TIME-DOMAIN BASED STEADY-STATE INITIALIZATION FOR THE EMTP

by

BRIAN KENNETH PERKINS

B.A.Sc., The Univeristy of Toronto, 1991

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF

THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF APPLIED SCIENCE

in

THE FACULTY OF GRADUATE STUDIES

DEPARTMENT OF ELECTRICAL ENGINEERING

We accept this thesis as conforming

to the required standard

THE UNIVERSITY OF BRITISH COLUMBIA

April 1993

In presenting this thesis in partial fulfilment of the requirements for an advanced degree at the University of British Columbia, I agree that the Library shall make it freely available for reference and study. I further agree that permission for extensive copying of this thesis for scholarly purposes may be granted by the head of my department or by his or her representatives. It is understood that copying or publication of this thesis for financial gain shall not be allowed without my written permission.

(Signature)

Department of Electrical Engineering

The University of British Columbia
Vancouver, Canada

Date April 21, 1993

DE-6 (2/88)

Abstract

The existing steady-state initialization algorithm in the electromagnetic transients program EMTP is based on a frequency-domain admittance method. This yields a simple, efficient solution for linear, lumped element networks. However, the basic algorithm cannot be easily extended to handle nonlinear and variable structure networks.

A time-domain based steady-state initialization algorithm is formulated for the three classes of lumped element networks, namely, linear, nonlinear and variable structure. The primary contribution of this work is that each network element does not require a dedicated steady-state model and a separate steady-state solution algorithm. Furthermore, as the proposed initialization process is consistent with the time-domain character of the transient simulations themselves, the proposed algorithm can be incorporated into existing electromagnetic transient programs with minimal programming effort.

Table of Contents

List of Tables

List of Figures

List of Algorithms

Chapter 1 Introduction

1.1 Background

The EMTP is a digital computer-based program for the solution of transients in large scale networks typical of power systems [1]. The ordinary differential equations (ODEs) describing the network are formulated in nodal form. Solution methods are based on the trapezoidal rule of integration, an implicit numerical method which insures stable solutions of networks with widely seperated eigenvalues.[1]

EMTP development can be classified into two major categories: network component modelling and solution methods. Much work has gone into the developement of computer-based models of power system components and their implementation within the basic EMTP framework. Examples are transmission line models, synchronous machine models, and HVdc converter models, to name just a few. The basic EMTP solution methods, however, have not undergone any significant changes thanks in large part to the inherent robustness of the original formulation. There have, nonetheless, been significant contributions in this area. Solution methods for networks with nonlinear or time-varying elements [2] and the critical damping adjustment scheme (CDA) [3] are especially noteworthy. TACS is a contribution that merits membership in both of the above categories.

1.2 Existing methods

The periodic steady-state solution of a network is of particular interest to the power systems engineer. Typically, transient studies are executed from the periodic steady-state. Alternatively, the steady-state solution may be of interest in itself for power flow calculations and harmonic analysis.

The existing steady-state initialization algorithm is based on a frequency-domain admittance matrix approach [1]. It is limited to the solution of linear, lumped element networks. Harmonic balance methods have been applied to networks containing nonlinear elements

[1] Such systems are said to be *stiff*.

[4][5]. However, these methods apply to only a small class of networks and have yet to find widespread implementation.

The main drawback of frequency-domain methods for steady-state initialization is that they make little or no use of the time-domain solve code common to electromagnetic transient programs; they are essentially standalone programs that interface with the EMTP.

1.3 Scope of the work

Various time-domain approaches for the periodic steady-state solution of systems of ordinary differential equations have been proposed [6]. These approaches are invariably based on a state-space formulation of the ODEs. Among them, the autonomous shooting method is by far the simplest and most fundamental. Reference [7] details the application of such a method to the periodic steady-state solution of nonlinear networks with periodic inputs.

The proposed steady-state initialization method based on the seminal work of Aprille and Trick [7] offers two significant advantages:

1. the basic formulation applies globally to linear, nonlinear and variable structure networks[2]
2. the method is consistent with the time-domain character of the transient simulations themselves

1.4 Thesis organization

Chapter 1 serves as a general introduction. Chapter 2 introduces the state-space formulation for a continuous-time-domain based steady-state initialization algorithm. Chapters 3, 4 and 5 detail the development of the time-domain based steady-state initialization algorithm within the framework of the EMTP solve code for three classes of lumped element networks:

1. multiphase linear, lumped element networks (Chapter 3)
2. nonlinear, lumped element networks (Chapter 4)
3. variable structure networks (Chapter 5).

[2] Networks with ideal-switch devices (e.g. ideal diodes).

Chapter 6 concludes the thesis.

Chapter 2 Steady-state initialization: State-space formulation

2.1 Introduction

In the analysis of networks driven by periodic inputs, one may be interested in determining the steady-state periodic response. Even for transient simulations, one typically begins at steady-state initial conditions. Unless the network is linear, time-invariant (LTI) for which simple frequency domain analysis yields the steady-state response, one generally has no alternative but to simulate the system over a sufficiently long interval of time for the transient waveform to die out. This procedure is satisfactory if the transient decays rapidly. However, for lightly damped networks typical of power systems the transient will decay very slowly and it becomes prohibitively expensive for the computer to simulate over such a long transient regime. Before presenting an algorithm for overcoming this difficulty, it is instructive to examine a simple LTI example to motivate the problem.

2.2 Motivating Example

Consider the simple RLC circuit shown in Fig. 2.1.

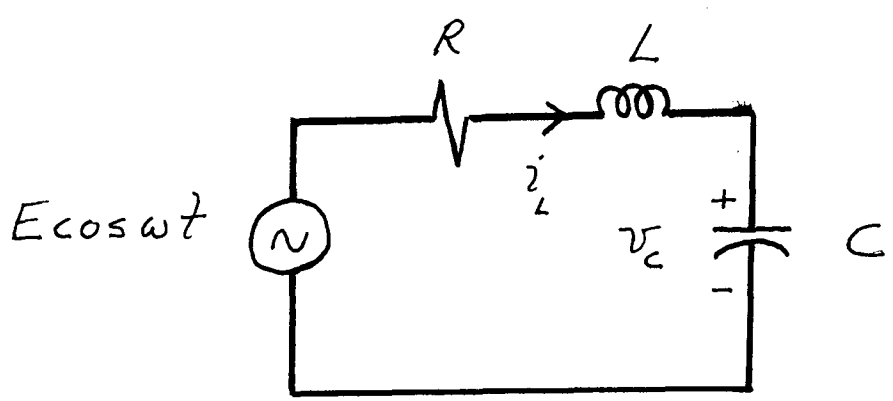

Figure 2.1 Simple RLC circuit.

The state equations are given by

$$\begin{aligned} \frac{d}{dt}v_C(t) &= \frac{1}{C}i_L(t) \\ \frac{d}{dt}i_L(t) &= \frac{1}{L}(E\cos\omega t - Ri_L(t) - v_C(t)) \end{aligned} \tag{2.1}$$

The complete response is given by

$$\begin{aligned} v_c(t) &= (k_1\cosh\beta t + k_2\sinh\beta t)\exp\alpha t + V_C\cos(\omega t+\theta) \\ i_L(t) &= (k_3\cosh\beta t + k_4\sinh\beta t)\exp\alpha t - \omega CV_C\sin(\omega t+\theta) \end{aligned} \tag{2.2}$$

where

$$\begin{aligned} V_C &= \frac{E}{\sqrt{(1-\omega^2LC)^2+(\omega CR)^2}} \\ \theta &= -\arctan\left[\frac{\omega CR}{1-\omega^2LC}\right] \\ \alpha &= -\frac{R}{2L} \qquad \beta = \sqrt{\left(\frac{R}{2L}\right)^2 - \frac{1}{LC}} \end{aligned} \tag{2.3}$$

The four arbitrary constants are determined by the initial states $v_C(0)$ and $i_L(0)$:

$$\begin{aligned} k_1 &= v_C(0) - V_C\cos\theta \\ k_2 &= \beta^{-1}\left[C^{-1}i_L(0) + \omega V_C\sin\theta - \alpha v_C(0) + \alpha V_C\cos\theta\right] \\ k_3 &= C[\alpha k_1 + \beta k_2], \quad k_4 = C[\alpha k_2 + \beta k_1] \end{aligned} \tag{2.4}$$

It follows from (2.4) that $k_1 = k_2 = k_3 = k_4 = 0$ if and only if

$$\begin{aligned} v_C(0) &= V_C\cos\theta \\ i_L(0) &= -\omega CV_C\sin\theta \end{aligned} \tag{2.5}$$

With this choice of initial states the complete solution is given by

$$\begin{aligned} v_C(t) &= V_C\cos(\omega t+\theta) \\ i_L(t) &= -\omega CV_C\sin(\omega t+\theta) \end{aligned} \tag{2.6}$$

both of which are periodic waveforms of period $T = \frac{2\pi}{\omega}$. Hence it is possible to suppress the transient component if the initial states are chosen as above.

Numerical example: Consider a lightly damped case of the simple RLC circuit with parameters

$$\begin{aligned} E = 1 \quad \omega = 100\pi \quad R = 0.04 \text{ ohms} \quad C = 200 \text{ uF} \quad L = 1 \text{ mH} \\ \alpha = -20 \quad \beta = \text{j}2236 \end{aligned} \tag{2.7}$$

Fig. 2.2 depicts the circuit response from zero initial conditions. The transient component takes over ten cycles to decay. Fig. 2.3 depicts the circuit response with steady-state initial conditions specified as in (2.5) corresponding to the steady-state solution

$$\begin{aligned} v_C(t) &= 1.0201 \cos(100\pi t - 0.0026) \\ i_L(t) &= -0.0641 \sin(100\pi t - 0.0026) \end{aligned} \tag{2.8}$$

§

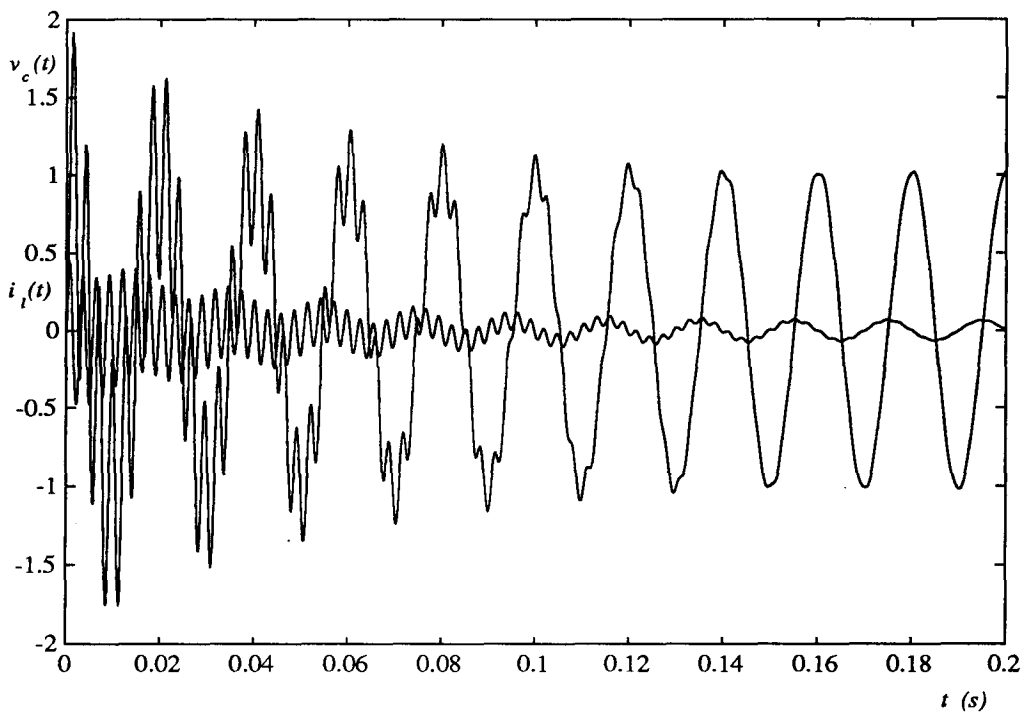

Figure 2.2 RLC circuit response from zero initial conditions

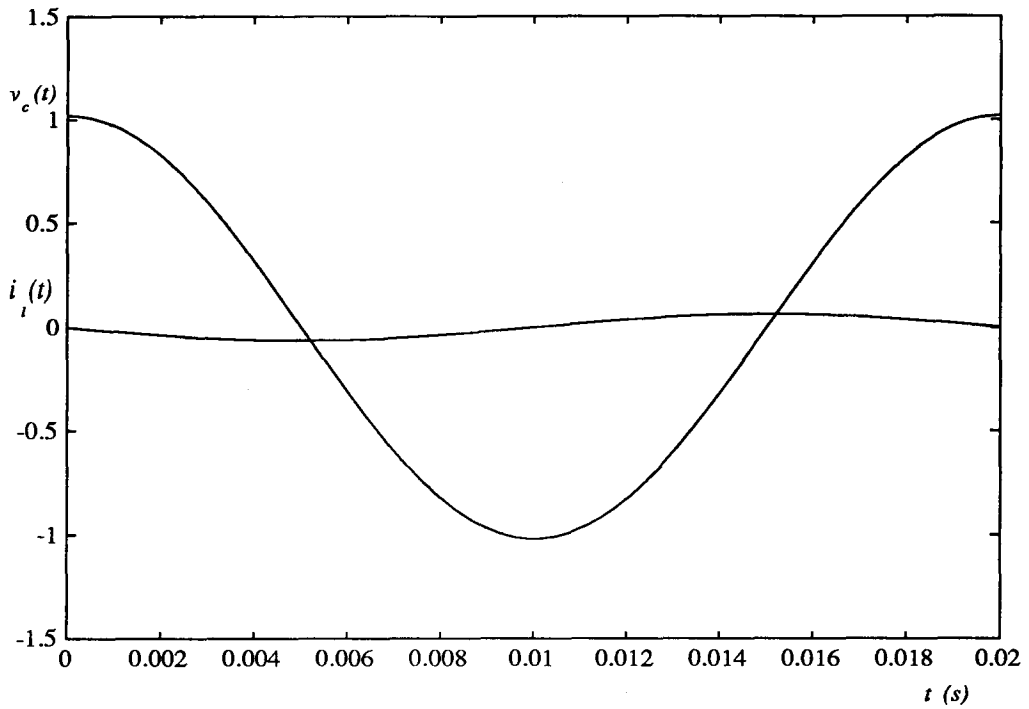

Figure 2.3 RLC circuit response from steady-state initial conditions

The objective is to generalize the concept of supressing the transient component and devise an algorithm for determining the appropriate initial state $x_0 = x(0)$ such that the complete solution to the state equations

$$\frac{dx}{dt} = f(x,t) \tag{2.9}$$

has no transient component under the assumption that all inputs are periodic in period T

$$f(x,t) = f(x,t+T) \tag{2.10}$$

where $f(\cdot)$ is possibly nonlinear.

