estimate may be obtained if the following "stability radius" type of question is answered: "Given $A \in \mathcal{D}$ determine the smallest r such that for $\|\Delta A\| > r$, in a given norm $\|\cdot\|$, $(A + \Delta A) \notin \mathcal{D}$." This is a problem currently under investigation. If the nonlinearities are of class $C^r, r \geq 2$, and bounds on the norms of the Jacobian matrices are available, then it may be possible to use bounds on the size of the neighborhoods in the inverse function theorem [20, p. 119] in the problem of estimating changes in the equilibrium due to perturbations.

Finally, note that existence and uniqueness of the equilibrium point are provable under a weaker condition ($\Theta \in \mathcal{P}_0$) than the condition assumed ($-\Theta \in \mathcal{D}$) and the same may be true of the stability proof. Thus, it is of interest to find the weakest conditions under which the results in this note hold.

Appendix

Lemma A1: $A \in \mathcal{D} \Rightarrow AK \in \mathcal{D}$, for all $K > 0$ diagonal.

Proof: $A \in \mathcal{D} \Rightarrow \exists P > 0$ diagonal such that $PA + A^TP = -Q < 0$

Thus,

$$KPAK + KA^TPK = -KQK < 0$$

or, $P_1(AK) + (AK)^TP_1 < 0$; $P_1 := PK = KP > 0$; diagonal. □

Lemma A2: $A \in \mathcal{D} \Rightarrow A^{-1} \in \mathcal{D}$.

Proof: $A \in \mathcal{D} \Rightarrow \exists P > 0$ diagonal such that $PA + A^TP = -Q < 0$.

Since $A \in \mathcal{D}$ is nonsingular

$$(A^{-1})^TPA(A^{-1}) + (A^{-1})^TA^TP(A^{-1}) = (A^{-1})^TP + P(A^{-1})$$
$$= -(A^{-1})^TQ(A^{-1}) < 0. \qquad \square$$

Acknowledgment

The authors would like to thank one of the reviewers who alerted them to the existence of a relationship between parametric and structural stability, which led to the discussion and comparisons in Section 2.1. They would also like to thank Prof. Jacob Palis of IMPA for some clarifications about the concept of structural stability.

