Complex Systems **9** (1995) 29–45

Signal Flow Graphs as an Efficient Tool for Gradient and Exact Hessian Determination

Stanislaw Osowski
Institute of the Theory of Electrical Engineering, and Electrical Measurements, Technical University, 00-661 Warsaw, pl. Politechniki 1, Poland

Jeanny Herault
Laboratoire TIRF, INPG, 46, av. F. Viallet, 38031 Grenoble, France

Abstract. This paper presents the application of signal flow graphs to the determination of the exact values of gradient vector and hessian matrix for a linear system. It is shown all information about the gradient and hessian is contained in the original and adjoint signal flow graphs at different terminations. To determine the gradient we have to perform two analyses, and to get the full hessian matrix we perform $(2K + 1)$ analyses of the graph (or the system), where K is the number of internal nodes of the graph. The examples included in the paper illustrate the method.

1. Introduction

The signal flow graph (SFG) [1, 2] is a graphical representation of a linear system of equations of the form

$$\mathbf{A}\mathbf{x} + \mathbf{b} = \mathbf{0}. \tag{1}$$

The vector $\mathbf{b}$ corresponds to the excitations of the system. The matrix $\mathbf{A}$ represents the connection weights between the variables forming vector $\mathbf{x}$. Without loss of generality we assume that all diagonal entries of $\mathbf{A}$ are equal to -1, that is,

$$\mathbf{A} = \begin{bmatrix} -1 & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & -1 & a_{23} & \cdots & a_{2n} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ a_{n1} & a_{n2} & a_{n3} & \cdots & -1 \end{bmatrix} \tag{2}$$

The system (1) can be represented in pictorial form as a flow of signals between nodes, where each node corresponds to the variable x_i and the value of

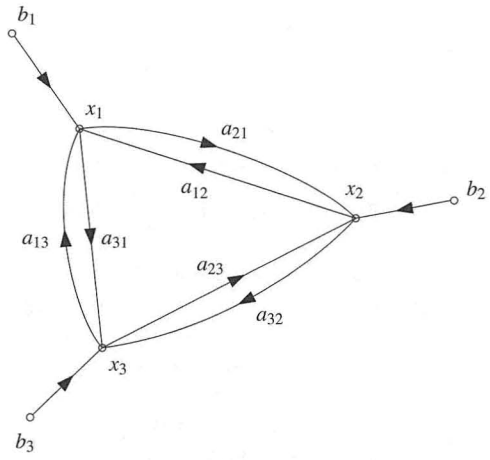

Figure 1: SFG corresponding to the 3 × 3 system of linear equation of the example.

the signal associated with the node is the sum of weighted signals flowing into that node. Figure 1 presents the example of a 3-node graph that represents the linear system of equations of the form

$$\begin{bmatrix} -1 & a_{12} & a_{13} \\ a_{21} & -1 & a_{23} \\ a_{31} & a_{32} & -1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} = \mathbf{0}.$$

Each node signal x_i $(i = 1, 2, 3)$ is given as the sum of weighted signals entering the ith node, that is, $x_i = \sum_{k=1, k\neq i}^{3} a_{ik} x_k + b_i$.

The SFG representation of the system has found many applications in the analysis and synthesis of linear systems [1, 2, 5, 6, 11]. As a graphical representation of the flow of the signals, the graph is an important factor in understanding the behavior and performance of the system at different working conditions. The Mason gain formula applied to the graphs simplifies the problem of analysis of such system and yields the solution in an explicit form of the parameters. There are already many computer programs that solve SFGs in an efficient way, making this method of analysis very interesting from the practical point of view.

The other application of the graph is the sensitivity calculation. Sensitivity is an important measure of the quality of the circuit and its ability to perform well in a noisy environment. It has been shown that the sensitivity of the system, defined as the derivative of any of its signal with respect to the gain of the branch, can be evaluated as the product of a signal in the original graph G and a signal in the adjoint graph $\widehat{G}$ [5, 11]. The linear graph $\widehat{G}$ is called *adjoint* to G if all its branches are reversed branches of G. After reversing the directions of branches the roles of nodes have also