2.3 Formulating the Fixed-Point Problem

Let $x(t)$ be the solution to (2.9) with initial state x_0. Integrating both sides of (2.9) from time 0 to time t we obtain

$$x(t) = \int_0^t f(x(t),t)dt + x_0 = x(t,x_0) \tag{2.11}$$

The right hand side of (2.11) is denoted by $x(t, x_0)$ to emphasize that the solution at any time depends on the initial state. The goal is to find x_0 such that at $t = T$, $x(T, x_0) = x_0$. Since this relationship is seldom satisfied for an arbitrary choice of x_0, define the function

$$F(x_0) = x(T, x_0) = \int_0^T f(x(t), t)dt + x_0 \tag{2.12}$$

Since we are fixing the time $t = T$, $F(x_0)$ is a function of x_0 and is independent of t. In the scalar case the function can be represented in the F versus x_0 plane. It follows from (2.12) that the initial state $x_0 = x_0^*$ which gives rise to a periodic solution $x_0^*(t)$ must satisfy the equation

$$x_0 = F(x_0) \tag{2.13}$$

which is identical to the standard form for a fixed-point iteration [8]. Hence the solution can be found by applying the fixed point iteration algorithm

$$x_0^{(i+1)} = F\left(x_0^{(i)}\right) \tag{2.14}$$

Invoking the assumption $f(x, t) = f(x, t+T)$ it follows that the $(i+1)$th iterate is given by

$$x_0^{(i+1)} = F\left(x_0^{(i)}\right) = \int_0^{(i+1)T} f(x(t), t)dt + x_0^{(0)} \tag{2.15}$$

This iteration must be repeated until

$$\left\| F\left(x_0^{(i)}\right) - x_0^{(i)} \right\|_2 < \epsilon \tag{2.16}$$

where ϵ is some prespecified quantity. The fixed-point iteration algorithm is equivalent to integrating (2.9) until all transient components have decayed.

Numerical example: Consider the simple RLC circuit of Section 2.2 with the states defined as $x(t) = [v_c(t) \quad i_L(t)]^T$. The progression of the fixed-point iteration can be calculated from the response (2.2). Table 2.1 summarizes the progression of the fixed-point iteration.

Note that in the LTI case, the speed of convergence is dependent on the eigenvalues of the system.§

i	$\left\|F\left(x_0^{(i)}\right) - x_0^{(i)}\right\|_2$
0	5.5786e-1
5	7.9206e-2
10	1.3779e-3
20	1.4568e-7
50	1.7792e-16

Table 2.1 Progresssion of the fixed-point iteration

2.4 State-space steady-state initialization algorithm

A more efficient algorithm is obtained by formulating (2.13) as a two-point boundary value problem [7]. Rewrite (2.13) in terms of an objective function defined

$$H(x_0) = F(x_0) - x_0 \tag{2.17}$$

for which the periodic steady-state initial conditions yield $H(x_0^*) = 0$. Convergence to x_0^* is accelerated by a Newton-Raphson iteration on (2.17)

$$x_0^{(i+1)} = x_0^{(i)} - \left[DF\left(x_0^{(i)}\right) - U\right]^{-1}\left[F\left(x_0^{(i)}\right) - x_0^{(i)}\right] \tag{2.18}$$

where $DF(x_0^{(i)})$ is the Jacobian matrix of $F(x_0)$ evaluated at $x_0^{(i)}$.

Thus we obtain quadratic convergence to the periodic steady-state as opposed to the possibly very slow convergence of the fixed-point iteration algorithm. In fact, for a LTI system convergence is realized in a single Newton-Raphson iteration.

Numerical example: For the simple RLC network of Section 2.2 the above algorithm yields

$$x_0^{(1)} = [\,1.0201 \quad -0.0641\,]^T \tag{2.19}$$

on the first iteration. The above is equivalent to the initial conditions obtained by (2.5).§

2.5 Evaluating the Jacobian Matrix

In general, one cannot obtain a closed form expression for the Jacobian matrix as in the LTI case. For a system of dimension n the Jacobian matrix is given by

$$DF(x_0) = \begin{bmatrix} \frac{\partial x_1(T,x_0)}{\partial x_1(0)} & \cdots & \frac{\partial x_1(T,x_0)}{\partial x_n(0)} \\ \vdots & \ddots & \vdots \\ \frac{\partial x_n(T,x_0)}{\partial x_1(0)} & \cdots & \frac{\partial x_n(T,x_0)}{\partial x_n(0)} \end{bmatrix} \tag{2.20}$$

Evaluating the Jacobian matrix by finite differences is inefficient and prone to numerical error as one must solve the possibly nonlinear differential equations $(n+1)$ times. A more accurate and efficient method involves the solution of related *sensitivity systems* which describe the linearization of the nonlinear system along the trajectory of the solution. This method involves the solution of the nonlinear system and its n associated linear sensitivity systems. The method is formulated as follows. Given that the possibly nonlinear system

$$\frac{dx}{dt} = f(x,t) \qquad x(0) = x_0 \tag{2.21}$$

has solution $x(t, x_0)$, then the n associated *sensitivity systems* are defined by

$$\frac{\partial}{\partial x_j(0)}\frac{dx}{dt} = \frac{\partial}{\partial x_j(0)} f(x,t) \quad \frac{\partial x_0}{\partial x_j(0)} = [0 \quad \cdots \quad 1 \quad \cdots \quad 0]^T \quad \{j = 1..n\} \tag{2.22}$$

whose solutions at time T represents the $j = 1..n$ columns of the Jacobian matrix.

Rearranging (2.22) the sensitivity systems become

$$\frac{d}{dt}\frac{\partial x}{\partial x_j(0)} = \frac{\partial f}{\partial x}\Big|_{x(t,x_0)} \cdot \frac{\partial x}{\partial x_j(0)} \tag{2.23}$$

where $\frac{\partial f}{\partial x}$ represents the first variation (linearization) of f along the solution trajectory. In the application of implicit integration methods for the solution of (2.9), $\frac{\partial f}{\partial x}$ is readily available.

2.6 Application to the EMTP

The state-space formulation presented in this chapter is fine for pedagogical purposes but impractical for large scale networks. The work details a formulation of the steady-state initialization algorithm within the framework of fixed time-step electromagnetic transients programs (EMTP). The algorithm is formulated for the three classes of lumped element networks, namely linear, nonlinear, and variable structure.

Chapter 3 Steady-state initialization: Linear Networks

3.1 Introduction

A time-domain based steady-state initialization algorithm for linear, lumped element networks within the framework of electromagnetic transient programs is presented in this chapter. The formulation and solution of such networks is also presented to facilitate the derivation of the steady-state initialization algorithm.

3.2 Formulation of the network equations

The linear equations of the discrete-time network can be formulated as follows [1]

$$[G][V(t)] = [J(t)] = [A_h][J_h(t)] + [A_j][J_j(t)] \tag{3.24}$$

where $[G]$ is the symmetric nodal conductance matrix, $[V(t)]$ is the vector of node voltages, $[J(t)]$ is a vector containing contributions from Norton equivalent history sources $[J_h(t)]$ and injected current sources $[J_j(t)]$. $[A_h]$ and $[A_j]$ are incidence matrices mapping branch current contributions onto the nodal current contributions.

The vector of node voltages is partitioned into unknown $[V_A(t)]$ and known $[V_B(t)]$ voltages. Subdividing the matrices accordingly, the unknown node voltages at each time-step are given by

$$[G_{AA}][V_A(t)] = [J_A(t)] - [G_{AB}][V_B(t)] \tag{3.25}$$

where

$$[J_A(t)] = [A_{hA}][J_h(t)] + [A_{jA}][J_j(t)] \tag{3.26}$$

The Norton equivalent history sources are updated recursively as follows[4]

$$[J_h(t + \Delta t)] = [U_h]([J_h(t)] + 2[G_h][A_v][V(t)]) \tag{3.27}$$

[4] This formulation is applicable to both monophase and multiphase networks.

where

$$[U_h] = \text{diag}\begin{pmatrix} +[U], & [J_h] \text{ associated with } [L] \\ -[U], & [J_h]\text{associated with } [C] \end{pmatrix} \tag{3.28}$$

$$[G_h] = \text{diag}\begin{pmatrix} \frac{\Delta t}{2}[L]^{-1}, & [J_h] \text{ associated with } [L] \\ \frac{2}{\Delta t}[C], & [J_h] \text{ associated with } [C] \end{pmatrix} \tag{3.29}$$

$$[A_v] = -[A_h]^T \tag{3.30}$$

3.3 Triangularization

The network equations are simplified prior to the first time step by upper triangularization of $[G_{AA}]$. The triangularization proceeds by downward operations as follows

$$[\,G_{AA} \quad G_{AB} \quad A_{hA} \quad A_{jA}\,] \leftarrow [P][\,G_{AA} \quad G_{AB} \quad A_{hA} \quad A_{jA}\,] \tag{3.31}$$

where $[P]$ represents the permutation matrix that upper triangularizes $[G_{AA}]$.

3.4 Solution method

For the above formulation a solution method akin to that of the EMTP is presented in Algorithm 3.1. Note that the algorithm presented differs slightly from that of the EMTP in that the history sources are updated at the end of the solve code loop as opposed to the beginning. This permits the initial conditions to be specified in terms of Norton equivalent history terms as opposed to branch voltages and currents.

solve_code(

$t \leftarrow t_0, [J_h] \leftarrow [J_{h0}]$

while $t < t_f$,

build_rhs(

Algorithm 3.1 Transient solve code for linear networks (Continued . . .)

```
            compute([V_B], [J_j])
                [rhs] ← [A_hA][J_h] + [A_jA][J_j] − [G_AB][V_B]
        )
        solve (
            [V_A] ← backsub([G_AA], [rhs])
        )
        update_history_terms(
            [J_h] ← [U_h] · ([J_h] + 2[G_h][A_v][V]
        )
        t ← t + Δt
    end_while
)
```

Algorithm 3.1 Transient solve code for linear networks

3.5 Numerical example: Transient analysis

Consider the simple network shown in Fig. 3.4(a). Fig. 3.4(b) depicts the corresponding discrete-time network with inductances and capacitances represented as equivalent conductances shunted by Norton equivalent history sources.

Formulation of the network equations: With nodes and history sources defined as in Fig. 3.4(b) the discrete-time network equations become

$$\begin{bmatrix} g_{R_s} + g_{L_s} & -g_{L_s} \\ -g_{L_s} & g_{L_s} + g_C + g_{L_m} \end{bmatrix} \begin{bmatrix} v_1(t) \\ v_2(t) \end{bmatrix} = \begin{bmatrix} 0 & 0 & -1 \\ -1 & -1 & 1 \end{bmatrix} \begin{bmatrix} j_{h_1}(t) \\ j_{h_2}(t) \\ j_{h_3}(t) \end{bmatrix} - \begin{bmatrix} -g_{R_s} \\ 0 \end{bmatrix} [v_3(t)] \tag{3.32}$$

with history sources updated recursively as follows

$$\begin{bmatrix} j_{h_1}(t+\Delta t) \\ j_{h_2}(t+\Delta t) \\ j_{h_3}(t+\Delta t) \end{bmatrix} = [U_h] \left(\begin{bmatrix} j_{h_1}(t) \\ j_{h_2}(t) \\ j_{h_3}(t) \end{bmatrix} + 2[G_h] \begin{bmatrix} 0 & 1 & 0 \\ 0 & 1 & 0 \\ 1 & -1 & 0 \end{bmatrix} \begin{bmatrix} v_1(t) \\ v_2(t) \\ v_3(t) \end{bmatrix} \right) \tag{3.33}$$

$$[U_h] = \text{diag}([\,1 \quad -1 \quad 1\,])$$

$$[G_h] = \text{diag}([\,g_{L_m} \quad g_C \quad g_{L_s}\,])$$

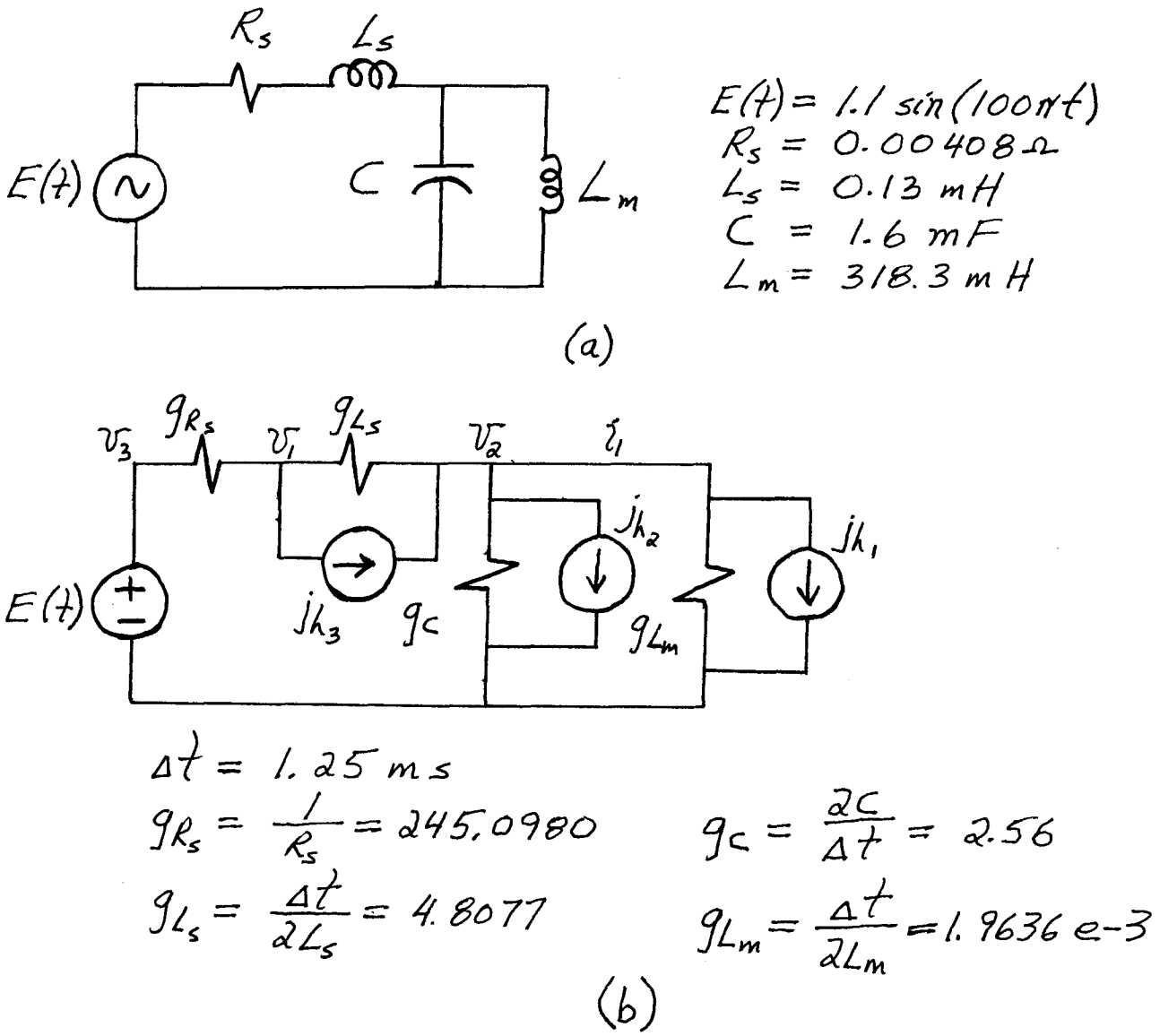

Figure 3.4 Simple linear, lumped element network (a) continuous-time network (b) discrete-time network

Triangularization: For the example at hand with $\Delta t = 1.25$ ms downward operations transform the augmented matrix $[\,G_{AA} \quad G_{AB} \quad A_{hA} \quad A_{jA}\,]$ to

$$\begin{bmatrix} 249.9057 & -4.8077 & -245.0980 & 0 & 0 & -1 \\ 0 & 7.2772 & -4.7152 & -1 & -1 & 0.9808 \end{bmatrix} \tag{3.34}$$

Solution method: Algorithm 3.1 with parameters $t_0 = 0$ and $t_f = 200$ ms is applied to the sample network. Fig. 3.5 represents the progression of the voltage and currrent waveforms

$v_2(t)$ and $i_1(t)$ from zero initial conditions. Note that even after ten cycles the solution has not yet settled to the periodic steady-state.

