, \cdots, q+m+1$.

Proof: The condition number κ of the rectangular matrix $\underline{D}$ is defined by $\kappa = \|\underline{D}\| \|\underline{D}^+\|$ where $\underline{D}^+$ denotes the pseudo inverse of $\underline{D}$. For $\lambda_1, \lambda_2, \cdots, \lambda_m$ being the eigenvalues of $\underline{D}'\underline{D}$, the 2-norm of $\underline{D}$ is defined as

$$\|\underline{D}\|_2 = \sqrt{\max_r \lambda_r}\ .$$

The 2-norm will be used in the rest of this section.

In [21] the following results on perturbation of a rectangular system is given.

For $\kappa \|\underline{E}\| / \|\underline{D}\| < 1$ the rank of $\underline{D}^* = \underline{D} + \underline{E}$ is greater than or equal to the rank of $\underline{D}$. As $\underline{D}$ per definition has full rank m, so has $\underline{D}^*$ for $\kappa \|\underline{E}\| / \|\underline{D}\| < 1$, and

$$\|c - c^*\| \leq \|\underline{D}^+\| (\|\Delta a_q\| + \|\underline{E}\| \|c\|) / (1 - \kappa \|\underline{E}\| / \|\underline{D}\|). \tag{3.15}$$

In [18] a relation is derived for the difference $\|\Delta z_0 - \Delta z_0^*\|$ where Δz_0 is computed from (2.17) and Δz_0^* is computed from (3.13).

The relation is

$$\|P^*(1)(\Delta z_0 - \Delta z_0^*)\| \leq (m+2) \sum_{r=0}^{m} |c_r^*| \, \|(\underline{I} - \underline{A})^{-1}\| \max_s (\|\Delta e_s\|, \rho^*) \tag{3.16}$$

where $P^*(x) = \sum_{r=0}^{m} c_r^* x^r$.

The expression $P^*(1)$ can be evaluated as follows:

$$P^*(1) = P(1) + (P^*(1) - P(1)) = P(1) + \sum_{r=0}^{m} (c_r^* - c_r)$$

$$|P^*(1)| \geq \left| |P(1)| - \left| \sum_{r=0}^{m} (c_r - c_r^*) \right| \right|$$

$$\left| \sum_{r=0}^{m} (c_r - c_r^*) \right| \leq \|c - c^*\|_1 \leq \sqrt{m} \, \|c - c^*\|_2.$$

For $\|\Delta a_q\|$ and $\|\underline{E}\|$ being sufficiently small, (3.15) gives that $|P(1)| - \sqrt{m} \, \|c - c^*\| \geq 0$, and

$$\|\Delta z_0 - \Delta z_0^*\| \leq (m+2) \sqrt{m} \, (\|c\| + \|c - c^*\|) \|(\underline{I} - \underline{A})^{-1}\| \cdot \max_s (\|\Delta e_s\|, \rho^*) / (|P(1)| - \sqrt{m} \, \|c - c^*\|). \tag{3.17}$$

For $\underline{D}c = \Delta y_q$ we have

$$\rho^* = \|\underline{D}(c^* - c) + \underline{E}c^* - \Delta a_q\| = 0(\|a\|) \tag{3.18}$$

for $\|a\| = \max \|a_r\|, r = q, q+1, \cdots, q+m+1$ where $y_r^* = y_r + a_r$.

For $\|a\|$ sufficiently small we have $\kappa \|\underline{E}\| / \|\underline{D}\| < 1$ and $P(1) - \sqrt{m} \, \|c - c^*\| > 0$ such that (3.17) can be estimated by

$$\|\Delta z_0 - \Delta z_0^*\| = 0(\|a\|).$$

As

$$\Delta z_0 - \Delta z_0^* = z_0 - z_0^* - y_q + y_q^*$$

we have

$$\|z_0 - z_0^*\| = 0(\|a\|)$$

which completes the proof of the theorem. ■

If the c coefficients are not the coefficients of an anihilating polynomial but determined by a least-squares technique such that

$$\rho = \min_c \|\underline{D}c - \Delta y_q\|_2$$

then (3.15) changes into [21]:

$$\|c - c^*\| \leq \|\underline{D}^+\| \big[\|\Delta a_q\| + \|\underline{E}\| \cdot (\|c\| + \rho \|\underline{D}^+\| / (1 - \kappa \|\underline{E}\| / \|\underline{D}\|)) \big] / (1 - \kappa \|\underline{E}\| / \|\underline{D}\|)$$

which means that $\|c - c^*\|$ is still 0 ($\|a\|$) whereas (3.18) changes into

$$\rho^* = \rho + 0(\|a\|).$$

This means that $\|z_0 - z_0^*\| = 0(\|a\|)$ only if ρ^* is dominated by the term $0(\|a\|)$.

IV. Convergence of the Extrapolation Algorithm

Define the following iteration algorithm for the computation of a fixed point of the nonlinear function $F: R^N \to R^N$.

Extrapolation Algorithm

a) Let $y_0 = x^{(n)}$.

b) Generate a sequence $y_1, y_2, \cdots$ by $y_{r+1} = F(y_r)$.

c) Apply an ϵ-algorithm (scalar or vector) to $y_q, y_{q+1}, \cdots, y_{q+2m}$ or the minimum polynomial extrapolation to $y_q, y_{q+1}, \cdots, y_{q+m+1}$.

d) Let $x^{(n+1)} = \epsilon_{2m}^{(0)}$ for the ϵ-algorithms and $x^{(n+1)} = \Delta z_0 + y_q$ for minimum polynomial extrapolation.

We will now prove the basic convergence theorem.

Theorem 4.1

Assume that F has the fixed point z_0 such that $F(z_0) = z_0$. Assume further that F is differentiable with a Lipschitz continuous derivative in $D \subset R^N$ where $z_0 \in D$ and that $\underline{F}'(z_0) - \underline{I}$ is nonsingular.

When m in each step is chosen as the degree of the minimum polynomial of $y_q - z_0$ with respect to $\underline{F}'(z_0)$, then the Extrapolation Algorithm will converge quadratically to z_0 for $x^{(0)} \in D$ sufficiently close to z_0:

$$\|x^{(n+1)} - z_0\| = 0(\|x^{(n)} - z_0\|^2).$$

For the ϵ-algorithms it is further assumed that all of the steps leading to $\epsilon_{2m}^{(0)}$ exist.

Proof: From the Lipschitz continuity of $\underline{F}'$ we have

$$y_r - z_0 = \underline{F}'(z_0)(y_{r-1} - z_0) + 0(\|y_{r-1} - z_0\|^2)$$

and

$$y_r - z_0 = (\underline{F}'(z_0))^r (y_0 - z_0) + 0(\|y_0 - z_0\|^2). \quad (4.1)$$

Let $P(x) = \sum_{r=0}^{m} c_r x^r$ be the minimum polynomial of $y_q - z_0$ with respect to $\underline{F}'(z_0)$. The sequence $y_1, y_2, \cdots$ generated by (4.1) can be considered of the linear type (2.1) but with errors $0(\|y_0 - z_0\|^2)$. As $\underline{F}'(z_0) - \underline{I}$ is nonsingular, $P(1) \neq 0$, and the proof follows immediately from Corollary 2.1 and Theorems 2.4, 3.1, and 3.2. ■

The Extrapolation Algorithm has two parameters q and m. In practice there is often a third parameter δ, a measure of the error of the numerical computation of $\boldsymbol{F}$. It will now be discussed how the choice of q, m, and δ affects the convergence properties of the Extrapolation Algorithm.