References

[1] D. Tank and J. Hopfield, "Simple neural optimization networks: an A/D converter, signal decision circuit, and a linear programming network," *IEEE Trans. Circ. and Syst.*, vol. CAS-33, pp. 533–541, May 1986.
[2] G. Avitabile, M. Forti, S. Manetti, and M. Marini, "On a class of nonsymmetrical neural networks with application to ADC ," *IEEE Trans. Circ. and Syst.*, vol. CAS-38, pp. 202–209, Feb. 1991.
[3] M. Forti, S. Manetti, and M. Marini, "A condition for global convergence of a class of symmetric neural circuits," *IEEE Trans. Circ. and Syst.*, vol. 39, pp. 480–483, June 1992.
[4] T. Roska, "Some qualitative aspects of neural computing systems," in *Proc. 1988 IEEE Int. Symp. Circ. Syst.*, pp. 751–754, June 1988.
[5] D. Hershkowitz, "Recent directions in matrix stability," *Lin. Algebra Appl.*, vol. 171, pp. 161–186, 1992.
[6] E. Kaszkurewicz and A. Bhaya, "Robust stability and diagonal Liapunov functions," *SIAM J. Matrix Anal. Appl.*, vol. 14, pp. 508–520, Apr. 1993.
[7] K. Matsuoka, "Stability conditions for nonlinear continuous neural networks with asymmetric connection weights," *Neural Networks*, vol. 5, pp. 495–500, 1992.
[8] P. J. Moylan, "Matrices with positive principal minors," *Lin. Algebra Appl.*, vol. 17, pp. 53–58, 1977.
[9] M. Ikeda, Y. Ohta, and D. Šiljak, "Parametric stability," in *New Trends in Systems Theory—Proc. of the Univ. Genova - The Ohio State Univ. Joint Conf.*, (G. Conte, A. Perdon, and B. Wyman, Eds.), July 1990.
[10] A. Michel, J. Farrell, and W. Porod, "Qualitative analysis of neural networks," *IEEE Trans. Circ. and Syst.*, vol. 36, pp. 229–243, Feb. 1989.
[11] J. Hopfield, "Neurons with graded response have collective computational properties like those of two-state neurons," *Proc. Nat. Acad. Sci.*, vol. 81, pp. 3088–3092, 1984.
[12] J. Hopfield and D. Tank, "Computing with neural circuits: a model," *Sci.*, vol. 233, pp. 625–633, 1986.
[13] I. Sandberg and A. Willson, Jr., "Some network-theoretic properties of nonlinear dc transistor networks," *Bell Syst. Tech. J.*, pp. 1293–1311, May-June 1969.
[14] S. Persidskii, "Problem of absolute stability," *Automat. and Remote Cont.*, vol. 12, pp. 1889–1895, 1969.
[15] M. Fiedler and V. Pták, "On matrices with nonpositive off-diagonal elements and positive principal minors," *Czechoslovak Math. J.*, vol. 12, pp. 382–400, 1962.
[16] E. Kaszkurewicz and A. Bhaya, "On a class of globally stable neural circuits," *Techn. Rep.*, COPPE/UFRJ, 1992.
[17] M. Hirsch and S. Smale, *Differential Equations, Dynamical Systems and Linear Algebra.* San Diego, CA: Academic Press, 1974.
[18] F. Salam, Y. Wang, and M. Choi, "On the analysis of dynamic feedback neural nets," *IEEE Trans. Circ. and Syst.*, vol. 38, no. 2, pp. 196–201, 1991.
[19] B. Kosko, *Neural Networks and Fuzzy Systems.* Englewood Cliffs, N.J.: Prentice-Hall, 1992.
[20] R. Abraham, J. Marsden, and T. Ratiu, *Manifolds, Tensor Analysis, and Applications.* 2nd Ed., New York: Springer-Verlag, 1988.

Modified Hopfield-Tank Neural Networks Applied to the "Unitized" Maximum Flow Problem

Toshinori Munakata, Yoshiyasu Takefuji, and Henrik Johansson

***Abstract*— Two new approaches called "graph unitization" are proposed to apply neural networks similar to the Hopfield-Tank models to determine optimal solutions for the maximum flow problem. They are: (1) *n*-vertex and n^2-edge neurons on a unitized graph; (2) *m*-edge neurons on a unitized graph. Graph unitization is to make the flow capacity of every edge equal to 1 by placing additional vertices or edges between existing vertices. In our experiments, solutions converged most of the time, and the converged solutions were always optimal, rather than near optimal.**

I. Introduction

The maximum flow problem is one of several well-known basic problems for combinatorial optimizations in weighted directed graphs. Because of its importance in many areas of applications, such as computer science, engineering, and operations research, the maximum flow problem has been extensively studied by many researchers using a variety of methods [1], [2]. They include: a classic approach [3], translation of the maximum flow problem into maximal flow problem in layered network [4], an O(n^2 log n) parallel algorithm [5], O(n^2 log n) to O(n^3) distributed algorithms [6], [7], and recent

Manuscript received March 31, 1992; revised June 12, 1993. This paper was recommended by Associate Editor Kenji Onaga.

T. Munakata is with the Department of Computer and Information Science, Cleveland State University, Cleveland, OH 44115.

Y. Takefuji and H. Johansson, Department of Electrical Engineering, Case Western Reserve University, Cleveland, OH 44106.

IEEE Log Number 9215706D.