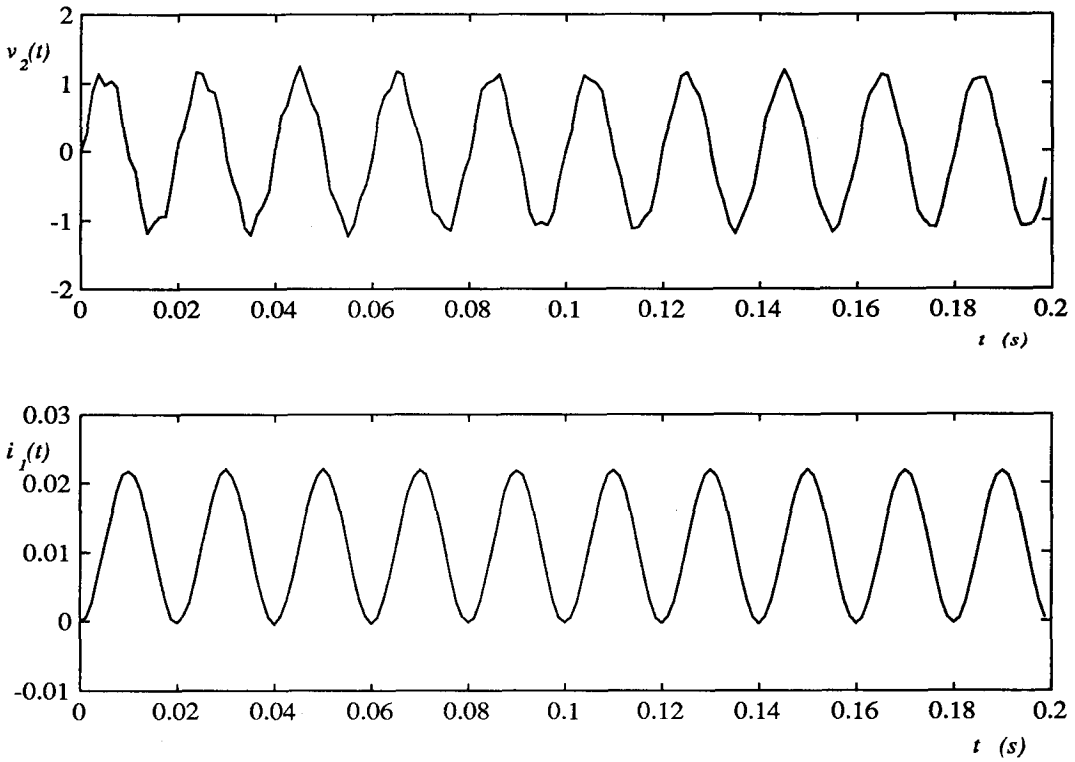

Figure 3.5 Solution of sample network from zero initial conditions

3.6 Steady-state initialization

The Norton equivalent history sources represent the state of a discrete-time network. Define

$$[F(J_{h0})] = [J_h(T; J_{h0})] \tag{3.35}$$

as the state at time-step T on the solution trajectory uniquely defined by $[J_{h0}]$ and computed by simulating the network over one period.

A linear network with sources of period T has a steady-state solution of period T. The discrete-time network has reached the periodic steady-state when

$$[F(J_{h0})] = [J_{h0}] \tag{3.36}$$

In general, (3.36) is not satisfied by an arbitrary choice of $[J_{h0}]$. We thus define the objective function

$$[H(J_{h0})] = [F(J_{h0})] - [J_{h0}] \tag{3.37}$$

for which the steady-state initial condition $[J^*_{h0}]$ yields the solution (zero of the objective function). The two-point boundary value problem can be solved by Newton-Raphson on (3.37). For a linear network only a single iteration is required for convergence to the steady-state initial conditions. We thus have

$$[J^*_{h0}] = [J_{h0}] - [DF(J_{h0}) - U]^{-1}[F(J_{h0}) - J_{h0}] \tag{3.38}$$

Since $[J_{h0}]$ is some arbitrary initial state, choose $[J_{h0}] = [0]$. The steady-state initialization problem thus reduces to solving the linear system

$$[DF(0) - U][J^*_{h0}] = -[F(0)] \tag{3.39}$$

3.7 Computing the Jacobian Matrix

The proposed approach requires the efficient and accurate computation of $[DF(J_{h0})]$. The *brute-force* approach involves perturbing each of the n states (Norton equivalent history sources) in turn and computing the Jacobian matrix by finite-differences. This approach is computationally inefficient and prone to numerical innaccuracy. The *sensitivity networks* approach computes the Jacobian matrix by solving n linear sensitivity networks that are intrinsically related to the discrete-time network.

The sensitivity networks approach is motivated as follows. $[F(J_{h0})]$ is computed by solving the discrete-time network (3.25) from $t = 0$ to $t = T$ with history sources updated recursively as in (3.27). Analogously, $[DF(J_{h0})] = \left[\frac{\partial J_h(T;J_{h0})}{\partial J_{h0}}\right]$ is computed by solving

$$[G_{AA}]\left[\frac{\partial V_A(t)}{\partial J_{h0}}\right] = [A_{hA}]\left[\frac{\partial J_h(t)}{\partial J_{h0}}\right] \tag{3.40}$$

from $t = 0$ to $t = T$ with history sources updated recursively as follows

$$\begin{aligned} \left[\frac{\partial J_h(0)}{\partial J_{h0}}\right] &= [U], \qquad t = 0 \\ \left[\frac{\partial J_h(t+\Delta t)}{\partial J_{h0}}\right] &= [U_h]\left(\left[\frac{\partial J_h(t)}{\partial J_{h0}}\right] + 2[G_h][A_{vA}]\left[\frac{\partial V_A(t)}{\partial J_{h0}}\right]\right), \qquad t \geq \Delta t \end{aligned} \tag{3.41}$$

Note that the solution of the n sensitivity networks is simply the ***matrix-valued solution of the discrete-time network with the time-dependent sources zeroed.*** The sensitivity networks approach is a computationally efficient and accurate method for computing the Jacobian matrix.

3.8 Steady-state initialization algorithm

Algorithm 3.2 depicts the proposed steady-state initialization algorithm in pseudocode form. Note that the algorithm is implemented within the framework of the solve code presented in Section 3.4. (The additional code over and above that in Algorithm 3.1 is marked with a double-dagger symbol (‡)).

steady-state_init (

$t \leftarrow 0, t_f \leftarrow T, [J_h] \leftarrow [0], [DJ_h] \leftarrow [U]$ ‡

while $t < t_f$,

build_rhs(

compute($[V_B], [J_j]$)

$[rhs] \leftarrow [A_{hA}][J_h] + [A_{jA}][J_j] - [G_{AB}][V_B]$

$[Drhs] \leftarrow [A_{hA}][DJ_h]$ ‡

)

solve (

$[V_A] \leftarrow$ **backsub**($[G_{AA}], [rhs]$)

$[DV_A] \leftarrow$ **backsub** ($[G_{AA}], [Drhs]$) ‡

)

Algorithm 3.2 Time-domain steady-state initialization algorithm for linear networks (Continued . . .)

update_history_terms (

$[J_h] \leftarrow [U_h] \cdot ([J_h] + 2[G_h][A_v][V])$

$[DJ_h] \leftarrow [U_h] \cdot ([DJ_h] + 2[G_h][A_{vA}][DV_A])$ ‡

)

$t \leftarrow t + \Delta t$

end_while

newton_raphson_iteration(

$[J_{h0}] \leftarrow [DJ_h - U]^{-1} \cdot [-J_h]$ ‡

)

)

Algorithm 3.2 Time-domain steady-state initialization algorithm for linear networks

3.9 Numerical example: Steady-state initialization

The algorithm can be used by itself to initialize the discrete-time network for transient studies. A time-domain representation of the steady-state waveforms requires transient simulation over one period from the steady-state initial conditions. For the example of Section 3.5, $[J_h(0)]$ is computed by both the existing frequency domain and the proposed time-domain steady-state initialization algorithm. $[J_h(T)]$ represents the state of the network after an additional period of simulation. Fig. 3.6 represents the solution from the steady-state initial conditions computed by the time-domain method.

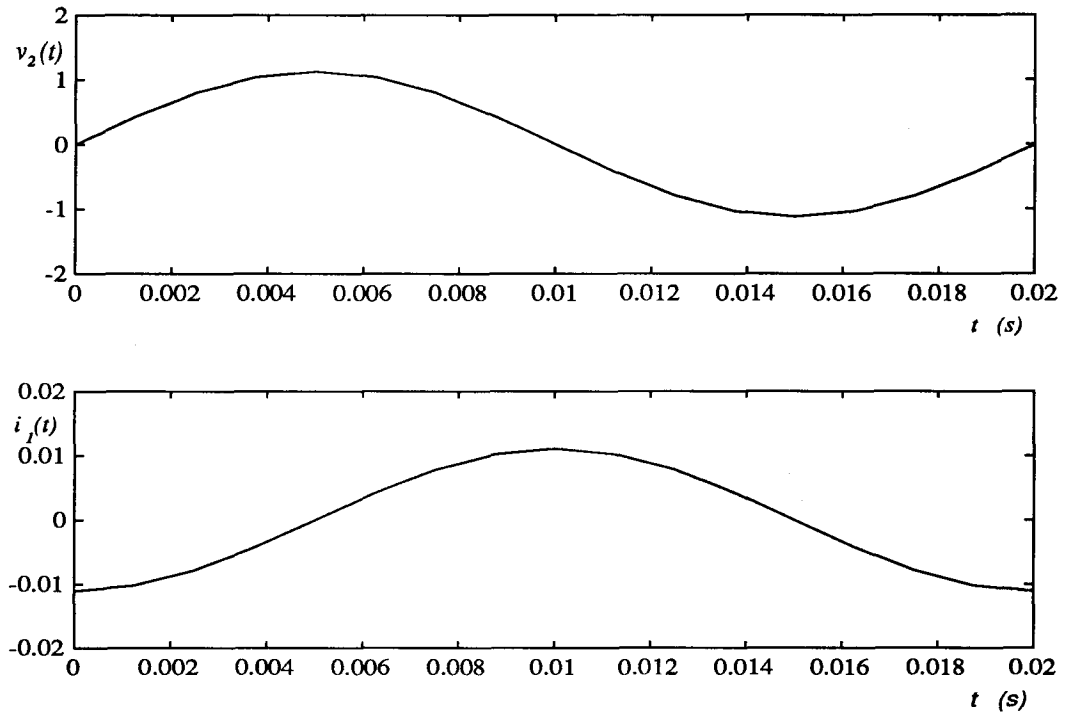

Figure 3.6 Solution of sample network from steady-state initial conditions

With the existing frequency-domain initialization

$$[J_h(0)] = \begin{bmatrix} -0.011231 \\ 0.55838 \\ 0.55327 \end{bmatrix} \qquad [J_h(T)] = \begin{bmatrix} -0.011242 \\ 0.56912 \\ 0.58210 \end{bmatrix} \tag{3.42}$$
$$\|J_h(T) - [J_h(0)]\|_2 = 3.0772e-2$$

Note that the computed initial condition does not satisfy the periodicity constraint of the discrete-time network. For the proposed time-domain method

$$[J_h(0)] = \begin{bmatrix} -0.011083 \\ 0.57793 \\ 0.55327 \end{bmatrix} \qquad [J_h(T)] = \begin{bmatrix} -0.011083 \\ 0.57793 \\ 0.55327 \end{bmatrix} \tag{3.43}$$
$$\|J_h(T) - J_h(0)\|_2 = 0$$

the periodicity constraint is in fact satisfied.

It is convenient, particularly for power flow calculations, to express the steady-state solution in phasor form. For example, given the sequence

$$[v] = [\, v(0) \quad v(\Delta t) \quad v(2\Delta t) \quad \cdots \quad v((N-1)\Delta t)\,] \tag{3.44}$$

where $N = \frac{T}{\Delta t}$, the phasor representation (in rectangular coordinates) is

$$\mathbf{V} = \sum_{n=0}^{N-1} v(n\Delta t) \exp\left(-\mathrm{j}n\frac{2\pi}{N}\right) \tag{3.45}$$

for which the polar form is expressed as$\mathbf{V}\angle\phi$, $V = \left(\frac{\sqrt{2}}{N}\right)\mathrm{mag}(\mathbf{V})$, $\phi = \arg(\mathbf{V})$.

Table 3.2 itemizes the phasor representation of the continuous-time solution and the discrete-time network solution of the example at hand. The continuous-time solution is the exact

	Continuous-time network	Discrete-time network ($\Delta t = 1.25$ ms)
$\mathbf{V_1}$ (V rms)	$0.7778 \angle -90.1175^o$	$0.7778 \angle -90.1245^o$
$\mathbf{V_2}$ (V rms)	$0.7938 \angle -90.1175^o$	$0.7956 \angle -90.1245^o$
$\mathbf{V_3}$ (V rms)	$0.7778 \angle -90.0^o$	$0.7778 \angle -90.0^o$
$\mathbf{I_1}$ (A rms)	$0.0079 \angle 179.8825^o$	$0.0076 \angle 179.8755^o$

Table 3.2 Phasor representation of steady-state solution for continuous-time and discrete-time network

solution of the network. The discrete-time network solution shows some discrepancy from the exact solution as the discrete-time network only approximates the continuous-time network.