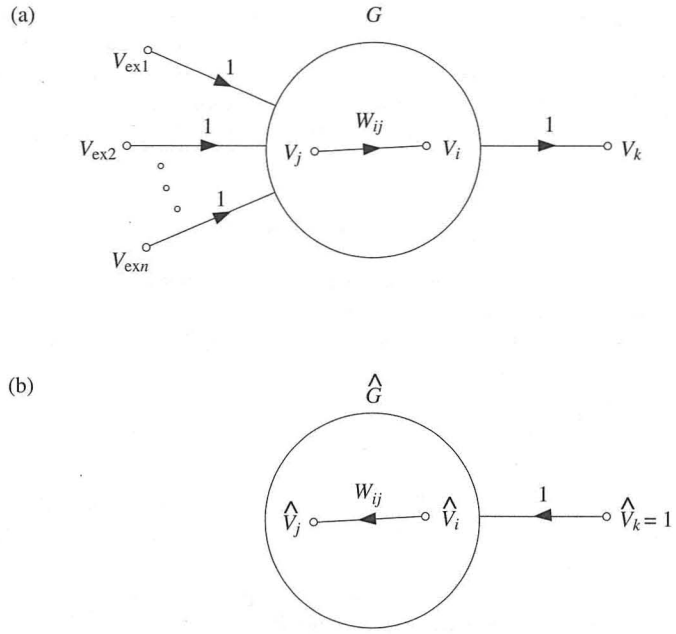

Figure 2: Illustration of the sensitivity calculation using SFG. (a) Original graph G with terminations. (b) Terminations of adjoint graph $\widehat{G}$.

been changed. The former summing node is now the simple source node, and vice versa. The adjoint graph, which corresponds directly to the notion of a transposed system in system theory, has been defined for both linear and nonlinear branches. Here we will be concerned mainly with the linear ones. If the ith node signal in the original and adjoint graph are denoted by V_i and $\widehat{V}_i$, respectively, and the gain of the branch from the jth to the ith node is denoted by W_{ij}, then the sensitivity of any node signal V_k of the system with respect to the gain W_{ij} may be described by the formula [5, 11]

$$\frac{dV_k}{dW_{ij}} = V_j \widehat{V}_i^{(k)} \tag{3}$$

in which $\widehat{V}_i^{(k)}$ means the signal of the ith node in the adjoint graph $\widehat{G}$ at the unity excitation applied at kth node (Figure 2), and V_j is the signal of the jth node of G at normal excitations $V_{ex1}, V_{ex2}, \ldots, V_{exn}$.

The sensitivity and its extension in the form of a gradient is of great importance in supervised learning strategies of neural networks [7, 11]. SFGs represent a new look at the process of learning of such systems and may be convenient in the analysis of a particular algorithm or its implementation in circuit form.

In this paper we will show that SFG is a convenient and efficient tool for determining the gradient and exact evaluation of the hessian matrix. The sensitivity relation (3) will be generalized to the gradient of the specified energetic function and then to the matrix of second-order derivatives, the hessian. Specific examples illustrating the proposed method will be given and discussed.

2. Gradient determination using SFG

Usually in engineering applications—such as neural networks or an optimization approach to the design of electronic circuits—the energetic (objective) function is defined as the quadratic function of the chosen node signals; that is,

$$E(\mathbf{W}) = \frac{1}{2}\sum_{k=1}^{M}(V_k - d_k)^2 \tag{4}$$

where V_k is the actual kth node signal, d_k is the desired value for this node, and M is the number of output nodes that take part in the definition of the energetic function. Although equation (4) is quadratic with respect to the node signals it is generally nonlinear with respect to the optimized parameters (adjusted weights W_{ij}) because V_k is a nonlinear function of the weights.

The gradient ∇E is defined as the vector of derivatives of the energetic function with respect to the parameters (weights) of the system. Simple differentiation of (4) leads to the relation

$$\frac{\partial E}{\partial W_{ij}} = \sum_{k=1}^{M}(V_k - d_k)\frac{\partial V_k}{\partial W_{ij}} \tag{5}$$

which states that the derivative of the objective function needs the calculation of sensitivity of the appropriate node signals. Application of SFG can significantly simplify this process. Note that the adjoint SFG is a linear one (even in the case of a nonlinear system, not considered here). Instead of applying the unity excitation at the node $\widehat{V}_k$ of $\widehat{G}$ and summing up the appropriately weighted products of the signals of G and $\widehat{G}$, we can apply all excitations at once (Figure 3) forcing the linear superposition rule to generate the relation (5) in the form of simple multiplication of two signals, analogously to (3):

$$\frac{\partial E}{\partial W_{ij}} = V_j\widehat{V}_i. \tag{6}$$

The only difference between sensitivity (relation (3)) and gradient calculation (relation (6)) is the excitation of the adjoint graph [11]. In the case of sensitivity it is the unity signal applied at the appropriate node of $\widehat{G}$, while in the gradient calculation this signal is equal to the difference between the actual value of V_k and the target value d_k.