Let the eigenvalues of $\underline{F}'(z_0)$ be $|\lambda_1| \geq |\lambda_2| \geq \cdots \geq |\lambda_N|$, and let the corresponding principal vectors be $\boldsymbol{d}_1, \boldsymbol{d}_2, \cdots, \boldsymbol{d}_N$. Expand $y_0 - z_0$ after the principal vectors

$$y_0 - z_0 = \sum_{r=1}^{N} \alpha_r \boldsymbol{d}_r$$

and by (4.1):

$$y_n - z_0 = \sum_{r=1}^{N} \alpha_r \underline{F}'(z_0)^n \boldsymbol{d}_r + 0(\|y_0 - z_0\|^2)$$

if $|\lambda_m| \gg |\lambda_{m+1}|$, we can write

$$y_n - z_0 = \sum_{r=1}^{m} \alpha_r \underline{F}'(z_0)^n \boldsymbol{d}_r + \gamma_n + 0(\|y_0 - z_0\|^2)$$

where

$$\gamma_n = \sum_{r=m+1}^{N} \alpha_r \underline{F}'(z_0)^n \boldsymbol{d}_r \quad (4.2)$$

and

$$\|\gamma_n\| / \|y_n - z_0\| \ll 1, \qquad \text{for } n \geq q.$$

If $\boldsymbol{F}$ is only computed to a limited precision so that y_n^* is the numerical approximation of y_n and $y_n^* = y_n + \delta_n$, then the extrapolation is applied to a sequence $y_q^*, y_{q+1}^*, \cdots$ generated by

$$y_n^* - z_0 = \sum_{r=1}^{m} \alpha_r \underline{F}'(z_0)^n \boldsymbol{d}_r + \gamma_n + \delta_n + 0(\|y^*{}_0 - z_0\|^2). \quad (4.3)$$

When the sequence subject to extrapolation is generated by (4.3) rather than (4.1), we find

$$\|x^{(n+1)} - z_0\| = 0(\|\gamma\|) + 0(\|\delta\|) + 0(\|x^{(n)} - z_0\|^2) \quad (4.4)$$

where

$$\|\gamma\| = \max \|\gamma_r\|, \qquad r = q, q+1, \cdots, s$$
$$\|\delta\| = \max \|\delta_r\|, \qquad r = q, q+1, \cdots, s$$

with $s = q + 2m$ for the ϵ-algorithms and $s = q + m + 1$ for minimum polynomial extrapolation.

According to (4.4) the Extrapolation Algorithm will be quadratically convergent as long as the deviation from linearity $0(\|x^{(n)} - z_0\|^2)$ is dominant. When this is no longer the case, the convergence stops.

The magnitude of $\|\gamma\|$ can be decreased by increasing q. In fact, the value of q determines how much m is smaller than N, so by increasing q it may be possible to decrease m.

The numerical error in the evaluation of $\boldsymbol{F}$ is described by $\boldsymbol{\delta}$. When $\boldsymbol{F}$ is computed by integrating ordinary differential equations, $\|\boldsymbol{\delta}\|$ can be decreased by decreasing the step-length in the numerical integration.

Through the definition (A7) of $\boldsymbol{F}$ the state-equation formulation of an electrical network is assumed. For different formulations, e.g., the modified nodal equations [22], the results based on state-equations will still apply under fairly mild restrictions as the following theorem shows.

Theorem 4.2

Let $\boldsymbol{H}: R^N \to R^M$ have a Lipschitz continuous derivative $\underline{\boldsymbol{H}}'$ and a differentiable inverse $\boldsymbol{H}^{-1}$, and let the sequence $w_0, w_1, \cdots$ be derived from $y_0, y_1, \cdots, y_{r+1} = \boldsymbol{F}(y_r)$, by $w_r = \boldsymbol{H}(y_r), r = 0, 1, \cdots$. Under the assumptions of Theorem 4.1 (same value of m) the Extrapolation Algorithm applied to the transformed sequences (Step c) $w_q, w_{q+1}, \cdots$ will produce a quadratically convergent sequence towards $w = \boldsymbol{H}(z_0)$.

Proof: From the Lipschitz continuity of $\underline{\boldsymbol{H}}'$ follows

$$w_n - \boldsymbol{H}(z_0) = \underline{\boldsymbol{H}}'(z_0)(y_n - z_0) + 0(\|y_n - z_0\|^2). \quad (4.5)$$

For the polynomial $P(x)$ in the proof of Theorem 4.1:

$$\sum_{r=0}^{m} c_r w_{n+r} = \sum_{r=0}^{m} c_r \{\boldsymbol{H}(z_0) + \underline{\boldsymbol{H}}'(z_0)(y_{n+r} - z_0) + 0(\|y_{n+r} - z_0\|^2)\}.$$

By using (4.1) and the fact that $P(x)$ is the minimum polynomial of $y_q - z_0$ with respect to $\underline{F}'(z_0)$, we find the following:

$$\sum_{r=0}^{m} c_r w_{n+r} = \sum_{r=0}^{m} c_r \{\boldsymbol{H}(z_0) + \underline{\boldsymbol{H}}'(z_0)(\underline{F}'(z_0))^{n+r}(y_0 - z_0) + 0(\|y_0 - x_0\|^2)\}$$
$$= \boldsymbol{H}(z_0) \sum_{r=0}^{m} c_r + 0(\|y_0 - z_0\|^2), \qquad \text{for } n = q, q+1, \cdots.$$

Since $(\underline{\boldsymbol{H}}^{-1}(w))' \underline{\boldsymbol{H}}'(z_0) = \underline{I}$ the nonzero eigenvalues of $\underline{\boldsymbol{H}}'(z_0)\underline{F}'(z_0)(\underline{\boldsymbol{H}}^{-1}(w))'$ are identical to the nonzero eigen-

values of $\underline{F}'(z_0)$ which are assumed to be different from unity.

From (4.5) it follows that $0(\|y_0 - z_0\|^2) = 0(\|w_0 - H(z_0)\|^2)$, and the rest of the proof follows from Theorems 2.2, 2.3, 2.4, 3.1, and 3.2. ■

From Theorem 4.2 it follows that the efficiency of extrapolation is independent of the formulation of the network equations (under the conditions of Theorem 4.2). There is, therefore, no particular reason for choosing state-equations, and no set of state-variables has to be identified.

V. Control of the Extrapolation Algorithm

As mentioned in the previous section the parameters q and m must be chosen properly for the Extrapolation Algorithm. The choice of q is mainly based on *a priori* knowledge of the transient part of the solution. By inspection of the circuit or by analysis of the circuit linearized at various points, it may be revealed that some of the transients of the circuit decay very rapidly compared to the period of the steady-state solution. For $q=1$ or 2 these rapidly decaying transients have often disappeared, and m is equal to the number of remaining, slowly decaying transients.

A very strong indication of the correct choice of q and m is quadratic convergence behavior (4.4). If either m or q is too small, $\|\gamma\|$ is too large for the convergence to proceed properly, and it must be investigated which parameter should be increased. Experiment is often the only way to make this decision. If a good estimate of the errors $a_0, a_1, \cdots$ of the data $y_0, y_1, \cdots$ is available, the formulas (3.3) or (3.8) can be used to follow the error propagation through the ϵ-algorithms. The ϵ-algorithms, without errors, terminate when $\epsilon_{2m}^{(0)} = \epsilon_{2m}^{(1)} = \cdots$. In the presence of errors the termination criterion can be taken to be $\|\Delta\epsilon_{2m}^{(s)}\| < \|\Delta a_{2m}^{(s)}\|$ for some s (see (3.6)).

This termination criterion is compatible with the precision of the data, but the precision may not be sufficient to assure quadratic convergence.

When the minimum polynomial method is used, an extra tool is available for deciding on the value of m. For m increasing from 1, the value of ρ^* (3.12) shows a very pronounced drop in value (several orders of magnitude) when the correct value of m is reached. Unfortunately $x^{(n)}$ must be reasonably close to z_0 before this approach is reliable, but it often works already for the second iteration. For the first iteration a safe value of m should then be used (e.g., $m=N$).