3.10 Application: Capacitive and Inductive Coupling into fence[5]

A fence runs parallel to a three-phase power line as depicted in Fig. 3.7. By simply treating the fence as an additional conductor the following [C], [R] and [L] symmetrical matrices for a nominal multiphase $\pi-$circuit are obtained:

$$\frac{1}{2}[C] = \begin{bmatrix} 3.78545 & \cdot & \cdot & \cdot \\ 0.81330 & 3.65440 & \cdot & \cdot \\ 0.81520 & 0.41745 & 3.64995 & \cdot \\ 0.08440 & 0.13790 & 0.05945 & 3.48635 \end{bmatrix} \text{ nF} \tag{3.46}$$

$$[R] = \begin{bmatrix} 0.8108 & \cdot & \cdot & \cdot \\ 0.1148 & 0.8108 & \cdot & \cdot \\ 0.1148 & 0.1148 & 0.8108 & \cdot \\ 0.1162 & 0.1162 & 0.1162 & 3.7214 \end{bmatrix} \Omega \tag{3.47}$$

[5] This case is provided by Dr. H.W. Dommel

$$[L] = \begin{bmatrix} 5.2304 & \cdot & \cdot & \cdot \\ 2.2627 & 5.2304 & \cdot & \cdot \\ 2.2627 & 1.9852 & 5.2304 & \cdot \\ 1.6807 & 1.7495 & 1.6149 & 5.2802 \end{bmatrix} \text{mH} \tag{3.48}$$

Power line conductors:

R = 0.348 ohms/km
Xa = 0.386 ohms/km (60 Hz)
(reactance at 1 foot spacing)

diameter = 12.7 mm
frequency = 60 Hz

Fence:
R = 1.802 ohms/km
solid conductor (nonmagnetic)
diameter = 4.064 mm
length = 2 km

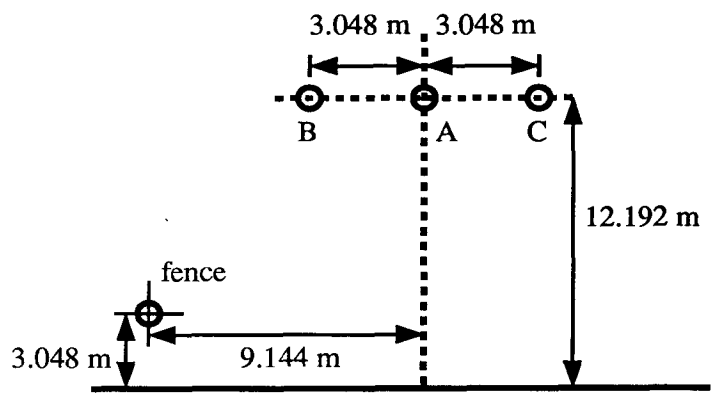

Figure 3.7 Electrostatic and electromagnetic coupling from power line to fence

Capacitive coupling: Under the assumption that the fence is insulated and is nowhere grounded, the steady-state voltage on the fence due to capacitive coupling is calculated for two cases of power line energization:

1. symmetrical operation at 345 kV rms line-to-line
2. phase A grounded because of a nearby single line-to-ground fault, phases B and C at normal voltage.

Inductive coupling: Under the assumption that the 2 km long fence is grounded at one end and open at the other end (insulated in between), the steady-state voltage at the open end of the fence is calculated for two cases:

1. symmetrical operation with I_{phase} = 1 kA rms
2. I_A = 10 kA, $I_B = I_C$ =0 because of a nearby single line-to-ground fault.

The preceeding cases were solved by the existing EMTP frequency-domain based steady-state initialization algorithm as well as the proposed time-domain based steady-state initialization algorithm (for two vales of Δt). Table 3.3 summarizes the computed values for the induced voltage on the fence.

CASE	EMTP frequency-domain method	Proposed time-domain method ($\Delta t = 0.52083$ ms)	Proposed time-domain method ($\Delta t = 2.0833$ ms)
Capacitive coupling (case 1)	$3.9664 \angle -101.87^o$ kV	$3.9663 \angle -101.87^o$ kV	$3.9663 \angle -101.87^o$ kV
Capacitive coupling (case 2)	$6.8446 \angle -145.45^o$ kV	$6.8444 \angle -145.45^o$ kV	$6.8443 \angle -145.45^o$ kV
Inductive coupling (case 1)	$42.782 \angle -179.87^o$ V	$42.907 \angle -179.85^o$ V	$45.112 \angle -179.85^o$ V
Inductive coupling (case 2)	$6.4416 \angle -100.39^o$ kV	$6.4618 \angle -100.36^0$ kV	$6.7834 \angle -99.86^o$ kV

Table 3.3 Computed steady-state values of the induced voltage on the fence.

Chapter 4 Steady-state initialization: Nonlinear Networks

4.1 Introduction

A general method for the formulation, solution, and steady-state initialization of nonlinear lumped element networks is presented in this chapter. The proposed formulation is limited to current-controlled nonlinear elements[6] although typically voltage-controlled nonlinear elements[7] can be expressed as current-controlled nonlinear elements.

4.2 Representation of nonlinear elements

A continuously differentiable scalar function of current is expressed in linearized form as follows

$$f(i(t)) = m(t) \cdot i(t) + b(t) \tag{4.49}$$

where

$$\begin{aligned} m(t) &= \left(\frac{\partial f}{\partial i}\right)_{i(t)} \\ b(t) &= f(i(t)) - m(t) \cdot i(t) \end{aligned} \tag{4.50}$$

Note that this permits a convenient representation of piecewise-linear functions. Current-controlled nonlinearities are thus represented as follows:

Nonlinear resistance: A nonlinear resistance with branch characteristic $v(t) = f(i(t))$ is expressed as

$$v(t) = r(t) \cdot i(t) + e_k(t) \tag{4.51}$$

where $r(t) = m(t)$ represents a time-varying resistance and $e_k(t)$ represents a time-varying 'knee' voltage.

---

6 Current-controlled nonlinear elements are those in which there is a one-to-one correspondence between the current and voltage (nonlinear resistance) or between the current and the state (nonlinear inductance).

7 Nonlinear conductances and capacitances.

Nonlinear inductance: Upon the application of the trapezoidal rule of integration, a nonlinear inductance with branch characteristic $v(t) = \frac{d\lambda(t)}{dt}$, $\lambda(t) = f(i(t))$ is expressed as

$$\begin{aligned} v(t) &= r(t) \cdot i(t) + e_k(t) + e_h(t) \\ e_h(t) &= -v(t - \Delta t) - r(t - \Delta t) \cdot i(t - \Delta t) - e_k(t - \Delta t) \end{aligned} \tag{4.52}$$

where $r(t) = \frac{2}{\Delta t} m(t)$ represents a time-varying resistance and $e_k(t) = \frac{2}{\Delta t} b(t)$ represents a time-varing 'knee' voltage. Note that the discretized nonlinear inductance characteristic is simply a scaled and shifted version of the nonlinear resistance characteristic.

A network may contain any number of nonlinear resistances and inductances. We thus introduce the notation

$$[V(I(t))] = [R(t)][I(t)] + [E_k(t)] + \begin{bmatrix} E_h(t) \\ 0 \end{bmatrix} \tag{4.53}$$

where $[V(I(t))]$ and $[I(t)]$ represent nonlinear branch voltages and currents respectively, and

$$\begin{aligned} [R(t)] &= \text{diag}([\cdots \quad \tfrac{\Delta t}{2} m_j(t) \quad \cdots \quad m_j(t) \quad \cdots]) \\ [E_k(t)] &= [\cdots \quad \tfrac{\Delta t}{2} b_j(t) \quad \cdots \quad b_j(t) \quad \cdots]^T \\ [E_h(t)] &= [\cdots \quad e_{h_j}(t) \quad \cdots]^T \end{aligned} \tag{4.54}$$

represent a diagonal matrix of time-varying resistances, a vector of time-varying 'knee' voltages, and a vector of Thevenin equivalent history sources respectively. Thevenin equivalent history sources are updated as follows

$$[E_h(t + \Delta t)] = [\,-U \quad 0\,] \cdot ([A_n][V(t)] + [R(t)][I(t)] + E_k(t)) \tag{4.55}$$

where $[\,-U \quad 0\,]$ indicates that the computation proceeds only for branches with nonlinear inductances. $[A_n] = [C]$ is an incidence matrix mapping the node voltages onto the nonlinear branch voltages, where $[C]$ is defined in the following section.

4.3 Formulation of the network equations

Current-controlled nonlinearities are incorporated into the nodal formulation of Section 3.2 by a compensation method as follows [9]

$$\begin{bmatrix} G & B \\ C & 0 \end{bmatrix} \begin{bmatrix} V(t) \\ I(t) \end{bmatrix} = \begin{bmatrix} J(t) \\ V(I(t)) \end{bmatrix} \tag{4.56}$$

where

$$[C] = \left[c_{(ij)}\right], \qquad c_{(ij)} = \left\{ \begin{array}{l} +1, \text{ positive incidence} \\ -1, \text{ negative incidence} \\ 0, \text{ no incidence} \end{array} \right\} \tag{4.57}$$

maps the linear node voltages onto the nonlinear branch voltages and

$$[B] = [C]^T \tag{4.58}$$

maps the compensation currents onto the linear part of the network.

As in Section 3.2 the vector of node voltages is partitioned into unknown and known node voltages. Partitioning the matrices accordingly, the unknown node voltages and the compensation currents are given by

$$\begin{bmatrix} G_{AA} & B_A \\ C_A & 0 \end{bmatrix} \begin{bmatrix} V_A(t) \\ I(t) \end{bmatrix} = \begin{bmatrix} J_A(t) \\ V(I(t)) \end{bmatrix} - \begin{bmatrix} G_{AB} \\ C_B \end{bmatrix} [V_B(t)] \tag{4.59}$$

The Norton equivalent history sources associated with linear capacitances and inductances are updated recursively as in Section 3.2. Thevenin equivalent history sources associated with nonlinear inductances are updated as in (4.55).

4.4 Network equivalents

A straightforward application of an iterative method for solving the nonlinear system of equations at each time-step is prohibitively computationally intensive for large scale networks. If, however, $[V_A(t)]$ is 'decoupled' from $[I(t)]$ by forming a network equivalent of the linear part of the network, an iterative solution method can be applied to the subset of nonlinear equations. The linear part of the network can then be solved by back substitution.

A multidimensional Thevenin equivalent with respect to the nonlinear branches is synthesized prior to the first time-step as follows

$$\begin{array}{c} \begin{bmatrix} G_{AA} & G_{AB} & B_A & A_{hA} & A_{jA} \\ 0 & C_B & -T & A'_{hA} & A'_{jA} \end{bmatrix} \leftarrow \\ \begin{bmatrix} P & 0 \\ -K & U \end{bmatrix} \begin{bmatrix} G_{AA} & G_{AB} & B_A & A_{hA} & A_{jA} \\ C_A & C_B & 0 & 0 & 0 \end{bmatrix} \end{array} \tag{4.60}$$

where $[P]$ represents the permutation matrix that upper triangularizes $[G_{AA}]$ and $[K] = [C_A][G_{AA}]^{-1}$. In practice, the LHS of (4.60) is synthesized by downward operations that

leaves $[G_{AA}]$ upper triangularized and $[C_A] = [0]$. We now have network equations of the form

$$\begin{bmatrix} G_{AA} & B_A \\ 0 & -T \end{bmatrix} \begin{bmatrix} V_A(t) \\ I(t) \end{bmatrix} = \begin{bmatrix} J_A(t) \\ V(I(t)) + J_A^{'}(t) \end{bmatrix} - \begin{bmatrix} G_{AB} \\ C_B \end{bmatrix} [V_B(t)] \tag{4.61}$$

where $\left[J_A^{'}(t)\right] = \left[A_{hA}^{'}\right][J_h(t)] + \left[A_{jA}^{'}\right][J_j(t)]$. Note that the nonlinear elements are now coupled to a Thevenin equivalent of the linear part of the network, namely

$$\left[V_o^{'}(t)\right] - [T][I(t)] = [V(I(t))] \tag{4.62}$$

where $\left[V_o^{'}(t)\right] = [C_B][V_B(t)] - \left[J_A^{'}(t)\right]$ represents the open circuit voltage.

4.5 Solution method

The proposed solution method is based on a piecewise-linear fixed-point iteration that yields Newton-Raphson-like convergence. Expressing $[V(I(t))]$ as in (4.53) yields

$$[V_o(t)] - [T][I(t)] = [R(t)][I(t)] + [E_k(t)] \tag{4.63}$$

where $[V_o(t)] = \left[V_o^{'}(t)\right] - \begin{bmatrix} E_h(t) \\ 0 \end{bmatrix}$. Casting (4.63) in the form of a fixed-point iteration $[I(t)] = [F'(I(t))]$, we define

$$\left[F^{'}(I(t))\right] = [T + R(t)]^{-1}[V_o(t) - E_k(t)] \tag{4.64}$$

where the iteration

$$\left[I^{(i+1)}(t)\right] = \left[F^{'}\left(I^{(i)}(t)\right)\right] \tag{4.65}$$

converges to a unique fixed-point $[I^*(t)]$ such that $[I^*(t)] = [F'(I^*(t))]$. $[I^*(t)]$ represents the intersection of the linear network Thevenin equivalent and the nonlinear branch characteristic as depicted in Fig 4.8(a) for the scalar case. The fixed-point iteration is 'seeded' with the solution from the previous time-step, namely $[I^{(0)}(t)] = [I^*(t - \Delta t)]$.

The nonlinear problem has been reduced to finding the segment of the piecewise-linear branch characteristics that uniquely solves the network at each time-step. We now discuss the convergence and divergence properties of the fixed-point iteration. For $[I^{(0)}(t)]$

sufficiently close to $[I^*(t)]$ the iterates will converge quadratically upon the solution. Figures 4.8(b) and (c) depict typical convergence behaviour in the scalar case. Divergence is characterized by a *cyclic path* among the piecewise-linear segments [8]. Fig. 4.8(d) depicts a typical divergence mechanism in the scalar case. Note the cyclic path between segments 1 and 3 — the iterates will never converge to the solution which lies on segment 2. An anologous divergence mechanism occurs in the multidimensional case among piecewise-linear hyperplanes (as opposed to segments). Cyclic paths must be detected and corrective action taken.

The proposed fixed-point iteration algorithm for the solution of the currents in the nonlinear branches is based on the following paradigms:

1. Convergence to the solution is achieved if two *successive* iterates visit the same piecewise-linear segment (hyperplane).
2. Divergence is detected if two *non-successive* iterates visit the same piecewise-linear segment (hyperplane). Corrective action consists of 're-seeding' the iteration on a piecewise-linear segment (hyperplane) that lies between the two iterated segments (hyperplanes).