The differential relations (3) and (6) have clear network interpretation form, because each graph is equivalent to some network. The sensitivity or

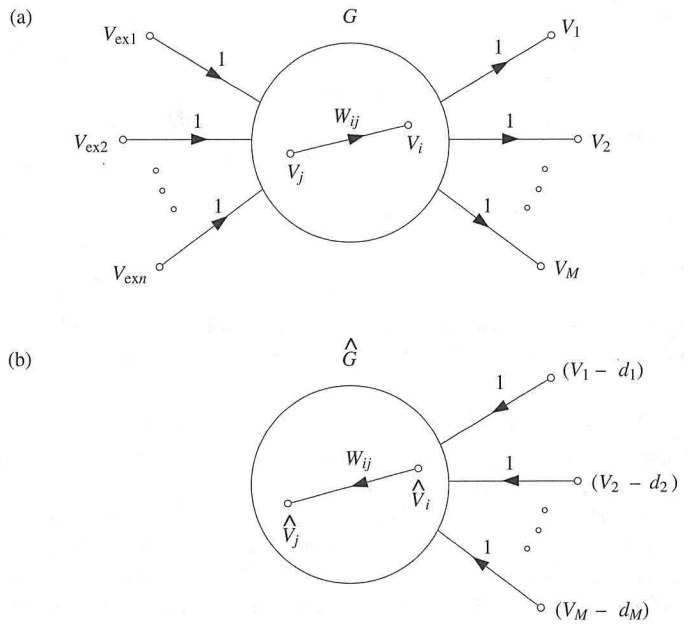

Figure 3: The graphs G and $\widehat{G}$ for gradient calculations. (a) Original graph G with terminations. (b) Terminations of adjoint graph $\widehat{G}$.

gradient analysis requires that we enlarge the original network by the adjoint one and analyze both. The results of the analysis form the gradient vector, which among other things may be used to adapt the weights of the system to provide the output vector $\mathbf{V}_{\text{out}}$ equal to $\mathbf{d}$. The enlarged network presented in the form of a self-adapting system is presented in Figure 4. The pairs of learning vectors $(\mathbf{V}_{\text{ex}}, \mathbf{d})$ are employed in the self-adaptation process. The vector $\mathbf{V}_{\text{ex}}$ is the excitation vector of the original system G, while the output $\mathbf{V}_{\text{out}}$ and target vector $\mathbf{d}$ are the excitation for the transposed system $\widehat{G}$. The outputs of both systems $\mathbf{V}$ and $\widehat{\mathbf{V}}$ are applied as the inputs to the block that generates the gradient-based adaptive algorithm—such as steepest descent, conjugate gradient, or variable metric—that adapts the weights of G and $\widehat{G}$ to make the output vector $\mathbf{V}_{\text{out}}$ of G equal to the target vector $\mathbf{d}$. The network is self-adapting and automatically adjusts its weights in the direction of minimization of the defined energetic function. The enlarged network can be fully realized in hardware form and its performance simulated on the computer using a circuit analysis package.

It should be restated that the linear graph is only a graphical representation of the linear system of equations expressed generally as $\mathbf{Ax} + \mathbf{b} = \mathbf{0}$, where $\mathbf{A}$ contains the information of the weights, $\mathbf{b}$ is the excitation vector, and $\mathbf{x}$ is the vector of node variables. On the other hand, the adjoint graph represents another linear system of equations $\mathbf{A}^t\hat{\mathbf{x}} + \hat{\mathbf{b}} = \mathbf{0}$, where this time