If some of the transients are extremely slowly decaying, lack of precision in the numerical integration can prevent convergence (4.4). It is obvious that the change of a transient from the sample point nT to $(n+1)T$ must be well above the level of the integration errors in order to make extrapolation work properly (cf. (3.10)).

In solving differential equations numerically we substitute a discretized system for the continuous. If the integration is performed with a fixed step-length and order pattern, the error δ_r of y_r will converge towards a limit δ as the number of iterations n tends to infinity.

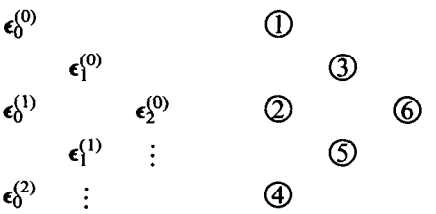

Fig. 2. Organization of ϵ-extrapolation.

From the error analysis of the ϵ-algorithms and the minimum polynomial extrapolation it is seen that a constant error δ of the initial data results in exactly the same error of the extrapolated results. This means that the error is neither amplified nor damped. This is in contrast to an error from a method which decides upon order and step-length from estimated values of the local truncation error.

But even for a discretization method with fixed step-length and order pattern the agreement between the continuous system and the discretized system must be close enough to represent the slowest changing transient in the continuous solution.

VI. Implementation

ϵ-Algorithms

The ϵ-algorithms are very easy to implement. The Extrapolation Algorithm can be implemented as a post processing of the results from a transient analysis program. In order to use storage as efficiently as possible the scheme in Fig. 1 should be computed in the sequence given in Fig. 2.

The data structure should be chosen such that a storage location (vector) can be released when the result it holds is no longer needed. At the computational stage of Fig. 2 only $\epsilon_0^{(2)}$, $\epsilon_1^{(1)}$, and $\epsilon_2^{(0)}$ are needed for further computation. For the computation in Fig. 2 only 4 vectors are needed, and it is easily shown that extrapolation from $\epsilon_0^{(0)}$, $\epsilon_0^{(1)}, \cdots, \epsilon_0^{(2m)}$ to $\epsilon_{2m}^{(0)}$ can be done using only $2m+2$ storage vectors (plus 1 auxiliary vector for $\Delta\epsilon_s^{(r)}$ for the vector ϵ-algorithm). Apart from the storage vectors some integer pointers are needed to manage the data structure.

For numerical reasons the data $y_q, y_{q+1}, \cdots$ should be scaled before application of the vector ϵ-algorithm. An obvious choice is scaling by the diagonal matrix $\underline{S}$ where $\underline{S} = \text{diag}\,(y_{1q}^{-1}, y_{2q}^{-1}, \cdots, y_{Nq}^{-1})$. If the sequence $y_q, y_{q+1}, \cdots$ fulfills the difference equation (2.14), then the scaled sequence $\hat{y}_q, \hat{y}_{q+1}, \cdots$ fulfills

$$\sum_{r=0}^{m} c_r \hat{y}_{n+r} = \hat{z}_0 \sum_{r=0}^{m} c_r, \qquad n = q, q+1, \cdots$$

where $\hat{z}_0 = \underline{S} z_0$.

Minimum polynomial extrapolation

Once the coefficients $c_0, c_1, \cdots, c_m$ have been computed, the minimum polynomial extrapolation formula (2.17) is straightforward to use. Therefore, the following section will concentrate on the computation of the coefficients of the minimum polynomial.

In Section III it is suggested to compute $c = (c_1, c_2, \cdots, c_m)^t$ as the least-squares solution of

$$\rho_m = \min_c \|\underline{D}c - \Delta y_q\|_2, \qquad c_0 = -1 \tag{6.1}$$

where $\underline{D}$ and Δy_q are defined in Section III. If m is known in advance, this technique is cheap and efficient.

If the values of $\rho_1, \rho_2, \cdots$ are wanted for $\underline{D}$ having $1, 2, \cdots$ columns, the least-squares approach is not very attractive. For each new ρ_s the computations have to start from the beginning, and if the copy of $\Delta y_q, \Delta y_{q+1}, \cdots$ (for (2.17)) is held on backing storage, it has to be retrieved for each new computation.

Instead of optimization the method of Krylov [11] can be used for computing the coefficients of the minimum polynomial (suggested in [18]). by the method of Krylov $c_1, c_2, \cdots, c_s$ are found from the $N \times s$ system

$$(\Delta y_{q+1} \quad \Delta y_{q+2} \quad \cdots \quad \Delta y_{q+s}) \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_s \end{bmatrix} = (\Delta y_q), \qquad c_0 = -1 \tag{6.2}$$

by applying LU-factorization column-wise and row permutation. It does not make sense to do permutation unless the system is equilibrated, but as the LU-factorization is performed one column at a time as the columns become available, the usual approach for finding the equilibration weights cannot be used. Under the assumption that the sequence $y_q, y_{q+1}, \cdots$ converges, the elements of Δy_q are taken as equilibration weights.

In the solution all of the right-hand side takes part in the forward substitution so that the right-hand side corresponds to the result of a Gaussian elimination. The U matrix is $s \times s$, and the backward substitution transforms the first s elements of the right-hand side into $c_1, c_2, \cdots, c_s$, while the last $N-s$ elements will be equal to the residual vector $r_s = \Delta y_q - \underline{D}_s c_s$ for $\underline{D}_s$ having s columns. By studying $\|r_s\|/\|c_s\|$ for increasing s, it is possible to decide on the value of m. In [18] a sudden drop in $\|r_s\|/\|c_s\|$ is taken as an indication of s having reached the value of m. The problem of finding m is basicly a numerical rank determination problem, which is considered very difficult. For the extrapolation it is, however, not disastrous if m is chosen too small for an iteration. The convergence will be slower than quadratic, and a larger value of m can be chosen for the next iteration.

The magnitude of $\|r_s\|$ is assumed to be of the same order as ρ_s, and this parameter is important in the error formulas (3.16) and (3.17).

After the LU-factorization the system (6.2) has been decomposed as follows:

$$\underline{L}\underline{U}c = \Delta y_q \tag{6.3}$$

where $\underline{L}$ is an $N \times s$ unit lower trapezoidal matrix, and $\underline{U}$ is an upper triangular matrix. Instead of solving the first s equations of (6.3), c can be found by least-squares techniques:

$$\underline{L}^t\underline{L}\underline{U}c = \underline{L}^t \Delta y_q. \tag{6.4}$$

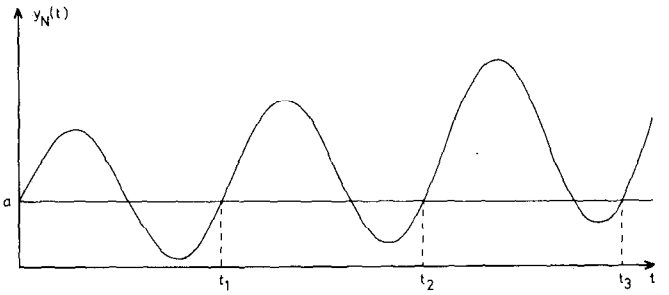

Fig. 3. Identification of the period of an autonomous system.
$V_{in}(t) = 10 \sin(120\pi t)$, $I_d(y_1) = 10^{-6}(\exp(40y_1) - 1)$.

After a Choleski decomposition of $\underline{L}^t\underline{L}$ the system (6.4) can be solved by one forward and two backward substitutions.

The least-squares problem could also be solved by forming the normal equations from (6.2). The resulting square matrix is, however, likely to be very ill-conditioned because the condition number of $\underline{D}^t\underline{D}$ is the squared value of the condition number of $\underline{D}$. A squaring of condition numbers also takes place in computing $\underline{L}^t\underline{L}$, but $\underline{L}$ is usually well conditioned whereas $\underline{U}$ is ill-conditioned if $\underline{D}$ is [23].