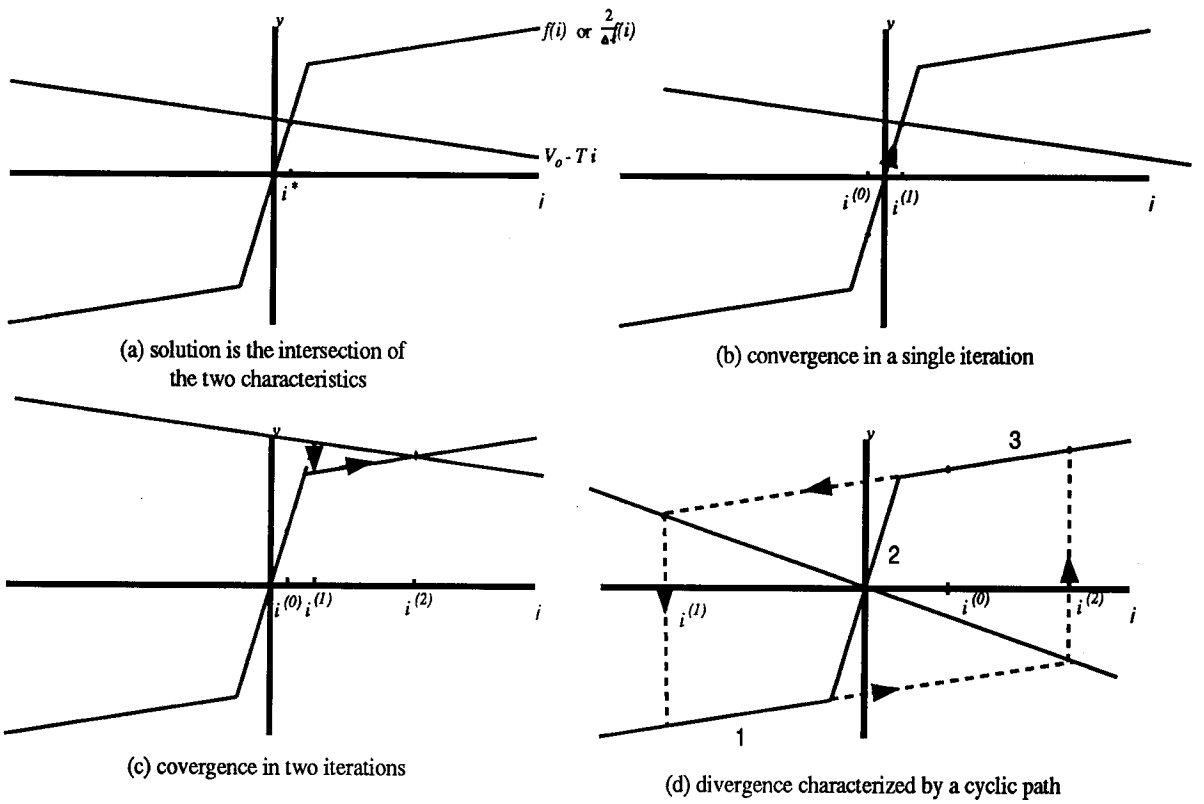

Figure 4.8 Convergence and divergence behaviour of fixed-point iteration

The unknown node voltages associated with the linear part of the network are found by back-substitution. Such a scheme can be readily incorporated into the algorithm presented in Section 3.3 as presented in Algorithm 4.3.

nonlinear_solve_code (

$t \leftarrow t_0, [J_h] \leftarrow [J_{h0}], [E_h] \leftarrow [E_{h0}]$

while $t < t_f$,

build_rhs(

compute($[V_B], [J_j]$)

$[rhs] \leftarrow [A_{hA}][J_h] + [A_{jA}][J_j] - [G_{AB}][V_B]$

)

Algorithm 4.3 Transient solve code for nonlinear networks (Continued . . .)

solve_compensation_currents(

$[J'_A] \leftarrow [A'_{hA}][J_h] + [A'_{jA}][J_j]$

$[V_o] = [C_B][V_B] - [J'_A] - \begin{bmatrix} E_h \\ 0 \end{bmatrix}$

$[I, R, E_k] \leftarrow$**fixed_point_iteration**(I, T, V_o)

$[rhs] \leftarrow [rhs] - [B_A][I]$

)

solve (

$[V_A] \leftarrow$ **backsub**$([G_{AA}], [rhs])$

)

update_history_terms (

$[J_h] \leftarrow [U_h] \cdot ([J_h] + 2[G_h][A_v][V])$

$[E_h] \leftarrow [-U \quad 0] \; ([A_n][V] + [R][I] + [E_k])$

)

$t \leftarrow t + \Delta t$

end_while

)

Algorithm 4.3 Transient solve code for nonlinear networks

4.6 Numerical example: Transient analysis

Consider the simple network of Section 3.5 with the linear inductance L_m replaced by a piecewise-linear inductance L_n defined by

$$L_n = \left\{ \begin{array}{l} 318.3 \text{ mH}, \; i < -i_{sat} \\ 1.591 \text{ mH}, \; -i_{sat} \leq i_1 < i_{sat} \\ 318.3 \text{ mH}, \; i_1 \geq i_{sat} \end{array} \right\} \tag{4.66}$$

where $i_{sat} = 0.01$ A.

Formulation of the network equations: With nodes defined as before and the Norton equivalent representation of L_m with history source j_{h_1} replaced by the Thevenin equivalent

representation of L_n with history source e_{h_1}, the discrete-time network equations become

$$\begin{bmatrix} g_{R_s}+g_{L_s} & -g_{L_s} & 0 \\ -g_{L_s} & g_{L_s}+g_C & 1 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} v_1(t) \\ v_2(t) \\ i_{1(t)} \end{bmatrix} = \begin{bmatrix} J_A(t) \\ v(i_1(t)) \end{bmatrix} - \begin{bmatrix} -g_{R_s} \\ 0 \\ 0 \end{bmatrix} [v_3(t)] \tag{4.67}$$

where

$$\begin{aligned} [J_A(t)] &= \begin{bmatrix} 0 & -1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} j_{h_2}(t) \\ j_{h_3}(t) \end{bmatrix} \\ v(i_1(t)) &= r(t) \cdot i_1(t) + e_k(t) + e_{h1}(t) \\ e_{h1}(t) &= -(v_2(t-\Delta t) + r(t-\Delta t) \cdot i_1(t-\Delta t) + e_k(t-\Delta t)) \end{aligned} \tag{4.68}$$

and Norton equivalent history sources updated recursively.

Network equivalents For the example at hand with $\Delta t = 1.25$ ms downward operations transform the augmented matrix (4.60) to

$$\begin{bmatrix} 249.9057 & -4.8077 & -245.0980 & 0 & 0 & -1 \\ 0 & 7.2752 & -4.7152 & 1 & -1 & 0.9808 \\ 0 & 0 & 0.6481 & -0.1375 & 0.1375 & -0.1348 \end{bmatrix} \tag{4.69}$$

Solution method: Fig. 4.9 represents the progression of the voltage and currrent waveforms $v_2(t)$ and $i_1(t)$ from zero initial conditions. Note that even after ten cycles the solution has not yet settled to the periodic steady-state.

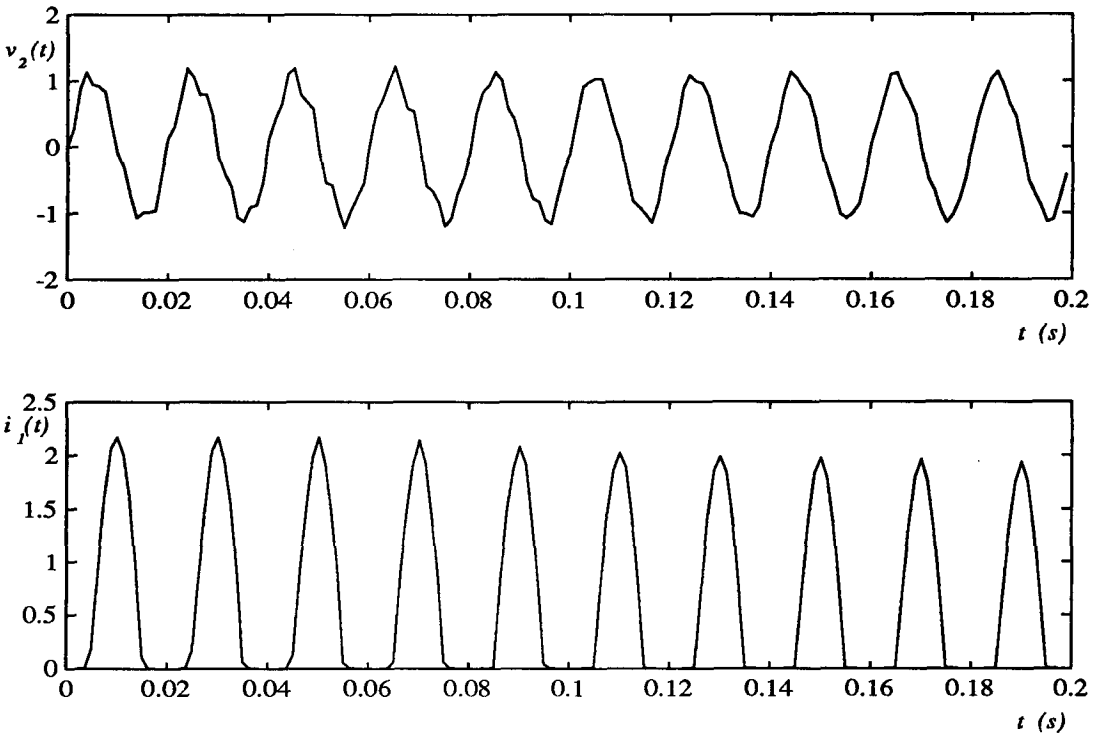

Figure 4.9 Solution of sample network from zero initial conditions

4.7 Steady-state initialization

The Norton and Thevenin equivalent history sources represent the state of a discrete-time network with current-controlled nonlinearities defined by$[h(t)] = \begin{bmatrix} J_h(t) \\ E_h(t) \end{bmatrix}$. Analogous to the development of Section 3.6, define

$$[F(h_0)] = [J_h(T; h_0)] \tag{4.70}$$

as the state at time-step T on the solution trajectory, which is uniquely defined by $[h_0]$ and computed by simulating the network over one period.

A nonlinear network with sources of period T may exhibit periodic, subharmonic, quasiperiodic, or even chaotic solutions [6]. The periodic steady state is, however, of particular interest. The discrete-time network has reached the periodic steady state when

$$[F(h_0)] = [h_0] \tag{4.71}$$

is solved uniquely by $[h_0^*]$, the steady-state initial conditions.

The two-point boundary value problem is solved by the Newton-Raphson iteration

$$\left[h_0^{(i+1)}\right] = \left[h_0^{(i)}\right] - \left[DF\left(h_0^{(i)}\right) - U\right]^{-1}\left[F\left(h_0^{(i)}\right) - U\right] \tag{4.72}$$

This approach yields *quadratic* convergence to $[h_0^*]$ at the additional expense of calculating the Jacobian matrix $[DF(h_0^{(i)})]$ at each iteration.

4.8 Computing the Jacobian Matrix

The proposed approach requires the efficient and accurate computation of the Jacobian matrix

$$[DF(h_0)] = \begin{bmatrix} \frac{\partial F(h_0)}{\partial J_{h0}} \\ \frac{\partial F(h_0)}{\partial E_{h0}} \end{bmatrix} \tag{4.73}$$

Along the same lines as Section 3.7, the Jacobian matrix is computed by the solution of n linear, time-varying *sensitivity networks* related to the *variational network*[8]

$$\begin{bmatrix} G_{AA} & B_A \\ 0 & -T - R(t) \end{bmatrix} \begin{bmatrix} \frac{\partial V_A(t)}{\partial h_0} \\ \frac{\partial I(t)}{\partial h_0} \end{bmatrix} = \begin{bmatrix} A_{hA}\frac{\partial J_h(t)}{\partial h_0} \\ \frac{\partial E(t)}{\partial h_0} + A'_{hA}\frac{\partial J_h(t)}{\partial h_0} \end{bmatrix} \tag{4.74}$$

from $t = 0$ to $t = T$ with history sources updated as follows

$$\begin{bmatrix} \frac{\partial J_h(0)}{\partial h_0} \\ \frac{\partial E_h(0)}{\partial h_0} \end{bmatrix} = \begin{bmatrix} U_1 & 0 \\ 0 & U_2 \end{bmatrix} \qquad t = 0 \tag{4.75}$$

$$\left[\frac{\partial J_h(t+\Delta t)}{\partial h_0}\right] = [U_h]\left(\left[\frac{\partial J_h(t)}{\partial J_h}\right] + 2[G_h][A_{vA}]\left[\frac{\partial V_A(t)}{\partial h_0}\right]\right) \tag{4.76}$$

$$\left[\frac{\partial E_h(t+\Delta t)}{\partial h_0}\right] = [-U \quad 0]\left([A_{nA}]\left[\frac{\partial V_A(t)}{\partial h_0}\right] + [R(t)]\left[\frac{\partial I(t)}{\partial h_0}\right]\right) \tag{4.77}$$

Thus the solution of the sensitivity networks is simply the *matrix-valued solution of the variational network with independent sources and 'knee' voltage sources zeroed.*

---

8 The variational networks follows by substituting (4.53) in (4.61).

4.9 Steady-state initialization algorithm

Algorithm 4.4 depicts the proposed steady-state initialization algorithm for networks with current-controlled nonlinearities in pseudocode form.

steady-state_init(

while $\left\| \begin{bmatrix} J_h \\ E_h \end{bmatrix} - \begin{bmatrix} J_{h0} \\ E_{h0} \end{bmatrix} \right\|_2 > \epsilon,$

$t \leftarrow 0, t_f \leftarrow T, [J_h] \leftarrow [J_{h0}], [E_h] \leftarrow [E_{h0}]$

$[DJ_h] \leftarrow [U_1 \quad 0],\ [DE_h] \leftarrow [0 \quad U_2]$

while $t < t_f$,

build_rhs(

compute$([V_B], [J_j])$

$[rhs] \leftarrow [A_{hA}][J_h] + [A_{jA}][J_j] - [G_{AB}][V_B]$

$[Drhs] \leftarrow [A_{hA}][DJ_h]$

)

solve_compensation_currents(

$[J'_A] \leftarrow [A'_{hA}][J_h] + [A'_{jA}][J_j]$

$[V_o] = [C_B][V_B] + [J'_A] - \begin{bmatrix} E_h \\ 0 \end{bmatrix}$

$[DV_o] \leftarrow -[A'_{hA}][DJh] - \begin{bmatrix} DE_h \\ 0 \end{bmatrix}$

$[I, R, E_k] \leftarrow$**fixed_point_iteration**(I, T, V_o)

$[DI] \leftarrow [T + R]^{-1} \cdot [DV_o]$

$[rhs] \leftarrow [rhs] - [B_A][I]$

$[Drhs] \leftarrow [Drhs] - [B_A][DI]$

)

solve (

$[V_A] \leftarrow$ **backsub**$([G_{AA}], [rhs])$

$[DV_A] \leftarrow$ **backsub** $([G_{AA}], [Drhs])$

)

Algorithm 4.4 Time-domain steady-state initialization algorithm for nonlinear networks (Continued . . .)

update_history_terms (

$$[J_h] \leftarrow [U_h] \cdot ([J_h] + 2[G_h][A_v][V])$$

$$[DJ_h] \leftarrow [U_h] \cdot ([DJ_h] + 2[G_h][A_{vA}][DV_A])$$

$$[E_h] \leftarrow [-U \quad 0] \; ([A_n][V] + [R][I] + [E_k])$$

$$[DE_h] \leftarrow [-U \quad 0] \; ([A_{nA}][DV_A] + [R][DI])$$

)

$t \leftarrow t + \Delta t$

end_while

newton_raphson_iteration(

$$\begin{bmatrix} J_{h0} \\ E_{h0} \end{bmatrix} \leftarrow \begin{bmatrix} J_{h0} \\ E_{h0} \end{bmatrix} - \left(\begin{bmatrix} DJ_h \\ DE_h \end{bmatrix} - \begin{bmatrix} U_1 & 0 \\ 0 & U_2 \end{bmatrix} \right)^{-1} \left(\begin{bmatrix} J_h \\ E_h \end{bmatrix} - \begin{bmatrix} J_{h0} \\ E_{h0} \end{bmatrix} \right)$$

)

end_while

)

Algorithm 4.4 Time-domain steady-state initialization algorithm for nonlinear networks

4.10 Numerical example: Steady-state initialization

For the example of Section 4.6, Fig. 4.10 and Fig. 4.11 show the progression of the steady-state initialization algorithm from zero initial conditions and from the steady-state initial conditions of the network linearized about $[I] = [0]$. Note that only 5 and 2 iterations, respectively, were required for convergence to the periodic steady-state.