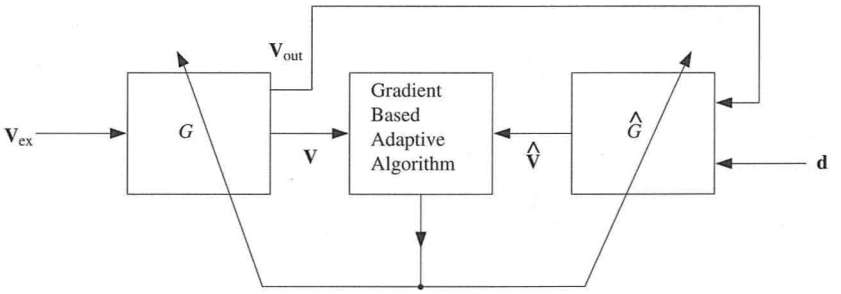

Figure 4: Illustration of the use of the generalized concept of an adjoint system for self-adaptation (learning) of the system.

the excitation vector $\hat{\mathbf{b}}$ is formed on the basis of a solution of a normal system. Pure numerical implementation of the algorithm is possible by applying any numerical linear solver.

The important advantage of the flow-graph method is that it is a simple way of taking into account the shared values of the weights. If any weight appears many times in the graph as the gain of different branches, the sensitivity and gradient formulas with respect to this gain are the simple superpositions of the appropriate individual derivatives of each branch separately. To be more specific, if the weight W appears twice in the network, say between the nodes i, j and k, l (the second index means the node from which the branch starts), and all nodes are denoted by V (normal system) and $\widehat{V}$ (the adjoint system), with appropriate index, respectively, then the gradient component with respect to the shared weight W is described by the sum of two components. This follows from the fact that the weight W appears twice (between nodes i, j and k, l). For this particular case the appropriate component of the gradient is given by

$$\frac{\partial E}{\partial W} = V_j \widehat{V}_i + V_l \widehat{V}_k. \tag{7}$$

It can be stated that this rule, coming out from the signal flow relations, corresponds directly to the active constraints method, the most effective way of taking into account linear constraints in an optimization problem. Thanks to this strategy the equality constraints of the type described above do not complicate a learning process that involves the gradient; in fact the opposite is true, reducing the problem complexity by reducing the number of effective variables that appear in the calculations.

As an example consider the eigenvalue problem of the $n \times n$ real symmetric matrix $\mathbf{B}$ [3] defined by

$$\mathbf{B} = \mathbf{V}^t \mathbf{L} \mathbf{V}$$

where $\mathbf{L} = \mathrm{diag}[\lambda_1, \lambda_2, \ldots, \lambda_n]$ with λ_i being the eigenvalue and $\mathbf{V}$ being the orthogonal matrix composed of n orthogonal vectors $\mathbf{v}_i$, $\mathbf{V} = [\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n]$,

where each vector $\mathbf{v}_i$ is the eigenvector that corresponds to λ_i. The matrix $\mathbf{V}$ is of unity length, that is, $\mathbf{V}^t\mathbf{V} = \mathbf{1}$. By multiplying these equations at each time t by a nonzero vector $\mathbf{x} = [x_1(t), x_2(t), \ldots, x_n(t)]$ it has been shown [8] that we can formulate the eigenvalue problem as the minimization of the following energetic function E:

$$E = \frac{1}{2}[\| \mathbf{V}^t\mathbf{L}\mathbf{V}\mathbf{x} - \mathbf{B}\mathbf{x} \|_2^2 + \| \mathbf{V}^t\mathbf{V}\mathbf{x} - \mathbf{x} \|_2^2].$$

The weights W_{ij} to be calculated are the components of the eigenvectors $\mathbf{v}_i$ and the eigenvalues λ_i. Figure 5(a) presents the SFG in a matrix form that corresponds to the defined energetic function. The symbolic branches depicted inside the boxes denote the entries of the matrix $\mathbf{V}$ and $\mathbf{L}$ (the weights of the neural network). The terminating points (the nodes of the graph) are the components of the appropriate vectors of the graph. The branches denoted by the dashed lines define only the targets $\mathbf{d}^{(1)}$ and $\mathbf{d}^{(2)}$, and do not belong to the graph G. The adjoint graph for the gradient calculation is created by reversing the branches of G and, in the case of matrix descriptions, by transposing corresponding matrices (Figure 5(b)). The excitations of the adjoint graph are now formed by the appropriate differences of the actual output and its target values—that is, $\hat{\mathbf{y}}^{(1)} = (\mathbf{y}^{(1)} - \mathbf{d}^{(1)})$ and $\hat{\mathbf{y}}^{(2)} = (\mathbf{y}^{(2)} - \mathbf{d}^{(2)})$—and are placed at the former output nodes. To get the gradient component of any weight we have to multiply the node signals from which the weight (branch) originates in the normal and adjoint systems, as expressed by relation (6).