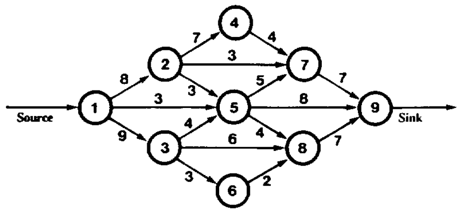

Fig. 1. A maximum flow problem, where the numbers at the edges are the flow capacities.

sequential algorithms of, e.g., O(mn) for $m > n^{1+\epsilon}$ for any constant ϵ [8], with n vertices and m edges. Also, a genetic algorithm, another "guided random search" technique as neural networks, has recently been applied to the problem [9]. The problem discussed here is to determine an optimal solution for a given unidirected, integer-weighted graph with no loops (See Fig. 1). The weight at each edge represents the flow capacity of the edge. Under theses constraints, we want to maximize the total flow from the source (vertex no. 1) to the sink (vertex no. n).

Recently Hopfield-Tank neural network models have been applied to various optimization problems [10], [11]. In this article, we discuss two new approaches called "graph unitization" to apply modified Hopfield-Tank models to the maximum flow problem. Graph unitization means to make the flow capacity of every edge equal to one. This is performed by placing additional vertices/edges between the existing vertices in the first approach. In the second approach, additional edges are placed between existing vertices. The reason for these unitization processes is that considering potential complications of the use of a Hopfield-Tank type model [12], [13], we followed many other applications of two-value (0 or 1) models (e.g., the traveling salesman problem (TSP) [10]).

As in other Hopfield-Tank models, we think of neurons as abstract representation of the vertices and edges of a graph, which do not carry any values themselves. Actual numeric values are carried by variables associated with these neurons. An "equation of motion" is employed to direct u_{ij} for balancing and maximizing the flow, where u_{ij} is an input to edge neuron ij for an edge from vertex i to vertex j. Using u_{ij}, an "activation function" determines flow V_{ij}, the output of edge neuron ij. Starting with randomized values of an initial u_{ij}, the equations are repeatedly used to determine an optimal solution. Our models do not assume symmetric connectivity and continuous state and time, which are deviations from standard Hopfield-Tank models. The optimality of the Hopfield-Tank type models is an open problem, and generally it is not guaranteed. In our experiments, solutions converged most of the time, and the converged solutions were always optimal, rather than near optimal.

II. Model 1: n-Vertex And n^2-Edge Neurons On A Unitized Graph

2.1 The Algorithm

A. Unitization of the Original Graph: The graph is unitized by placing additional vertices between existing vertices so that the flow capacity of every existing edge is 1. For example, if the flow capacity from vertices a to b is 3, then we insert three vertices, say, a_1, a_2, and a_3 between a and b, so that each of the six new edges has a flow capacity of 1 (Fig. 2). After unitization, the graph for Fig. 1 is converted to a graph of n = 92 vertices. The graph is represented by the flow capacity matrix, $[c_{ij}]$, i, j = 1, n, where c_{ij} = 1 if an edge from vertex i to vertex j exists, and c_{ij} = 0 otherwise.

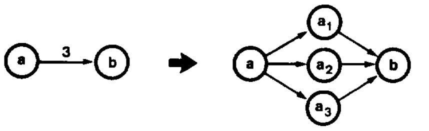

Fig. 2. Converting a general weighted directed graph to a unitized directed graph by inserting new vertices.

B. Basic Equations: We consider n vertex neurons for i = 1, n, and n^2 edge neurons for i= 1, n and j = 1, n. Two variables, u_{ij} and V_{ij}, are associated with edge neuron ij. In the following, (1) is the equation of motion and (2) is an incremental equation, which together determine u_{ij}. Equation (3) is the activation function. Our problem is to determine whether V_{ij} = 1 (flow) or V_{ij} = 0 (no flow).

$$du_{ij}/dt = A\{(1+\alpha RND_i)EX_i - (1+\alpha RND_j)EX_j\} + B \quad (1)$$

$$u_{ij}^{(t+1)} = u_{ij}^{(t)} + (du_{ij}/dt)\Delta t \quad (2)$$

$$V_{ij} = 0 \text{ if } u_{ij} < 0; V_{ij} = 1 \text{ otherwise} \quad (3)$$

In the above, A, B, and α are constants. RND_i is a uniform random number over [0, 1] for vertex i. EX_i represents an excess inflow into vertex i and relates to V as defined in the Step 1 below. The A terms are for balancing the flow and the last B term is for maximizing the flow. No "energy equation" is used in this model. Instead, the solution of the flow itself is checked, as described in Steps 3 and 4 below.