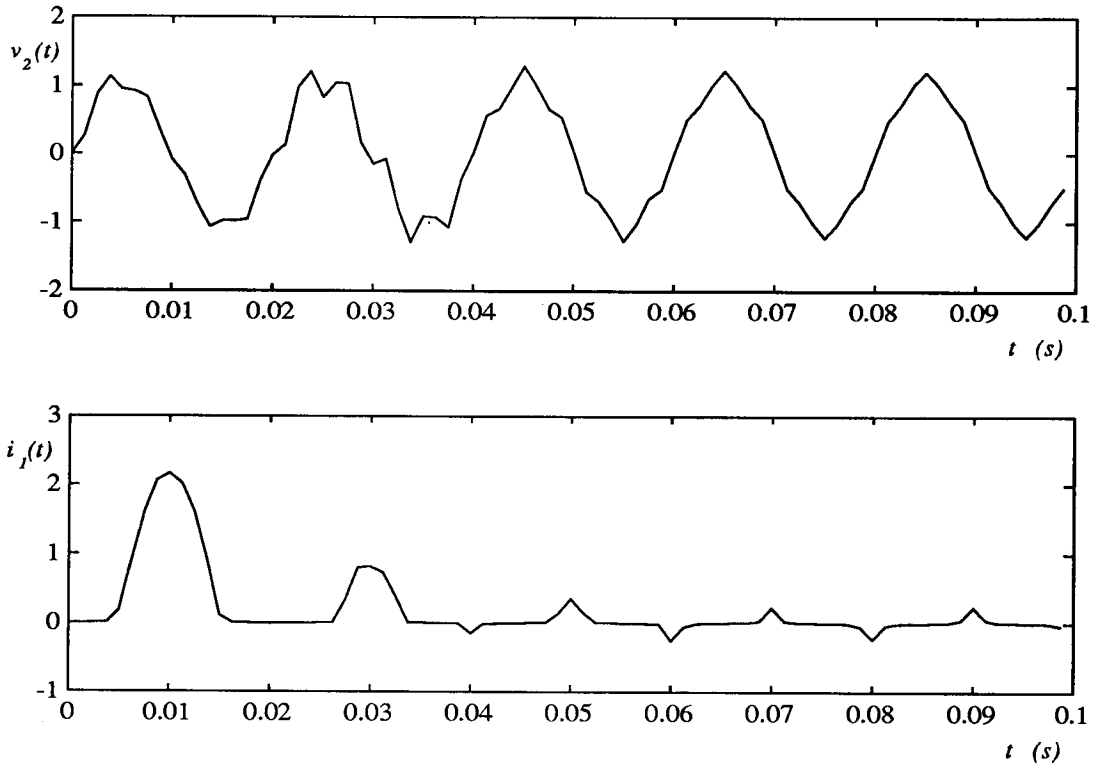

Figure 4.10 Progression of the steady-state initialization algorithm from zero initial conditions

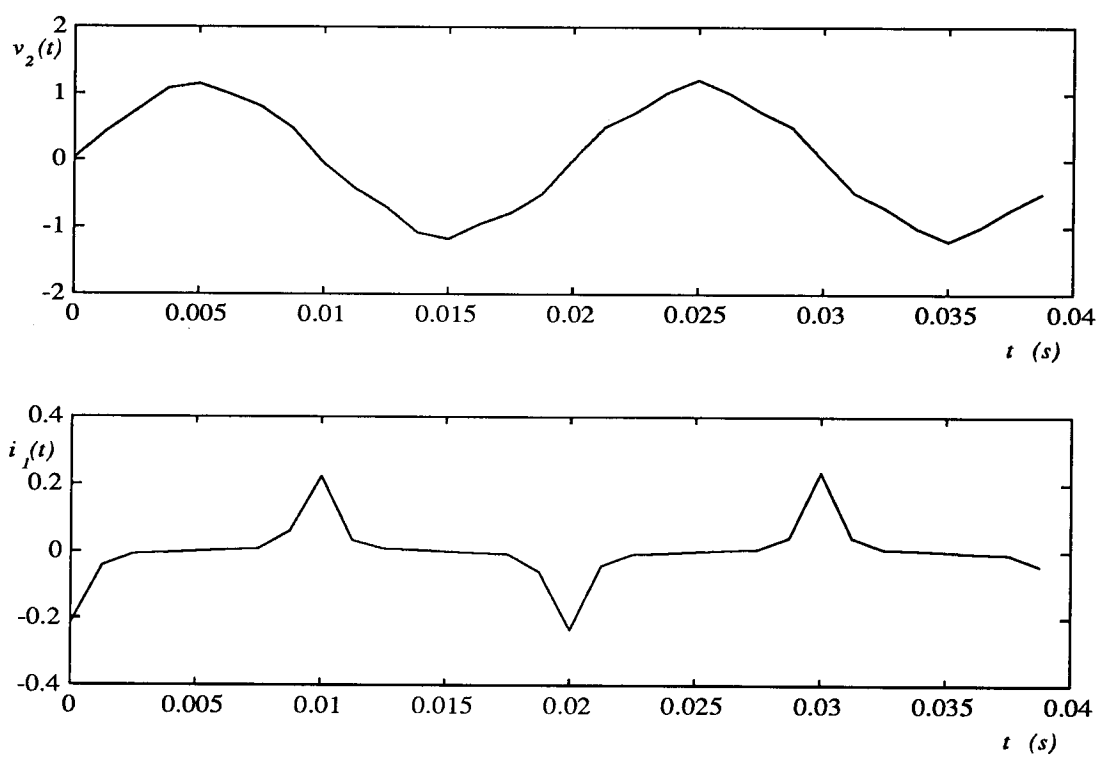

Figure 4.11 Progression of the steady-state initialization algorithm from linearized network initial conditions

4.11 Application: Harmonics due to Transformer Saturation

The example of Section 4.6 is a simplified representation of a nonlinear transformer terminating an open-circuited line with the nonlinear magnetizing characteristic defined piecewise-linear [10]. Harmonics due to transformer saturation are generally difficult to predict as they are highly dependent on the magnitude of the applied voltage.

Figure 4.12 shows the nonlinear inductor voltage and current waveforms for $\Delta t = 312.5\mu s$ obtained by the steady-state initialization algorithm. Table 4.4 depicts the corresponding phasor representation of the harmonics of interest. Note that the choice of Δt imposes a bandwidth constraint on the harmonic content of the steady-state waveform.

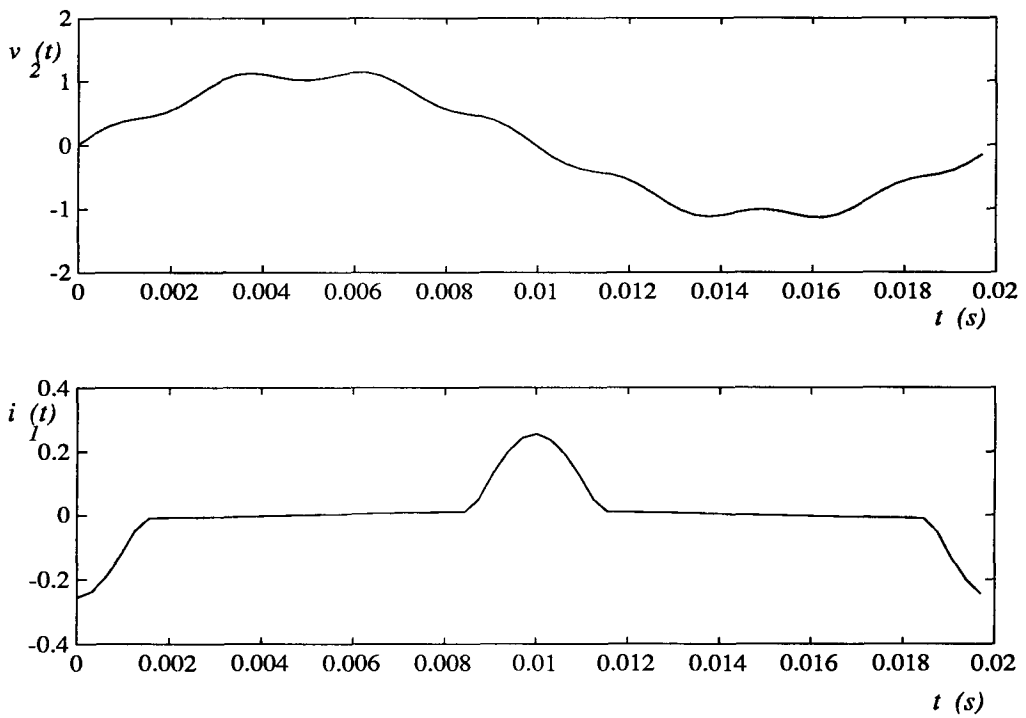

Figure 4.12 Steady-state waveform for the sample network ($\Delta t = 312.5\mu s$)

Harmonic number	Voltage Harmonic (V rms)	Current Harmonic (A rms)
1	$7.9129e-1\angle-90.10^o$	$6.8418e-2\angle-179.82^o$
3	$7.8842e-3\angle 88.36^o$	$5.1181e-2\angle-179.31^o$

Table 4.4 Phasor representation of voltage and current harmonics in nonlinear inductor ($\Delta t = 312.5\mu s$) (Continued . . .)

5	$1.6733e-2\angle 88.91^o$	$3.7372e-2\angle -178.68^o$
7	$6.9259e-2\angle -79.00^o$	$2.1392e-2\angle -177.57^o$
9	$3.3725e-3\angle -83.86^o$	$7.7646e-3\angle -174.51^o$
11	$2.0951e-4\angle 55.97^o$	$8.6580e-4\angle -34.26^o$
13	$6.3113e-4\angle 86.79^o$	$3.7703e-3\angle -3.3157^o$

Table 4.4 Phasor representation of voltage and current harmonics in nonlinear inductor ($\Delta t = 312.5\mu s$)

Chapter 5 Steady-state initialization: Variable-structure networks

5.1 Introduction

A general method for the formulation, solution, and steady-state initialization of variable structure networks[9] is presented in this chapter.

5.2 Representation of ideal-switch devices

An ideal-switch device is a two state switch governed by a set of Boolean equations of the terminal characteristics. The present discussion is restricted to an ideal diode. Define

$$s = \left\{ \begin{array}{l} 0, \text{ switch off} \\ 1, \text{ switch on} \end{array} \right\} \tag{5.78}$$

as the state of an ideal diode. Furthermore, define the boolean variables

$$s_v = \left\{ \begin{array}{l} 1, \ v \geq 0 \\ 0, \text{ otherwise} \end{array} \right\}, \quad s_i = \left\{ \begin{array}{l} 1, \ i > 0 \\ 0, \text{ otherwise} \end{array} \right\} \tag{5.79}$$

where v represents the anode-to-cathode voltage and i represents the anode-to-cathode current. The switch transitions, governed by changes in branch current and voltage are succinctly expressed in Fig. 5.13.

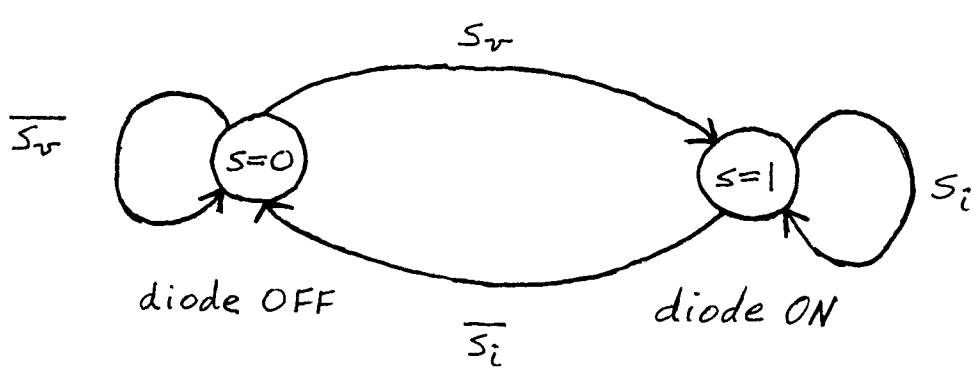

Figure 5.13 State transition diagram for an ideal diode

9 Variable structure networks are those containing ideal-switch devices, such as ideal diodes.

5.3 Formulation of the network equations

Ideal-switch devices are incorporated into the nodal formulation of the network equations by a compensation method as follows

$$\begin{bmatrix} G & B \\ S \cdot C & \overline{S} \end{bmatrix} \begin{bmatrix} V(t) \\ I(t) \end{bmatrix} = \begin{bmatrix} J(t) \\ 0 \end{bmatrix} \tag{5.80}$$

where $[I(t)]$ is a vector of switch branch currents and

$$\begin{aligned} [S] &= \text{diag}([\cdots \quad s_j \quad \cdots]) \\ [\overline{S}] &= \text{diag}([\cdots \quad \overline{s_j} \quad \cdots]) \end{aligned} \tag{5.81}$$

are diagonal matrices of the switch state (and its conjugate) introduced in the previous section. Again partitioning the vector of node voltages into unknown and known voltages yields

$$\begin{bmatrix} G_{AA} & B_A \\ S \cdot C_A & \overline{S} \end{bmatrix} \begin{bmatrix} V_A(t) \\ I(t) \end{bmatrix} = \begin{bmatrix} J_A(t) \\ 0 \end{bmatrix} - \begin{bmatrix} G_{AB} \\ S \cdot C_B \end{bmatrix} [V_B(t)] \tag{5.82}$$

For variable structure networks, the absolutely stable backward Euler rule of integration is employed with the time-step halved [3]. The Norton equivalent history sources are thus updated as follows

$$[J_h(t + \Delta t)] = [U_h][G_h][A_v][V(t)] + \frac{1}{2}[U + U_h][J_h(t)] \tag{5.83}$$

where $[U_h], [G_h]$ and $[A_v]$ are previously defined.