The strategy for computing the minimum polynomial is then as follows. As the vectors $\Delta y_{q+1}, \Delta y_{q+2} \cdots$ become available, perform a LU-factorization column-wise, and solve the upper square part of the system (6.3). Evaluate the relative residuals $\|r_s\|/\|c_s\|$ and decide when $s = m$, the degree of the minimum polynomial, is reached. Then form the normal system of equations (6.4) and solve for c.

The approximate minimum polynomials computed from (6.3) and (6.4) have not been subject to extensive comparisons, but when compared the least-squares approach (6.4) was the better, and it is recommended as it only adds little extra computational cost to Krylov's method.

Automonous systems

The calculation of a sequence $\tilde{y}_0, \tilde{y}_1, \cdots$ by the formula $\tilde{y}_{r+1} = G(\tilde{y}_r)$ (B6) can be interpreted as integrating the autonomous system (B1) from $y(0) = (\tilde{y}_0, a)$. The values $\tilde{y}_1, \tilde{y}_2, \cdots$ are the first $N-1$ components of $y(t_1), y(t_2), \cdots$ where $a = y_N(0) = y_N(t_1) = \cdots$. See Fig. 3.

This approach involves several difficulties. First a must be found so that $y_N(t)$ actually assumes the value a in the time interval necessary for extrapolation. The time instants t_1, t_2 are not just the values for which $y_N(t) = a$. We want $t_{r+1} - t_r \to T$, where T is the period, for $r \to \infty$. For the sine-like graph in Fig. 3, $t_1, t_2, \cdots$ can be taken as every other time instant for which $y_N(t) = a$. However, this might not work for a more complicated function. Finally, the accurate calculations of $t_1, t_2, \cdots$ must be considered.

A relatively good estimate of the steady-state period T is usually available for an autonomous system. With the given initial vector the system (B1) is integrated from zero to the estimated period, and the maximum and the minimum of $y_N(t)$ is found in this interval. The average of these extremes is then used for a. The variable chosen as

y_N (renumbering) should have a large amplitude in the steady-state compared to the amplitude of the transient terms, and it should not contain rapid oscillations compared to the fundamental oscillation.

Based on the estimate T_e of the period an interval $[T_e(1-\Delta), T_e(1+\Delta)]$ is defined. This interval is searched for a value t_1 such that $y_N(t_1)=a$. If no crossing or several crossings are found, an error message is given. For $T_1=t_1-0$ the next interval is $[t_1+T_1(1-\Delta), t_1+T_1(1+\Delta)]$, etc. In Examples 3 and 4 in Section VII the value $\Delta=0.1$ is used.

Once a zero-crossing of $y_N(t)-a$ has been detected, the precise value of t_r, where $y_N(t_r)=a$, must be found. In the experimental implementation used for the examples in Section VII, the differential equations are integrated by backward differentiation formulas [24], [25]. The old information is stored as divided differences. This formulation lends itself to interpolation by Newton's interpolation formula with divided differences [26].

$$y_N(t)=y_N(\tau_0)+\sum_{r=1}^{k}\prod_{s=0}^{r-1}(t-\tau_s)y_N[\tau_0,\tau_1,\cdots,\tau_r]. \quad (6.5)$$

The expression (6.5) is a polynomial in t, and t_r, where $y_N(t_r)=a$, is easily found by Newton iteration. When t_r is found, the interpolation formula (6.5) is used for the rest of the components of $y(t)$ to find $y(t_r)$.

VII. Examples

The examples in this section are produced by special-purpose analysis programs based on state-equation formulation of the electrical circuits. Both the scalar and vector ϵ-algorithms and the minimum polynomial extrapolation algorithm are, however, implemented in the general circuit analysis program WATAND (WATerloo ANalysis and Design) [27], [28].

The extrapolation algorithms in WATAND have been used extensively in a project of designing a bipolar transistor for a class C microwave amplifier. Extrapolation has also proved to be useful for analysing class E amplifiers.

In [8] extrapolation is used in connection with extensive distortion analyses of a RF power amplifier. For this project one of the special purpose programs was used, and it was mainly the scalar ϵ-algorithm that was employed.

For the examples the state-equations $y'=f(t,y)$ and the Jacobian $\partial f/\partial y$ are programmed as Fortran subroutines, and the integration of the differential equations is performed by a variable order variable step-length implementation of the backward differentiation formulas of orders 1–6 [25]. The specified local truncation error δ is compared with a weighted estimate of the local truncation error, and the step-length of the integration formula is adjusted accordingly. For each solution component the weight is chosen as the numerically largest value encountered so far.

Example 1

The first example is the dc power supply described in [3] (see Fig. 4).

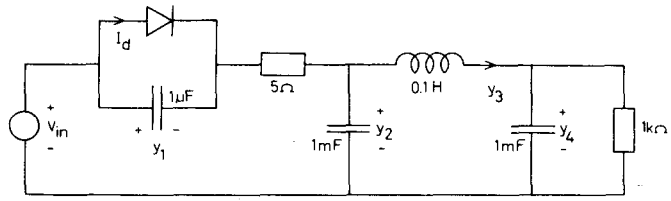

Fig. 4. Simple dc power supply [3].

TABLE I

	Scalar ϵ-algorithm $m=4, q=0, \delta=10^{-6}$		Aprille and Trick [3]	
Iteration	$\lvert x_2^{(n)}-z_2\rvert$	$\lvert x_3^{(n)}-z_3\rvert$	$\lvert x_2^{(n)}-z_2\rvert$	$\lvert x_3^{(n)}-z_3\rvert$
1	0.75141	0.09720E-3	7.1034	3.5803E-1
2	0.00884	0.00013E-3	1.6497	4.9609E-2
3	0	0	0.2531	6.4540E-3
4			0.0155	0.3689E-3
5			0.0001	0.0018E-3
6	0	0	0	0
	$P(z_0)=5.6E-8$		$P(z_0)=8.4E-8$	

By inspection it is seen that the time-constant associated with the state-variable y_1 is much smaller than the remaining time constants. Closer analysis reveals that the fast transient has disappeared after 0.01 period ($T=1/60$). The Extrapolation Algorithm in Section IV can thus be used with the parameters $q=0.01$ and $m=3$. The preceding description only deals with integer values of q, but a noninteger value of q simply means: integrate from 0 to qT before applying extrapolation.

Table I shows a comparison with the Newton algorithm proposed in [3]. To permit a direct comparison of the convergence, $q=0$ and $m=4$ is chosen and $x^{(0)}=0$.

The point z_0 of the steady-state solution is

$$z_0=(-9.0762, 9.0573, 8.9953E-3, 9.1030)$$

and the error measure is $P(z_0)=5.6E-8$ where $P(z_0)=\delta^t(z_0)\delta(z_0)$ and $\delta(z_0)=F(z_0)-z_0$ as defined in the introduction. In Table I the error of the second and third component of $x^{(n)}$ is given. The Extrapolation Algorithm converges considerably faster than the Newton iteration. Both algorithms possess second-order convergence when $x^{(n)}$ is sufficiently close to z_0, but this point is reached after the first iteration for the Extrapolation Algorithm, whereas the Newton algorithm needs 3 iterations to get so far.

It is difficult to compare the computational efficiency of the Newton iteration and the Extrapolation Algorithm. The results in [3] are obtained with an integration method of order 2. This means that Newton iteration only adds little extra cost to the integration (a factor 2 or 3). The Extrapolation Algorithm was used in connection with the most sophisticated methods for numerical integration making it possible to perform the integration of one period considerably cheaper than with an integration method of order 2.