C. Initialization of Values Associated with the Vertex and Edge Neurons: The edge neurons associated with c_{ij} = 0 are deactivated. Next, following the convention of neural network models, initial values of u_{ij}, $u_{ij}^{(0)}$, are assigned randomly, then initial solution V_{ij} is computed using (3). Initialization of variables associated with the vertex neurons are not needed.

D. Iteration Process: Repeat the following steps until a maximum flow is obtained in Step 4.

- Step 1. Compute the following for each of n - 2 vertex neurons, i = 2, n -1. (i = 1 for the source, i = n for the sink). $I_i = \sum V_{pi}$, (Total) inflow to vertex i, where Σ is taken over the incoming edges to vertex i. $O_i = \sum V_{iq}$, (Total) outflow from vertex, where Σ is taken over the outgoing edges from vertex i. $EX_i = I_i - O_i$, Excess in flow to (if $EX_i > 0$) or outflow from (if $EX_i < 0$) vertex i. When EXi = 0, vertex i is said to be locally balanced. (Exceptions: $EX_1 = EX_n = 0$ always)
- Step 2. Compute the following for each of vertex i, i = 1, n - 1. $BLK_i = 1$ if vertex i is blocked; $BLK_i = 0$ otherwise. Vertex i is blocked if for all outgoing edges from vertex i to vertices j's: (1) V_{ij} = 1 (i.e., edge ij is saturated), or (2) vertex j is blocked. The sink is never blocked.
- Step 3. Check globally balanced condition, BALANCE, where BALANCE = 1 if every vertex is locally balanced; BALANCE = 0 otherwise.
- Step 4. If BALANCE = 1 (i.e., the flow is globally balanced), and $BLK_i = 1$ (i.e., the source is blocked), then STOP. Otherwise, continue.
- Step 5. Compute du_{ij}/dt, u_{ij}, and V_{ij} for the next iteration step using (1)–(3). Go to Step 1.

2.2 Experimental Results

The algorithm was applied to various sizes of graphs; for each graph, the program was run 10 times starting with different random

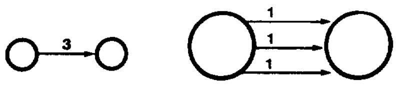

Fig. 3. Unitizing a general weighted directedgraph to a unitized graph by splitting vertices.

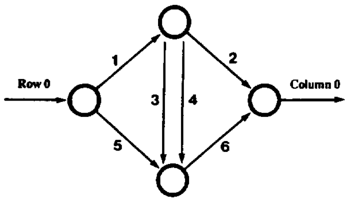

Fig. 4. A simple unitized graph with the edge numbers. The flow capacity of each edge is 1.

number seeds. All cases converged giving optimal solutions. The value of the time step, Δt, was 0.01. For the unitized version of Fig. 1 (n = 92), the average number of iterations was 568.5, with the constant values: A = 20, B = 5, and α= 17.

The sequential time complexity of each iteration is O(n) for Steps 1 through 4 and O(m) for Step 5. Hence, the complexity is O(m) if m $\gg n$, roughly O(n) otherwise. For parallel implementation, the upper bound of Step 2 is O(n). As in many other Hopfield-Tank model applications, the difficult part is predicting the number of iterations. In this model, the number depends on several factors: the size of the graph, the characteristics of the flow matrix, the values of constant coefficients, and each run that uses different random values. Based on our limited experience, the number of iterations appears to be roughly a low-order polynomial of n, say, O(n).