5.4 Network equivalents

Akin to Section 4.4, $[S \cdot C_A]$ is eliminated as follows[10]

$$\begin{aligned} &\begin{bmatrix} G_{AA} & G_{AB} & B_A & A_{hA} & A_{jA} \\ 0 & C_B^* & -T^* & A_{hA}^* & A_{jA}^* \end{bmatrix} \leftarrow \\ &\begin{bmatrix} P & 0 \\ -K^* & U \end{bmatrix} \begin{bmatrix} G_{AA} & G_{AB} & B_A & A_{hA} & A_{jA} \\ S \cdot C_A & S \cdot C_B & -\overline{S} & 0 & 0 \end{bmatrix} \end{aligned} \tag{5.84}$$

[10] In practice the network equivalent is synthesized by downward operations.

where $[P]$ represents the permutation matrix that upper triangularizes $[G_{AA}]$ and $[K^*] = [S \cdot C_A][G_{AA}]$. Note that the network equivalent is dependent on the switch state $[S]$. Thus the network equivalent must be re-synthesized after each switch transition[11].

We now have network equations of the form

$$\begin{bmatrix} G_{AA} & B_A \\ 0 & -T^* \end{bmatrix} \begin{bmatrix} V_A(t) \\ I(t) \end{bmatrix} = \begin{bmatrix} J_A(t) \\ -J_A^*(t) \end{bmatrix} - \begin{bmatrix} G_{AB} \\ C_B^* \end{bmatrix} [V_B(t)] \tag{5.85}$$

where $[J_A^*(t)] = [A_{hA}^*][J_h(t)] + [A_{jA}^*][J_j(t)]$. Note that the ideal-switch branches are now coupled to a pseudo-Thevenin equivalent of the linear part of the network

$$[V_o^*(t)] - [T^*][I(t)] = [0] \tag{5.86}$$

where $[V_o^*(t)] = [C_B^*][V_B(t)] + [J_A^*(t)]$ represents the open circuit voltage. Eq. (5.85) is dubbed a pseudo-Thevenin equivalent since the solution $[I(t)]$ is a vector containing short circuit currents and zero currents depending on the switch states.

5.5 Solution method

A straightforward non-iterative solution method simply solves the network with the switch states computed at the previous time-step. This is consistent with the existing solution method in the EMTP. A new network equivalent is synthesized only if a switch transition occurs.

variable_structure_solve_code(

$t \leftarrow t_0, [J_h] \leftarrow [J_{h0}]$

while $t < t_f$,

build_rhs(

compute$([V_B], [J_j])$

$[rhs] \leftarrow [A_{hA}][J_h] + [A_{jA}][J_j] - [G_{AB}][V_B]$

)

Algorithm 5.5 Transient solve code for variable structure networks (Continued . . .)

[11] $[G_{AA}]$, however, need only be triangularized once prior to the first time-step.

solve_compensation_currents(

$[J_A^*] \leftarrow [A_{hA}^*][J_h] + [A_{jA}^*][J_j]$

$[V_o^*] \leftarrow [C_B^*][V_B] + [J_A^*]$

$[I] \leftarrow [T^*]^{-1} \cdot [V_o^*]$

$[rhs] \leftarrow [rhs] - [B_A][I]$

)

solve(

$[V_A] \leftarrow$ **backsub**$([G_{AA}], [rhs])$

)

switch_transition(

$[S, transition] \leftarrow$ **switch**$([A_n] \cdot [V], [I])$

if $transition$,

$[C_B^*, T^*, A_{hA}^*, A_{jA}^*] \leftarrow$ **net_equiv**$(G_{AA}, G_{AB}, B_A, A_{hA}, A_{jA}, S)$

end

)

update_history_terms(

$[J_h] \leftarrow [U_h][G_h][A_v][V] + 0.5[U + U_h][J_h]$

)

$t \leftarrow t + \frac{\Delta t}{2}$

end_while

)

Algorithm 5.5 Transient solve code for variable structure networks

5.6 Numerical example: Transient analysis

Consider the single-phase half-bridge rectifier with freewheeling diode shown in Fig. 5.14(a). Fig. 5.14(b) depicts the corresponding discrete-time network with the diodes represented as ideal switches with states s_1 and s_2.

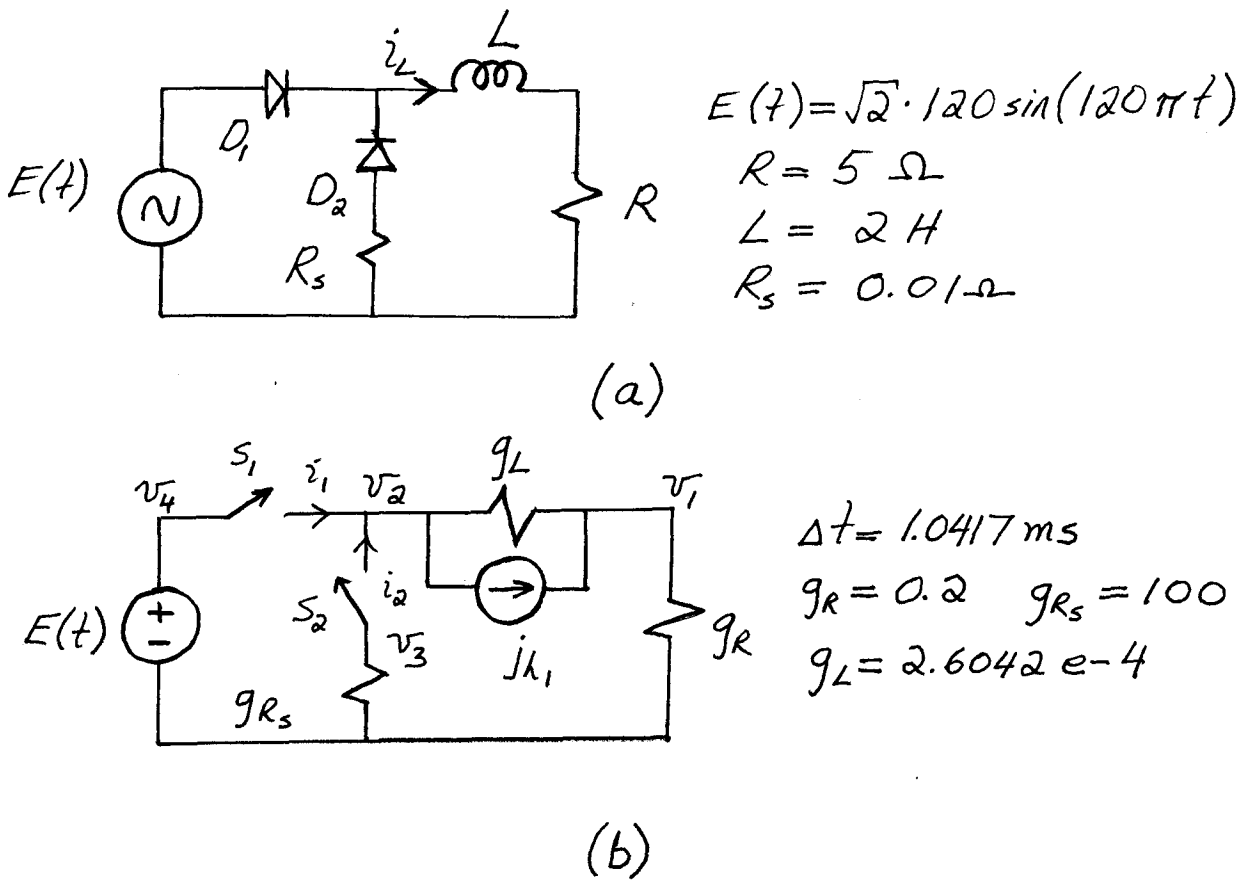

Figure 5.14 Half-bridge rectifier with freewheeling diode (a) continuous-time network (b) discrete-time network

Formulation of the network equations: With nodes and history sources defined as in Fig. 5.14(b) the discrete-time network equations become

$$\begin{bmatrix} g_L+g_R & -g_L & 0 & 0 & 0 \\ -g_L & g_L & 0 & -1 & -1 \\ 0 & 0 & g_{R_s} & 0 & 1 \\ 0 & -s_1 & 0 & \overline{s_1} & 0 \\ 0 & -s_2 & s_2 & 0 & \overline{s_2} \end{bmatrix} \begin{bmatrix} v_1(t) \\ v_2(t) \\ v_3(t) \\ i_1(t) \\ i_2(t) \end{bmatrix} = \begin{bmatrix} j_h(t) \\ -j_h(t) \\ 0 \\ 0 \\ 0 \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ 0 \\ s_1 \\ 0 \end{bmatrix} [v_4(t)] \qquad (5.87)$$

where $j_h(t)$ is updated as in (5.83).

Triangularization and network equivalents: The initial switch states $s_1 = s_2 = 0$ are assumed thus only $[G_{AA}]$ need be upper triangularized prior to the first time-step. For the example at hand with $\Delta t = 1.25ms$ and switches initially open downward operations

transform the augmented matrix (5.84) to

$$\begin{bmatrix} 0.2003 & -0.0003 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0.0003 & 0 & 0 & -1 & -1 & -0.9987 \\ 0 & 0 & 100.0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \tag{5.88}$$

Solution method: Algorithm 5.5 with parameters $t_0 = 0$ and $t_f = 200ms$ is applied to the sample network. Fig. 5.15 represents the progression of the load current and node 2 voltage waveforms $i_L(t)$ and $v_2(t)$ from zero initial conditions. Note that even after twelve cycles the solution has not yet settled to the periodic steady-state.

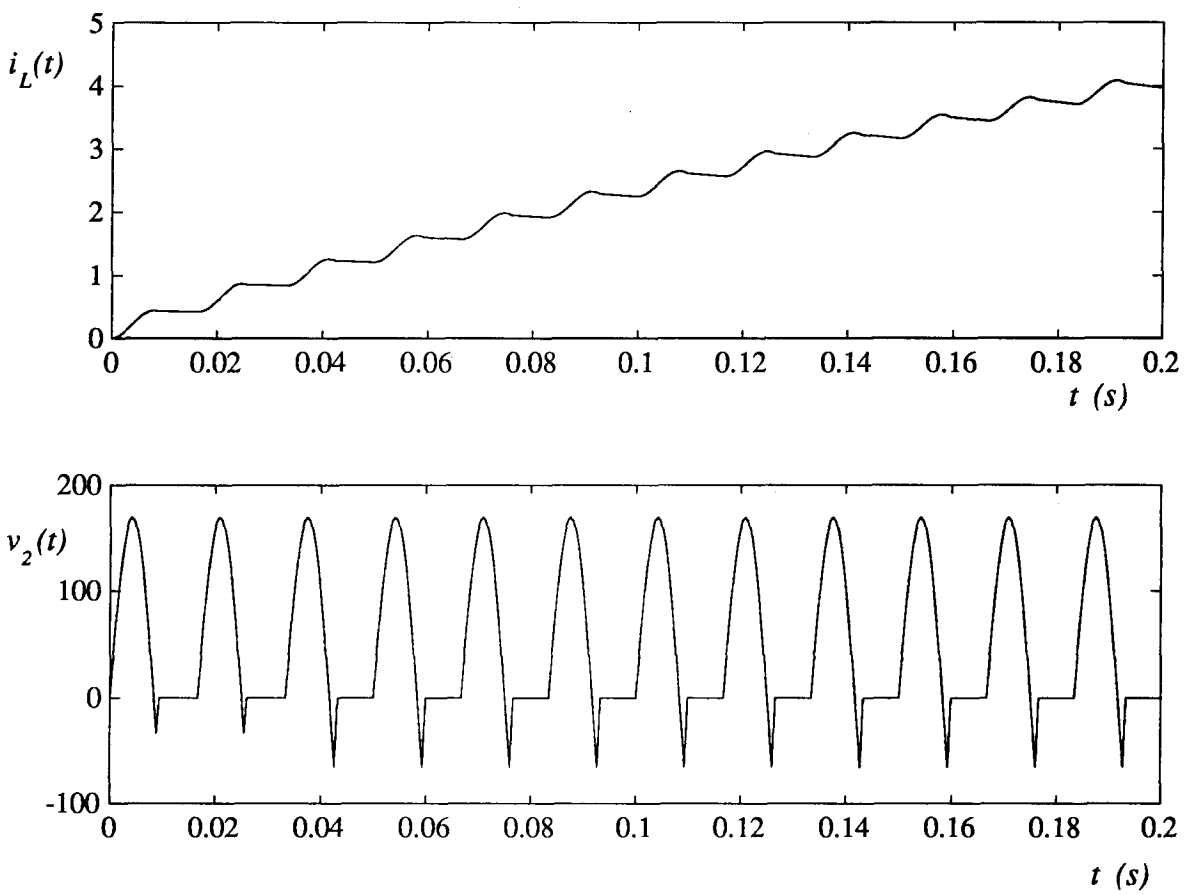

Figure 5.15 Solution of sample network from zero initial conditions

5.7 Steady-state initialization algorithm

Based on the development introduced in Chapter 3, a steady-state initialization algorithm is implemented within the variable structure network solve code as in Algorithm 5.6.

steady_state_init(

$t \leftarrow 0, t_f \leftarrow T, [J_h] \leftarrow [0], [DJ_h] \leftarrow [U]$

while $t < t_f$,

build_rhs(

compute$([V_B], [J_j])$

$[rhs] \leftarrow [A_{hA}][J_h] + [A_{jA}][J_j] - [G_{AB}][V_B]$

$[Drhs] \leftarrow [A_{ha}][DJ_h]$

)

solve_compensation_currents(

$[J_A^*] \leftarrow [A_{hA}^*][J_h] + [A_{jA}^*][J_j]$

$[V_o^*] \leftarrow [C_B^*][V_B] + [J_A^*]$

$[I] \leftarrow [T^*]^{-1} \cdot [V_o^*]$

$[DI] \leftarrow [T^*]^{-1}[A_{hA}^*][DJh]$

$[rhs] \leftarrow [rhs] - [B_A][I]$

$[Drhs] \leftarrow [Drhs] - [B_A][DI]$

)

solve(

$[V_A] \leftarrow$ **backsub**$([G_{AA}], [rhs])$

$[DV_A] \leftarrow$ **backsub**$([G_{AA}], [Drhs])$

)

switch_transition(

$[S, transition] \leftarrow$ **switch**$([A_n] \cdot [V], [I])$

if $transition$,

$[C_B^*, T^*, A_{hA}^*, A_{jA}^*] \leftarrow$ **net_equiv**$(G_{AA}, G_{AB}, B_A, A_{hA}, A_{jA}, S)$

Algorithm 5.6 Time-domain steady-state initialization algorithm for variable structure networks (Continued . . .)

end

)

update_history_terms(

$[J_h] \leftarrow [U_h][G_h][A_v][V] + 0.5[U + U_h][J_h]$

$[DJ_h] \leftarrow [U_h][G_h][A_{vA}][DV_A] + 0.5[U + U_h][DJ_h]$

)

$t \leftarrow t + \frac{\Delta t}{2}$

end_while

newton_raphson_iteration(

$[J_h] \leftarrow [DJ_h - U]^{-1} \cdot [-J_h]$

)

)

Algorithm 5.6 Time-domain steady-state initialization algorithm for variable structure networks

5.8 Numerical example: Steady-state initialization

Fig. 5.16 represents the solution from the steady-state initial conditions computed by the time-domain method. Note the scaling of the axis in the first plot. The waveform represents a dc current with ripple.