In Table II the extrapolation algorithm is compared with the gradient method described in [5]. One iteration with the extrapolation algorithm based on the scalar ϵ-

TABLE II

Scalar ϵ-algorithm $m=3, q=0.01, \delta=10^{-6}, x^{(0)}=0$		Gradient Method	
Iteration	$P(x^{(n)})$	Iteration	$P(x^{(n)})$
0	1.95E 1	0	8.36
1	5.06E-2	3	4.47E-1
2	1.72E-4	6	1.98E-1
3	1.40E-8	9	7.87E-5
		12	2.15E-8

TABLE III

Extrapolation Algorithm, $m=3$, $q=0.01$, $\delta=10^{-6}$, $x^{(0)}=0$

	ϵ-algorithms		
	scalar	vector	min. pol.
Iteration	$P(x^{(n)})$	$P(x^{(n)})$	$P(x^{(n)})$
0	1.95E 1	1.95E 1	1.95E 1
1	5.06E-2	5.28E-2	5.86E-2
2	1.72E-4	1.68E-4	1.77E-4
3	1.40E-8	2.16E-8	1.00E-8

algorithm (6 periods of itegration) corresponds to 3 iterations (3×2 periods) with the gradient method.

Before the gradient method was applied, the system was integrated for 3 periods from the initial vector 0. This accounts for the difference in $P(x^{(0)})$. In total, the Extrapolation Algorithm required 18 periods of integration whereas the gradient method required 27 periods.

Table III shows a comparison of scalar and vector ϵ-algorithms and the minimum polynomial extrapolation. There is no significant difference between the methods applied to this example, but the computational cost of the minimum polynomial method is only 4 periods/iteration compared to 6 periods/iteration for the ϵ-algorithms.

Example 2

The second example is the class C amplifier shown in Fig. 5. This example originates from [29] where the steady state is computed with Newton iteration. In [5] the gradient method is applied to this example.

The input and output circuits are almost identical. The poles with the transistor removed are as shown:

$$-1.36\text{E}9, -4.07\text{E}8 \pm 2.60\text{E}9i, -1.48\text{E}7 \pm 8.31\text{E}7i.$$

With the period $T=10^{-8}$ the transients corresponding to the real pole and the first complex pole pair have almost disappeared after one period. This analysis is valid for the output circuit when the transistor does not saturate, and it has approximate validity for the input circuit. The analysis leads to the following parameters for the Extrapolation Algorithm: $q=1$ and $m=4$. Table IV shows the results of various computations.

The first column of Table IV shows the scalar ϵ-algorithm applied according to the results of the linear analysis. Because of the simple transistor model there is no feedback in the transistor, and the transients of the output circuit are not transferred to the input circuit. Therefore, the scalar ϵ-algorithm with parameters $m=2$ and $q=2$ (column 2) works with one iteration delay compared to the first column. With $m=2$ the transients of the input

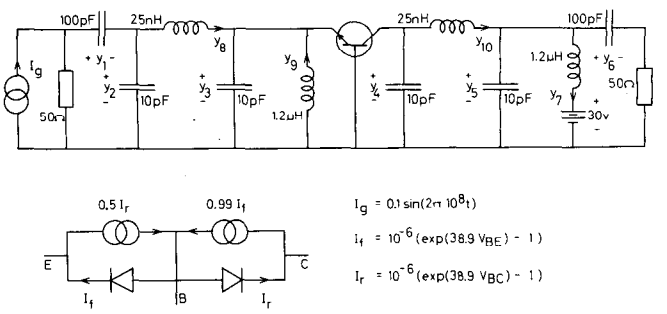

Fig. 5. Class C amplifier [29].

$I_f = 10^{-6}(\exp(40 V_{BE}) - 1)$, $I_r = 10^{-6}(\exp(40 V_{BC}) - 1)$

TABLE IV

Extrapolation Algorithm, $\delta=10^{-6}$, $x^{(0)}=0$

	ϵ-algorithm				
	scalar		vector	minimum polynomial	
	$m=4$, $q=1$	$m=2$, $q=2$	$m=4$, $q=1$	$m=4$, $q=1$	$m=5$, $q=1$
Iteration	$P(x^{(n)})$	$P(x^{(n)})$	$P(x^{(n)})$	$P(x^{(n)})$	$P(x^{(n)})$
1	6.71E-2	4.24	1.37E-2	16	15.4
2	5.97E-5	1.44E-1	2.34E-5	1.37E-2	1.73E-3
3	8.59E-11	5.27E-5	2.89E-6	4.84E-4	2.53E-7
4		1.80E-10		5.29E-6	

circuit are damped in the first iteration. In the second iteration the transients originating from the output circuit dominates in this part, and they are damped by the extrapolation, and the transients of the input circuit are damped even further.

This is an example of a circuit where the *scalar* ϵ-algorithm can exploit the partitioning imposed by a transistor. Another example is the push–pull amplifier analyzed in [8].

The third column shows the performance of the vector ϵ-algorithm. The initial convergence is very fast, but the final convergence of the vector algorithms (including the minimum polynomial method) seems to be more sensitive to the integration errors than the scalar ϵ-algorithm.

Columns 4 and 5 show results from the minimum polynomial method. The initial convergence is slower than for the ϵ-algorithms, but after the first iteration the convergence is quadratic.

The computational costs in the five cases are 9, 6, 9, 6, and 7 periods/iteration. The most efficient method for this problem is, therefore, the scalar ϵ-algorithm with the parameters $m=2$, $q=2$ needing 24 periods for 4 iterations.

A solution vector with $P(x^{(n)})=1.4\text{E}{-}11$ can be found by the gradient algorithm in 22 iterations and with a total computational cost of 49 periods [5]. Approximately the same precision is obtained after 7 iterations with Newton's method [29]. The computational cost is 22 periods. A modified Newton algorithm needs 10 iterations equivalent of 17 periods.

Example 3

This example is the van der Pol oscillator, which has been studied in [5] and [9] with $\mu=0.01$.

TABLE V

Scalar ϵ-extrapolation Algorithm, $m=1$, $q=1$, $\delta=10^{-7}$

Iteration	$P(x^{(n)})$	$x_1^{(n)}$	$x_2^{(n)}$	$T^{(n)}$
0	1.56	−4.39336E-2	−1.45157	6.28274
1	2.40E-3	−1.75964E-1	−2.51218	6.28473
2	6.01E-5	−1.80874E-1	−2.10210	6.28355
3	2.92E-7	−2.46750E-1	−1.99400	6.28325
4	2.53E-9	−7.16071E-2	−1.99908	6.28321
5	3.24E-10	−1.00225E-1	−1.99819	6.28322

TABLE VI

Scalar ϵ-extrapolation Algorithm, m = 3, q = 1, $\delta = 10^{-7}$

Iteration	$P(\underline{x}^{(n)})$	$T^{(n)}$
0	6.24E-3	1.21102E-4
1	1.81E-9	1.22203E-4
2	3.03E-11	1.22203E-4

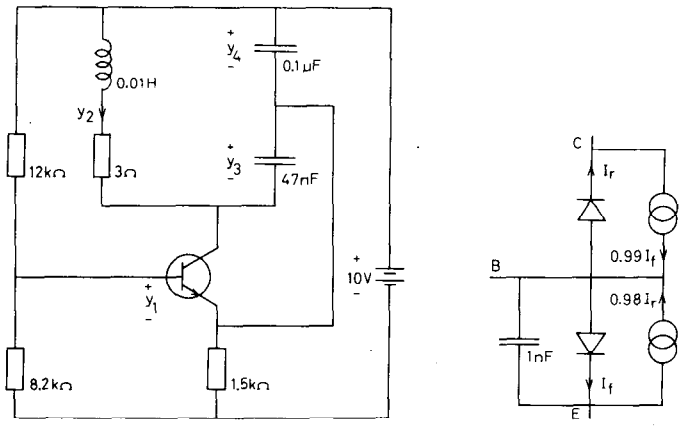

Fig. 6. Colpitts oscillator [9].

$$y_1' = y_2$$
$$y_2' = -y_1 + \mu(1-y_1^2)y_2.$$

With such a small value of μ, the system is close to the linear system which has $T=2\pi$. The variable y_1 is assigned a fixed value at $0, t_1, t_2, \cdots$ for the application of extrapolation.