III. Model 2. m-Edge Neurons On A Unitized Graph

3.1 The Algorithm

A. Unitization of a Graph: In this approach, the general weighted directed graph for our problem is unitized first by redrawing unit-capacity edges between each pair of vertices, where the number of new edges is equal to the original flow capacity. For example, when the flow capacity of an edge is 3, we would replace the edge with three unit-capacity edges (see Fig. 3).

B. Numbering Edges and Determining Three Fixed Matrices, F, S, and D: After the unitization, we number the edges as 1, 2, ...,m (see Fig. 4 where m = 6). As we can see in Fig. 4, an imaginary edge going into the source may be numbered as "row 0", and another imaginary edge coming out of the sink may be numbered as "column 0". Our problem is to determine whether V_k, the flow at each edge, is 0 or 1 for k = 1 to m.

Next, we define three fixed matrices F, S, and D for the given unitized graph as follows:

F, the flow matrix: $F = [f_{ij}], i,j = 0,m; f_{ij} = 1$ if there can be a flow from edge i to edge j, $f_{ij} = 0$ otherwise. S, the common originating vertex matrix: $S = [s_{ij}], i,j = 0,m; s_{ij} = 1$ if edges i and j have a common originating vertex, $s_{ij} = 0$ otherwise. D, the common destination vertex matrix: $D = [d_{ij}], i,j = 0,m$. This is similar to matrix S, except that the term "originating" is replaced by "destination". $d_{ij} = 1$ if edges i and j have a common destination vertex, $d_{ij} = 0$ otherwise. Matrices S and D are symmetric with respect to the main diagonals.

C. The Equation of Motion: The equation of motion for edge neuron k is given as follows.

$$du_k/dt = (1 - f_{0k})\sum V_i(f_{ik} - s_{ik}) + (1 - f_{k0})\sum V_i(f_{ki} - d_{ki}) \tag{4}$$

In the first term, the summation is taken over the edges incident with the vertex where edge k originates, and the term checks whether the originating vertex is balanced. If the total inflow and outflow are balanced at the vertex, then the summation will be zero. If there is more inflow than outflow, then the summation will be positive, and this effects to increase the flow of edge k. If there is more outflow than inflow, the term has the reversed effect. There is one exception to the first term. When the originating vertex is the source, the flow balance constraint should be dropped. The (1 - f_{0k}) factor is for this purpose. The second term is the same as the first, except that it checks the flow balance condition of the ending vertex of edge k.

In our model, the equation of motion contains only the flow balancing condition without maximizing effect. The effect can be easily implemented; e.g., in a simple way, we can add a constant term in the above equation. However, as we will see in the next, an appropriate flow initialization seems to work well for a practical purpose.

D. Iteration Process: The edges are saturated at the beginning. This is done by initializing u_k to the upper tenth of the allowed interval. For example, if the interval is [-50, 50], u_k are initialized in the interval of [40, 50]. The hysteresis McCulloch-Pitts function is used as our activation function to suppress possible oscillatory behavior [11]. The iterations are terminated when the entire flow is balanced.

3.2 Experimental Results

Many runs for different sizes of graphs, the values of m ranging 30 to 200, were carried out. The overall performance was good. In most cases the iterations converged with 100% or close to 100 (e.g., 98) % of the time. The worst case was for a graph having 200 neurons (edges). With the maximum number of iterations set to 2,000, the rate was 80%, requiring the average 1,020 iterations. The Fig. 1 problem converged at 100% rate and required 155.3 iterations as an average. For all the cases, whenever iterations converge, the final solutions were optimal. The reason for this optimality is most likely due to the initialization scheme where the flow is saturated.

The time complexities for each iteration are O(m) for sequential, and O(1) for parallel. As in Model 1, the number of iterations is hard to predict. Based on our experience, the number of iterations appears to be roughly a low-order polynomial of m and n, say, O(mn).