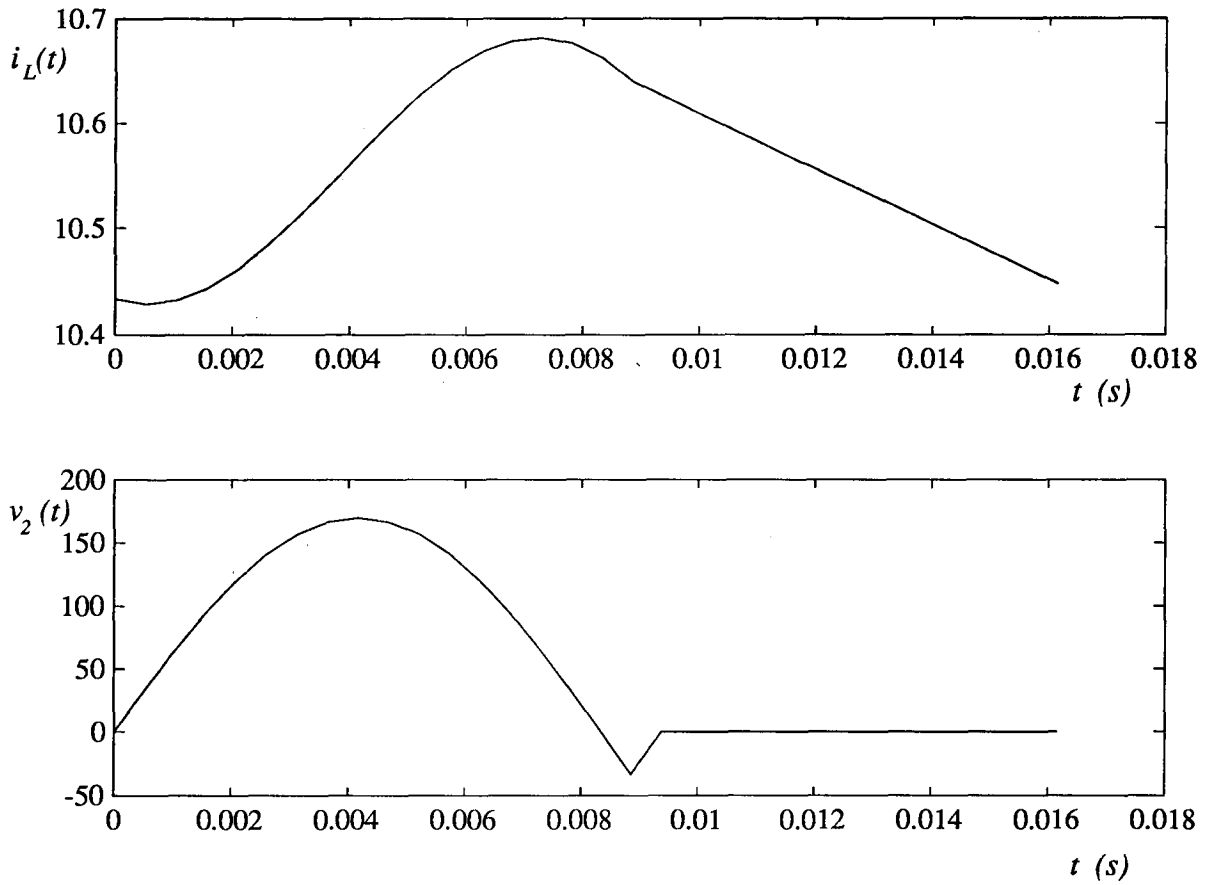

Figure 5.16 Solution of sample network from steady-state initial conditions

5.9 Application: Initialization of single and three-phase bridge networks

The following are sample cases from [11]. Figure 5.17 depicts a single-phase diode bridge rectifier. Figure 5.18 shows the steady-state waveforms obtained from Algorithm 5.6 for $\Delta t = 130.2083 \mu s$.

Figure 5.20 depicts the steady-state waveforms for the three-phase diode bridge rectifier (Fig. 5.19). Note that there is no frequency-domain method for the solution of such networks.

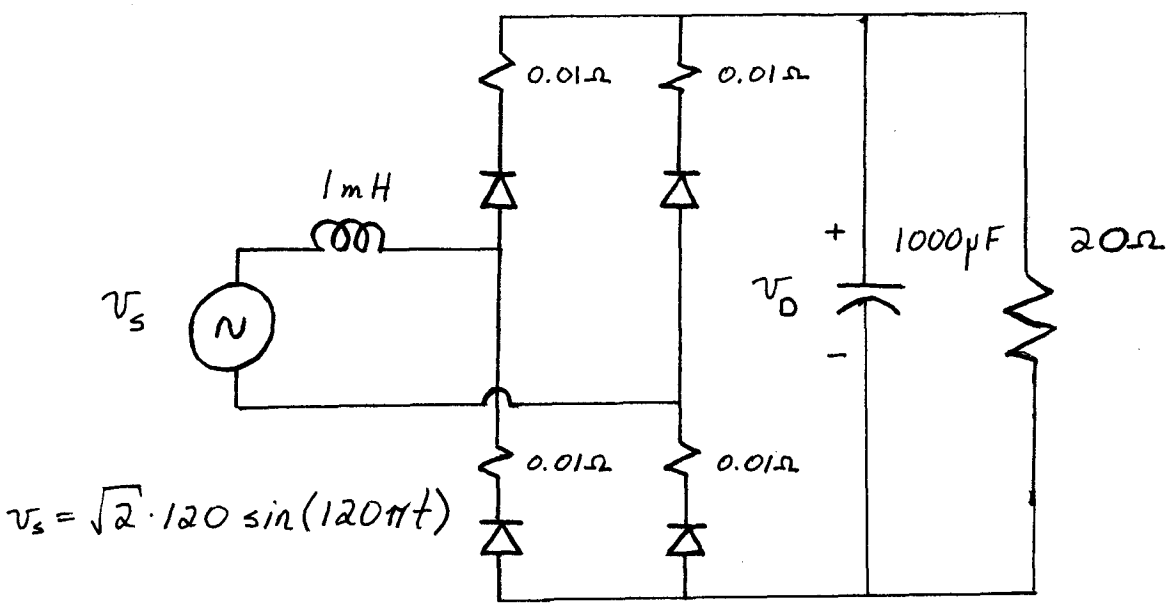

Figure 5.17 Single-phase diode bridge rectifier

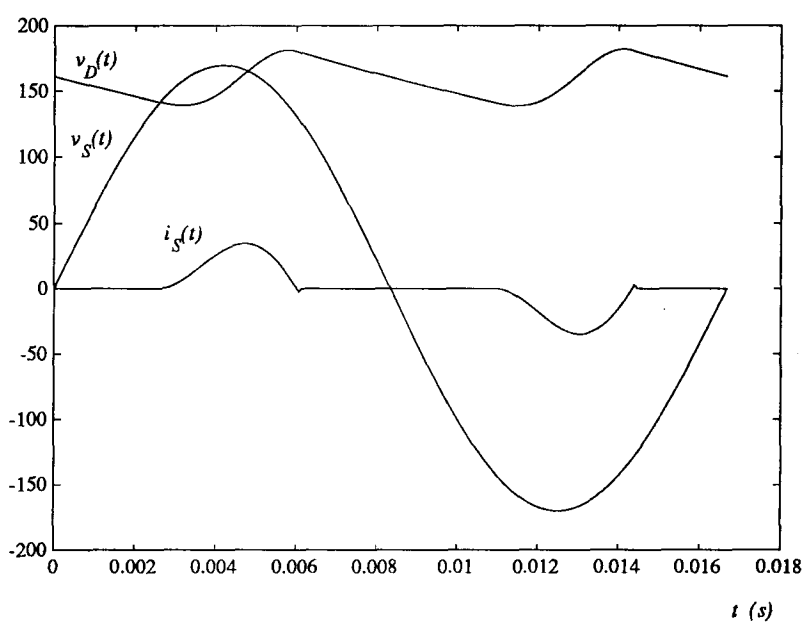

Figure 5.18 Steady-state waveforms for single-phase diode bridge rectifier

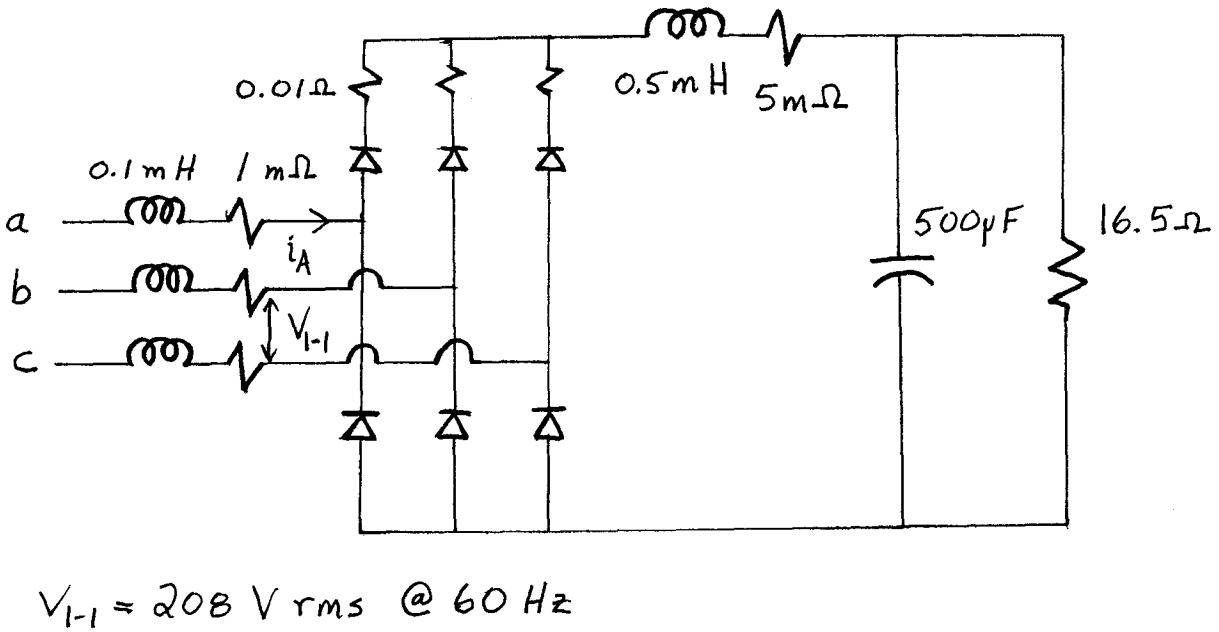

Figure 5.19 Three-phase diode bridge rectifier

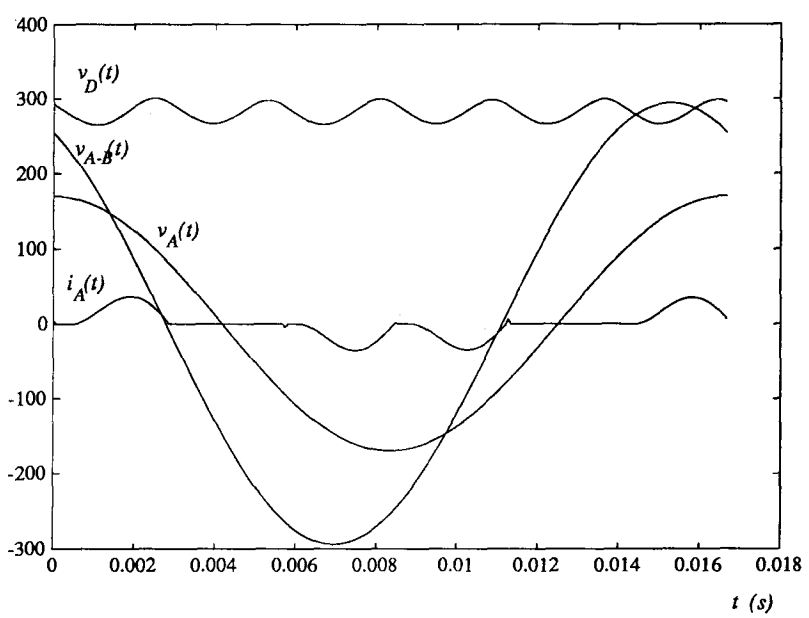

Figure 5.20 Steady-state waveforms for three-phase diode bridge rectifier

Chapter 6 Conclusions

6.1 Summary

A time-domain based steady-state initialization algorithm implementable within the solve code typical of electromagnetic transient programs (EMTP) has been proposed. The algorithm is developed for the three classes of lumped element networks: linear, nonlinear, and variable structure.

The primary contribution of this work is that each network element does not require a dedicated steady-state model and a separate steady-state solution algorithm. The proposed algorithm can be implemented in existing electromagnetic transients programs with minimal programming effort as the steady-state initialization process is consistent with the time-domain character of the transient simulations themselves.

Future efforts might extend the time-domain method to handle the initilization of distributed parameter elements (line models), TACS models, and machine models.

References

[1] H. Dommel, "Digital computer solution of electromagnetic transients in single and multiphase networks," *IEEE Transactions on Power Apparatus and Systems*, vol. PAS-88, no. 4, pp. 388–399, April 1969.

[2] H. Dommel, "Nonlinear and time-varying elements in digital simulation of electromagnetic transients," *IEEE Transactions on Power Apparatus and Systems*, vol. PAS-90, no. 6, pp. 2561–2567, Nov./Dec. 1971.

[3] J. R. Marti and J. Lin, "Supression of numerical oscilations in the emtp," *IEEE Transactions on Power Systems*, vol. PAS-4, no. 2, pp. 739–747, May 1989.

[4] H. D. A. Yan and S. Wei, "Harmonics from transformer saturation," *IEEE Transactions on Power Delivery*, vol. PWRD-1, no. 2, pp. 209–214, April 1986.

[5] A. S. E. Acha and J. Arrilaga, "Newton-type algorithms for the harmonic analysis of nonlinear power circuits in periodical steady state with special reference to magnetic nonlinearities," *IEEE Transactions on Power Delivery*, vol. PWRD-3, no. 3, pp. 1090–1098, July 1988.

[6] T. Parker and L. Chua, *Practical Numerical Algorithms for Chaotic Systems*. New York: Springer-Verlag, 1989.

[7] T. Aprille and T. Trick, "Steady-state analysis of nonlinear circuits with periodic inputs," *IEEE Proceedings*, vol. 60, no. 1, pp. 108–114, January 1972.

[8] L. Chua and P.-M. Lin, *Computer-aided Analysis of Electronic Circuits*. Englewood Cliffs, New Jersey: Prentice Hall, 1975.

[9] A. R. Chung-Wen Ho and P. Brennen, "The modified nodal approach to network analysis," *IEEE Transactions on Circuits and Systems*, vol. CAS-22, no. 6, pp. 504–509, June 1975.

[10] J. Usaola and J. G. Mayordomo, "Fast steady-state technique for harmonic analysis," *IEEE Transactions on Power Delivery*, vol. PWD-6, no. 4, October 1991.

[11]N. Mohan, *Computer Exercises for Power Electronics Education*. Minneapolis, MN: University of Minnesota, 1990.