Table V shows the results of a steady-state analysis. The initial values $x^{(0)}=(-1,-1)$ and $T_e=6$ were chosen. Each iteration involves 2 periods of integration plus one initial to find an appropriate value for y_1. The value of $P(x^{(n)})$ given in the table includes $T^{(n)}$. The first variable y_1 does not contribute as it is kept fixed to find T. The total cost of the extrapolation is 15 periods of integration.

A comparable accuracy is obtained by the gradient method in 8 iterations corresponding to 16 periods [5]. For the Newton method 15 iterations were needed, while a modified Newton method only needed 8 iterations.

Example 4

This example is a Colpitts oscillator which was analyzed in [5] and [9]. The diagram, also showing the state-variables, is in Fig. 6.

With the initial values $x^{(0)}=(0,0,0,10)$ and $T_e=10^{-4}$ the steady-state problem was solved by extrapolation. The results are shown in Table VI, where $P(x^{(n)})$ also includes the period $T^{(n)}$. The value of $P(x^{(0)})$ is after one leading period (with T_e) of integration. The basis–emitter voltage, y_1 is used for finding the period.

By the gradient method the error measure is $P(x^{(12)})=$ 2.55E-6 after 12 iterations [5]. The Newton method uses 6 iterations to obtain a solution with an error measure $P(x^{(6)})=2.55\text{E-}7$ [9].

VIII. Conclusion

The Extrapolation Algorithm has been presented and analyzed in connection with either scalar or vector ϵ-algorithms or the minimum polynomial algorithm. The Extrapolation Algorithm applied to nonlinear problems is shown to be quadratically convergent under fairly mild conditions, and to the best of our knowledge this result is new for the minimum polynomial method. The formulation of the *G*-function makes it possible also to use extrapolation for the steady-state analysis of autonomous systems.

Both Newton's method and extrapolation have quadratic convergence when the iterated value is close to the solution. While the global convergence properties of Newton's method were relatively poor in the examples given in Section VII, the global convergence properties of the extrapolation methods were excellent. In most of the examples the rate of convergence *slows down* to second order when the solution is approached, and this is the case even for strongly nonlinear systems.

The Extrapolation Algorithm is well suited for small-to-medium size circuits. The maximum size of circuits for which the extrapolation should be used can not be specified as the efficiency depends on the number of reactive elements causing slowly decaying transients and not on the total size of the circuit.

The Extrapolation Algorithm is very easy to implement in existing transient analysis programs as it operates only on solution vectors. The implementation is independent of circuit equation formulation and integration method. Therefore, the most convenient circuit formulation can be used, and advantage can be taken of sophisticated methods for numerical integration.

Appendix A

Let an initial value problem be given by

$$y' = f(t,y), \quad y(t_0)=y_0, \qquad t\in[t_0,t_f]. \tag{A1}$$

The functions y and f are real-valued vector functions, y: $[t_0,t_f]\to R^N$, f:$[t_0,t_f]\times R^N \to R^N$, and f is periodic in t with period T. Assume that f is differentiable with a Lipschitz continuous Jacobian such that

$$f(t,x)-f(t,y)=\frac{\partial f(t,y)}{\partial y}(x-y)+0(\|x-y\|^2). \tag{A2}$$

Because of the differentiability of f the problem (A1) has a unique solution.

Let the Jacobian of f along the solution $y(t)$ of (A1) (with initial vector y_0) be denoted by $\underline{J}(t,y_0)$. The Jacobian defines the variational equation

$$x' = \underline{J}(t,y_0)x, \qquad x(t_0) = x_0. \tag{A3}$$

The matrix solution $\underline{\Phi}(t,y_0)$ of (A3) for the initial matrix $\underline{\Phi}(t_0,y_0) = \underline{I}$ is called a fundamental matrix or state transition matrix and $x(t) = \underline{\Phi}(t,y_0)x_0$.

Let $y(t;t_0,y_0)$ denote the solution of (A1) as a function of t and the initial vector y_0 at t_0. Then the following theorem can be proved.

Theorem A1

Under the smoothness conditions of (A2) the solution $y(t;t_0,y_0)$ of (A1) is a differentiable function of y_0:

$$\partial y/\partial y_0 = \underline{\Phi}(t,y_0). \tag{A4}$$

The derivative is Lipschitz continuous such that

$$y(t;t_0,y_1) - y(t;t_0,y_0) = \underline{\Phi}(t,y_0)(y_1 - y_0) + 0(\|y_1 - y_0\|^2). \tag{A5}$$

■

This theorem is proved in [30] under the assumption $o(\|x-y\|)$ in (A2) but the proof is readily extended to the case $0(\|x-y\|^2)$.

The initial value problem (A1) can also be formulated as an integral equation:

$$y(t) = y_0 + \int_{t_0}^{t} f(\tau, y(\tau))\,d\tau, \qquad t \in [t_0, t_f].$$

The integration of one period can thus be formulated as

$$y(t+T) = y(t) + \int_{t}^{t+T} f(\tau, y(\tau))\,d\tau, \qquad t \in [t_0, t_f - T]. \tag{A6}$$

Let a vector-valued vector function $F: R^N \to R^N$ be defined by

$$F(w;T,t) = w + \int_{t}^{t+T} f(\tau, y(\tau;t,w))\,d\tau \tag{A7}$$

where T and t are parameters of F while w is the argument. When the choice of T and t are obvious, they will be omitted so that $F(w) \equiv F(w;T,t)$. In the notation of (A7) relation (A6) can be expressed by $y(t+T) = F(y(t);T,t)$.

From Theorem A1 it follows immediately that the function F is differentiable with respect to w and with a Lipschitz continuous derivative $\underline{F}'(y(t)) = \underline{\Phi}(t+T, y(t))$. The relation (1.4) is the analog of (A5).

A periodic solution $z(t)$ is defined to be asymptotically stable if for any $\epsilon > 0$ there exists a $\delta > 0$ such that

$$\|y(t_0) - z(t_0)\| < \delta \Rightarrow \|y(t) - z(t)\| < \epsilon, \qquad \text{for all } t \geq t_0$$
$$\|y(t) - z(t)\| \to 0, \qquad \text{for } t \to \infty.$$

where $y(t)$ and $z(t)$ are solutions of the differential equation in (A1)

Let the solution $z(t)$ of the differential equation (A1) with $z(t_0) = z_0$ be periodic with period T. In [31] the following theorem is proved.

Theorem A2

If the moduli of the eigenvalues of $\underline{\Phi}(t_0+T, z_0)$ are all smaller than one, then the periodic solution $z(t)$ of (A1) is asymptotically stable as $t \to \infty$. ■

When $z(t)$ is asymptotically stable, $y(t_0+nT)$ where $y(t_0+nT) = F(y(t_0+(n-1)T))$ will converge towards $z(t_0)$ for $n \to \infty$ if $\|y(t_0) - z(t_0)\| < \delta$. This follows immediately from the definition of asymptotic stability and the periodicity of $z(t)$. The mapping F is, therefore, a contraction for $\|y(t_0) - z(t_0)\| < \delta$.

The Lipschitz continuity of $\underline{F}'(y)$, proved in Theorem A1, and the nonsingularity of $\underline{F}'(z_0) - \underline{I}$ are necessary conditions of Theorem 4.1 for the quadratic convergence of the Extrapolation Algorithm.