IV. Conclusions

We have experimented with the two new graph unitization approaches to apply Hopfield-Tank-like models to the maximum flow problem. The rates of convergence were 100% for the first model and near 100% for the second model. Both models gave optimal solutions when they converge. A drawback common to both methods is that the increased number of vertices and edges by the unitizing processes. The number of required iterations is likely reduced further by fine tuning of the algorithms. In summary, the models discussed in this article and their variations can be potential methods for solving weighted graph problems in general.

Acknowledgment

We would like to thank graduate student Abby Duh for her programming efforts.

References

[1] R. E. Tarjan, *Data Structures and Network Algorithms*. Philadelphia, PA: Soc. for Ind. and Appl. Math., 1983.
[2] J. A. McHugh, Algorithmic Graph Theory, Englewood Cliffs, NJ: Prentice-Hall, chapter 6, 1990.
[3] L. R. Ford Jr. and D. R. Fulkerson, *Flows in Networks*. Princeton, NJ: Princeton University Press, 1962.
[4] E. A. Dinic, "Algorithm for solution of a problem of maximum flow in a network with power estimation," *Soviet Math. Dokl.*, vol. 11, pp. 1277-1280, 1970.
[5] Y. Shiloach and U. Vishkin, "An $O(n^2 \log n)$ parallel MAX-FLOW algorithm," *J. Algorithm*, vol. 3, pp. 128-146, 1982.
[6] J. Marberg and E. Gafni, "An $O(n^3)$ distributed maximum flow algorithm," in *Conf. on Inf. Sci. & Sys.*, Princeton, NJ, pp. 478- 482, 1984.
[7] D. Y. Yeh and T. Munakata, "A maximal flow algorithm in a distributed layer network," in *Proc. Int. Comput. Symp.*, Tainan, Taiwan, Dec., pp. 1221-1227, 1986.
[8] V. King, S. Rao, and R. Tarjan, "A faster deterministic maximum flow algorithms," *ACM-SIAM Symp. on Discrete Algorithms*, vol. 3, pp. 157-164, 1992.
[9] T. Munakata and D. J. Hashier, "A genetic algorithm applied to the maximum flow problem," in *5th Int. Conf. on Genetic Algorithms*, Urbana-Champaign, IL, July 17-22, 1993.
[10] J. J. Hopfield and D. W. Tank, "'Neural' computation of decisions in optimization problems," *Biological Cybern.*, vol. 52, pp. 141- 152, 1985.
[11] Y. Takefuji, *Neural Network Parallel Computing*. Boston, MA: Kluwer, 1992.
[12] G. V. Wilson and G. S. Pawley, "On the stability of the travelling salesman problem algorithm of Hopfield and Tank," *Biological Cybern.*, vol. 58, pp. 63-70, 1988.
[13] B. Kamgar-Parsi and B. Kamgar-Parsi, "On problem solving with Hopfield neural networks," *Biological Cybern.*, vol. 62, pp. 415-423, 1990.

A Newton-Powell Modification Algorithm For Harmonic Balance-Based Circuit Analysis

D. D'Amore, P. Maffezzoni, and M. Pillan

***Abstract*— In this paper a new numerical technique is proposed for the solution of nonlinear systems of equations. The techniques presented here are based on a *Modification* algorithm, and it has been employed to improve the numerical efficiency of harmonic balance-based frequency domain circuit simulator (Spectre). This technique exhibits high computational efficiency and rate of convergence, presenting its best performance in case of strong nonlinearities and high integration level. To validate the convergence properties of this technique, some simulations were performed and the results were compared with those obtained using Newton and Newton-Samanskii methods.**

I. Introduction

The analysis of nonlinear integrated circuits subjected to large input signals is still a critical problem, and great interest is now focused on numerical techniques that improve the efficiency and capability of existing simulation tools [1], [2]. Whenever the steady-state circuit response is required, time-domain circuit simulators (such as Spice) may not be efficient and accurate, since they need to perform a transient analysis to reach the steady state, while harmonic balance–based frequency domain simulators (HB) allow a direct and accurate periodic solution and a better modeling of distributed components. On the other hand, HB is not suitable in the simulation of large-scale integration circuits or in presence of large nonlinearities, since its computational complexity strongly increases with the number of considered harmonics. In circuit simulators, the Newton method is generally employed: thereafter in case of large circuits or strongly nonlinear ones, a great contribution to the computation burden is due to the LU decomposition required by each iteration. In literature, some simplification algorithms have been presented (i.e., Newton-Samanskii method [3]), but these methods can only be employed for almost linear circuits since they present poor convergence properties [4].