Appendix B

The mathematical model of an oscillator does not have an explicit t-dependence, but is modeled by an autonomous system of ordinary differential equations:

$$y' = g(y), \quad y(t_0) = y_0, \qquad t \in [t_0, t_f]. \tag{B1}$$

The functions y and g are real-valued vector functions, y: $[t_0, t_f] \to R^N$ and g: $R^N \to R^N$. It is assumed that g is Lipschitz continuous such that (B1) has a unique solution.

Assume that (B1) has a periodic steady-steady solution $z(t)$ with period T. For

$$F(w;p,t) = w + \int_{t}^{t+p} g(y(\tau;t,w))\,d\tau \tag{B2}$$

we must determine $z_0 = z(t_0)$ and T such that $F(z_0;T,t_0) = z_0$. This problem has N equations and $N+1$ unknowns, but by an approach similar to that in [9] one of the unknowns can be eliminated. By assuming that element r of the steady-state solution $z(t)$ is equal to a for $t = t_a, z_r(t_a) = a$, we obtain the equation

$$F(z(t_a);T,t_a) = z(t_a) \tag{B3}$$

which has N equations and N unknowns.

In order to apply the contraction principle and extrapolation, (B3) is reformulated as follows. Without loss of generality we set $r = N$ (the vector elements can be renumbered) and solve

$$F_N(w_1, w_2, \cdots, w_{N-1}, a; p, t_a) = a \tag{B4}$$

for p giving

$$p = F_N^{-1}(w_1, w_2, \cdots, w_{N-1}, a; t_a). \tag{B5}$$

Define the vector $\tilde{w}$ by $\tilde{w} = (w_1, w_2, \cdots, w_{N-1})^t$ and the function $G: R^{N-1} \to R^{N-1}$ by

$$G(\tilde{w};a,t_a) = \begin{bmatrix} F_1(\tilde{w},a;p,t_a) \\ F_2(\tilde{w},a;p,t_a) \\ \cdots\cdots\cdots \\ F_{N-1}(\tilde{w},a;p,t_a) \end{bmatrix} \tag{B6}$$

where $\tilde{w}$ is the variable and a and t_a are parameters; $p(\tilde{w},a;t_a)$ is given by (B5). Notice that because we are dealing with autonomous systems, t_a need not be known, and we can set $t_a = 0$.

The Extrapolation Algorithm can be applied to G for finding the periodic steady-state solution of (B1) and sufficient conditions on G for quadratic convergence of the Extrapolation Algorithm will now be given.

Theorem B1

Assume that $g(y)$ (B1) has a Lipschitz continuous Jacobian so that G fulfills a relation similar to (A2). If $|g_N(x)| \geq \delta, \delta > 0$ for all $x \in D \subset R^N$, then

$$G(\tilde{x}_n) - G(\tilde{y}_n) = (\underline{G}'(\tilde{y}_n))(\tilde{x}_n - \tilde{y}_n) + 0(\|\tilde{x}_n - \tilde{y}_n\|^2)$$

for $x_r, y_r \in D, r = 0, 1, \cdots$ where $x_r = (\tilde{x}_r, a)$ and $y_r = (\tilde{y}_r, a)$.

Proof: The proof is carried out by showing that $\underline{G}'$ exists and is Lipschitz continuous. The derivative $\underline{G}'$ is the Jacobian of G with respect to $\tilde{y}$. From (B6) we find

$$\underline{G}'(\tilde{x}) = \partial \tilde{F}(x)/\partial \tilde{y} + (\partial \tilde{F}(x)/\partial p)(\partial p(\tilde{x})/\partial \tilde{y})^t \quad \text{(B7)}$$

where $\tilde{F} = (F_1, F_2, \cdots, F_{N-1})$.

$\partial \tilde{F}(x)/\partial \tilde{y}$ is the upper left hand $(N-1)\times(N-1)$ part of $\partial F(x)/\partial y$ where $x = (\tilde{x}, a)$. From the assumptions and Theorem A1 it follows that $\partial \tilde{F}(x)/\partial \tilde{y}$ exists and is Lipschitz continuous with respect to $\tilde{x}$.

From (B2) we find

$$\partial F(x)/\partial p = g(y(p; 0, x)). \quad \text{(B8)}$$

As g is Lipschitz continuous and y is Lipschitz continuous with respect to x (y is differentiable with Lipschitz continuous derivative) so is $\partial F(x)/\partial p$.

From (B4) we find

$$\begin{aligned} \partial p(\tilde{x})/\partial \tilde{y} &= -(\partial F_N(x)/\partial \tilde{y})/(\partial F_N(x)/\partial p) \\ &= -(\partial F_N(x)/\partial \tilde{y})/g_N(y(p;0,x)). \end{aligned} \quad \text{(B9)}$$

As $\partial F_N/\partial \tilde{y}$ and g_N are both Lipschitz continuous with respect to $\tilde{x}$ and g_N is bounded away from zero, then $\partial p/\partial \tilde{y}$ is Lipschitz continuous.

The derivative $\underline{G}'(\tilde{x})$ is, therefore, Lipschitz continuous because the individual terms of the expression (B7) are Lipschitz continuous. ■

A periodic solution $z(t)$ of the differential equation (B1) is defined to be asymptotically orbitally stable if there exists an $\epsilon > 0$ such that $\|y(t_1) - z(t_2)\| < \epsilon$ implies that there exists a constant c giving

$$\|y(t) - z(t + c)\| \to 0, \qquad \text{for } t \to \infty$$

where $y(t)$ and $z(t)$ are solutions of (B1).

Let the Jacobian $\underline{J}(t, z_0)$ be defined as in Appendix A. The variational equation (A3) defines the state transition matrix $\underline{\Phi}(t, z_0)$.

In [31] the following theorem is proved.

Theorem B2

When $\underline{\Phi}(t_0 + T, z_0)$ has one eigenvalue at 1, the rest of the eigenvalues having moduli less than 1, then the periodic solution $z(t)$ of (B1) is asymptotically orbitally stable for $t \to \infty$. ■

When $(\tilde{y}_0, a)$ is close enough to the point $z(t_a)$, where $z_N(t_a) = a$, of an asymptotically orbitally stable periodic solution $z(t)$, the sequence $\tilde{y}_0, \tilde{y}_1, \cdots$ generated by $\tilde{y}_{r+1} = G(\tilde{y}_r)$ will converge towards $\tilde{z}(t_a)$ and not just towards $\tilde{z}(t_a + c)$ for some c.

The steady-state solution can be computed by extrapolation provided $\underline{G}'(\tilde{z}(t_a))$ does not have an eigenvalue at unity.

Theorem B3

When $\underline{\Phi}(T, z_0)$ where z_0 is on a periodic steady-state solution, has one eigenvalue at 1, the rest of the eigenvalues having moduli less than 1, and $g_N(z_0) \neq 0$, then $\underline{G}'(\tilde{z}_0) - \underline{I}$ is nonsingular.

Proof: In [9] it is proved that

$$\underline{Q} = (\Phi_1, \Phi_2, \cdots, \Phi_{N-1}, g(z_0)) - \text{diag}\,(1, 1, \cdots, 1, 0) \quad \text{(B10)}$$

is nonsingular. Column r of $\underline{\Phi}(T, z_0)$ is denoted by Φ_r, $r = 1, 2, \cdots, N-1$.

As $\underline{\Phi}(T, z_0)$ is equal to the derivative $\underline{F}'(z_0)$ at the steady-state solution, we can write the matrix $\underline{Q}$ in (B10) as follows:

$$\underline{Q} = \begin{bmatrix} \underline{\tilde{F}}' - \underline{\tilde{I}} & \tilde{g} \\ (\partial F_N/\partial \tilde{y})^t & g_N \end{bmatrix}.$$

The $\tilde{}$ denotes the first $N-1$ elements of the vectors and the upper left $(N-1)\times(N-1)$ part of a matrix.