In this work a new modification algorithm is presented that allows a meaningful improvement of the computation efficiency, quite maintaining Newton accuracy and convergence properties. This technique is mainly based on the application of the Davidon-Fletcher-Powell formula [3], [5] for the approximation of the inverse Jacobian matrix. This technique, which has been efficiently applied to the steady-state simulation with harmonic balance, could be extended to traditional time domain circuit simulators, improving their performances in case of very large systems or strongly nonlinear ones.

Manuscript received July 27, 1992; revised June 7, 1993. This paper was recommended by Associate Editor G. De Micheli.

D. D'Amore and M. Pillan are with the Dipartimento di Elettronica e Informazione, Politecnico di Milano, Via Ponzio, 34/5, I20133, Milano, Italy.
P. Maffezzoni is with the Dipartimento di Elettronica per l'Automazione University of Brescia 38 I25123 B, Italy.

IEEE Log Number 9214254.

II. Fundamentals

In the frequency domain, the modified nodal analysis (MNA) equations describing a nonlinear independent n-node circuit take the form

$$\boldsymbol{F}(\boldsymbol{V}) = \boldsymbol{I}(\boldsymbol{V}) + \Omega \boldsymbol{Q}(\boldsymbol{V}) + \boldsymbol{Y}\boldsymbol{V} + \boldsymbol{I}_S = 0 \qquad (1)$$

where

- $\boldsymbol{V}$-Fourier transform of the node voltage vector $\boldsymbol{v}(t)$ ($v_j(t)$ being the j-th node voltage) and $\boldsymbol{I}(\boldsymbol{V})$-Fourier transform of $\boldsymbol{i}[v(t)]$ (the generic component $i_j[\boldsymbol{v}(t)]$ being the sum of the currents due to nonlinear resistors connected to the jth node);
- $\Omega \boldsymbol{Q}(\boldsymbol{V})$-Fourier transform of $d\boldsymbol{q}(\boldsymbol{v})dt$ (the components $q_j[\boldsymbol{v}(t)]$ representing the sum of charges due to nonlinear capacitors connected to the jth node);
- $\boldsymbol{Y}\boldsymbol{V}$-Fourier transform of $\int_{-\infty}^{+\infty} \boldsymbol{y}(t-\tau)\boldsymbol{v}(\tau)\,d\tau$ corresponding to the currents due to linear elements. It is assumed that $\int_{-\infty}^{+\infty} \boldsymbol{y}(t)^T \boldsymbol{y}(t)\,dt < \infty$, which means that the electrical circuit is asymptotically stable;
- $\boldsymbol{I}_S$ corresponds to the sum of the currents injected by the independent current sources.

If only the first h harmonics are considered, the problem reduces to the solution of $\boldsymbol{F}(\boldsymbol{V}) = 0$, where $\boldsymbol{F}(\boldsymbol{V})\colon \Re^{n\times(2h+1)} \rightarrow \Re^{n\times(2h+1)}$, that is to the solution of $d = n(2h+1)$ nonlinear equations.

Several numerical techniques can be employed to achieve the solution of such a problem [4], [6]. The most employed is the Newton method, which also exhibits good convergence properties in presence of strong nonlinearities.

In computer implementation, the generic Newton iteration is formulated as

$$\boldsymbol{J}(\boldsymbol{F}(\boldsymbol{V}_k))(\boldsymbol{V}_{k+1} - \boldsymbol{V}_k) = -\boldsymbol{F}(\boldsymbol{V}_k) \qquad (2)$$