According to [32] the upper left $(N-1)\times(N-1)$ part of $\underline{Q}^{-1}$ can be expressed by

$$\underline{M}^{-1} = \left(\underline{\tilde{F}}' - \underline{\tilde{I}} - \tilde{g}(\partial F_N/\partial \tilde{y})^t/g_N\right)^{-1}.$$

Comparing with (B7) and (B9) we see that

$$\underline{M} = \underline{G}'(\tilde{z}_0) - \underline{\tilde{I}}$$

which is thus proved to be nonsingular. ■

The Lipschitz continuity of $\underline{G}'(\tilde{y})$, proved in Theorem B1, and the nonsingularity of $\underline{G}'(\tilde{z}_0) - \underline{\tilde{I}}$, proved in Theorem B3, are necessary conditions of Theorem 4.1 for the quadratic convergence of the Extrapolation Algorithm.

ACKNOWLEDGMENT

The author is grateful to the Circuits Group in the Department of Electrical Engineering at the University of Waterloo, Waterloo, Ontario, Canada, for the interest and support of this research project.

REFERENCES

[1] S. Skelboe, "Extrapolation methods for computation of the periodic steady-state response of nonlinear circuits," in *Proc. 1977 IEEE Int. Symp. on Circuits and Systems*, Phoenix, AZ, pp. 64–67.

[2] S. W. Director, A. J. Brodersen, and D. A. Wayne, "A method for quick determination of the periodic steady-state in nonlinear networks," in *Proc. 9th Allerton Conf. on Circuit and System Theory*, Univ. of Illinois, Urbana-Champaign, pp. 131–139, 1971.

[3] T. J. Aprille and T. N. Trick, "Steady-state analysis of nonlinear circuits with periodic inputs," *Proc. IEEE*, vol. 60, pp. 108–114, 1972.

[4] P. H. Strohband, R. Laur, and W. L. Engl, "TNPT—An efficient method to simulate forced nonlinear RF networks in time domain," *IEEE J. Solid-State Circuits*, vol. SC-12, pp. 243–246, 1977.

[5] M. S. Nakhla and F. H. Branin, "Determining the periodic response of nonlinear systems by a gradient method," *Int. J. Circuit Theory Appl.*, vol. 5, pp. 255–273, 1977.

[6] M. S. Nakhla and J. Vlach, "A piecewise harmonic balance technique for determination of periodic response of nonlinear systems," *IEEE Trans. Circuits Syst.*, vol. CAS-23, pp. 85–91, 1976.

[7] T. B. M. Neill and J. Stefani, "Self regulating Picard-type iteration for computing the periodic response of a nearly linear circuit to a periodic input," *Electron. Lett.*, vol. 17, pp. 413–415, 1975.

[8] S. Skelboe, "Simulation of distortion in a *RF* power amplifier," in *Proc. 1978 IEEE Int. Symp. on Circuits and Systems*, New York.

[9] T. J. Aprille and T. N. Trick, "A computer algorithm to determine the steady-state response of nonlinear oscillators," *IEEE Trans. Circuit Theory*, vol. CT-19, pp. 354–360, 1972.
[10] E. Gekeler, "On the solution of systems of equations by the epsilon algorithm of Wynn," *Math. Comp.*, vol. 26, pp. 427–436, 1972.
[11] J. H. Wilkinson, *The Algebraic Eigenvalue Problem*. London, England: *Oxford Univ.*, 1965.
[12] C. Brezinski and A. C. Rieu, "The solution of systems of equations using the ϵ-algorithm, and an application to boundary-value problems," *Math. Comp.*, vol. 28, pp. 731–741, 1974.
[13] P. Wynn, "On a device for computing the $e_m(S_n)$ transformation," *MTAC*, vol. 10, pp. 91–96, 1956.
[14] ——, , "Acceleration techniques for iterated vector and matrix problems," *Math. Comp.*, vol. 16, pp. 301–322, 1962.
[15] ——, , "Continued fractions whose coefficients obey a non-commutative law of multiplication," *Arch. Rat. Mech. Anal.*, vol. 12, pp. 273–312, 1963.
[16] J. B. McLeod, "A note on the ϵ-algorithm," *Computing*, vol. 7, pp. 17–24, 1971.
[17] J. R. Schmidt, "On the numerical solution of linear simultaneous equations by an iterative method," *Philos. Mag.*, Ser. 7, vol. 32, pp. 369–383, 1941.
[18] S. Cabay and L. W. Jackson, "A polynomial extrapolation method for finding limits and antilimits of vector sequences," *SIAM J. Numer. Anal.*, vol. 13, pp. 734–752, 1976.
[19] P. Wynn, "On the propagation of error in certain non-linear algorithms," *Numer. Math.*, vol. 1, pp. 142–149, 1959.
[20] ——, , "On the convergence and stability of the epsilon algorithm," *J. SIAM Numer. Anal.*, vol. 3, pp. 91–122, 1966.
[21] R. J. Hanson and C. L. Lawson, "Extensions and applications of the House-holder algorithm for solving linear least squares problems," *Math. Comp.*, vol. 23, pp. 787–812, 1969.
[22] C. W. Ho, A. E. Ruehli, and P. A. Brennan, "The modified nodal approach to network analysis," *IEEE Trans. Circuits and Systems*, vol. CAS-22, pp. 504–509, 1975.
[23] G. Peters and J. H. Wilkinson, "The least squares problem and pseudo inverses," *Computer Journal*, vol. 13, pp. 309–316, 1970.
[24] S. Skelboe, "The control of order and steplength for backward differentiation methods," *Nordistc Tidsstcrift for Informations Behandling*, vol. 17, pp. 91–107, 1977.
[25] ——, "INTGR for integration of stiff systems of ordinary differential equations", Inst. of Circuit Theory and Telecommunication, Technical University of Denmark, Rep. IT9, Mar. 1977.
[26] G. Dahlquist and A. Bjork, *Numerical Methods*. New York: Prentice Hall, 1974.
[27] I. N. Hajj, K. Singhal, and J. Vlach, "Efficient analysis of nonlinear networks by piecewise-linear approximation," in *Proc. 11th Allerton Conf. on Circuit and System Theory*, Allerton House, IL, pp. 635–642, Oct. 1973.
[28] I. Hajj and S. Skelboe, "Time-domain analysis of piecewise-linear networks," *Proc. 1978 European Conf. on Circuit Theory and Design*, Sept. 4–8, Lausanne, Switzerland.
[29] F. R. Colon and T. N. Trick, "Fast periodic steady-state analysis for large-signal electronic circuits," *IEEE J. Solid-State Circuits*, vol. SC-8, pp. 260–269, 1973.
[30] E. Hille, *Lectures on Ordinary Differential Equations*, New York: Addison-Wesley, 1969.
[31] E. A. Coddington and N. Levinson, *Theory of Ordinary Differential Equations*, New York: McGraw-Hill, 1955.
[32] B. Noble, *Applied linear algebra*. New York: Prentice-Hall, 1969.

Stig Skelboe (S'72–M'73) was born in Denmark, in 1948. He received the M.S. and Ph.D. degrees in electrical engineering and numerical analysis, respectively, from the Technical University of Denmark, Lyngby, in 1973 and 1975.

From 1975 to 1977 he was a Research Fellow at the Institute for Circuit Theory and Telecommunication, Technical University of Denmark, and from 1977 to 1978 he was a Visiting Assistant Professor in the Department of Electrical Engineering, University of Waterloo, Ontario, Canada. His current research interests include numerical methods and computer-aided analysis of nonlinear systems. In August 1978, he joined Elektronikcentralen, Danish Research Centre for Applied Electronics, Hørsholm, Denmark.