Applying this rule to the graphs of Figure 5 and taking into account that the elements of the matrix $\mathbf{V}$ appear at three different places in the graph, we get the final expressions for the components of gradient vectors:

- for eigenvalue λ_i,

$$\frac{\partial E}{\partial \lambda_i} = u_i \hat{z}_i$$

- for eigenvectors,

$$\frac{\partial E}{\partial v_{ij}} = [x_i \hat{u}_j + (y_i^{(1)} - d_i^{(1)}) z_j + (y_i^{(2)} - d_i^{(2)}) u_j]$$

  which can be described more generally in vector notation form as

$$\frac{\partial E}{\partial \mathbf{v}_k} = [\mathbf{x}\hat{u}_k + (\mathbf{y}^{(1)} - \mathbf{d}^{(1)}) z_k + (\mathbf{y}^{(2)} - \mathbf{d}^{(2)}) u_k].$$

The bold letters stand for vectors and the indexed variables are the components of the appropriate vectors as denoted in Figure 5. Note that only two analyses of the system are required to obtain all components of the gradient. The whole information is now contained in the solution of normal and adjoint graphs.

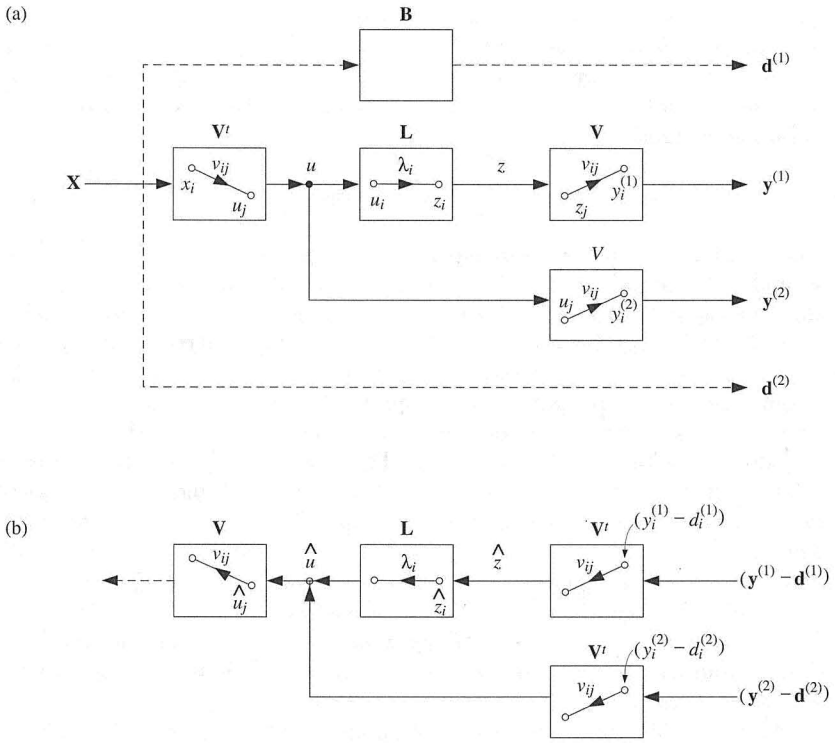

Figure 5: Signal flow graphs for eigenvalue problem. The boxes denote the matrices and the branches depicted inside the boxes are the adjusted weights. (a) Original matrix flow graph. (b) Adjoint matrix flow graph.

The SFG approach to the determination of the gradient may be applied in many fields, especially those that use optimization methods. An example of such a field is neural networks, where the flow graph approach presents a new perspective on the learning paradigm of such systems [11]. They may be used instead of backpropagation to train the hidden layers or to improve the performance of the self-organizing neural networks, such as source separation [9, 10].

3. Hessian determination using SFG

The hessian matrix $\mathbf{H}$ associated with the energetic function $\mathrm{E}(\mathbf{W})$, where $\mathbf{W}$ is the vector of adjusted weights $\mathbf{W} = [W_1, W_2, \ldots, W_n]$, is defined as the quadratic symmetric matrix of the second-order derivatives of the energetic

function

$$\mathbf{H} = \begin{bmatrix} H_{11} & H_{12} & \cdots & H_{1n} \\ H_{21} & H_{22} & \cdots & H_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ H_{n1} & H_{n2} & \cdots & H_{nn} \end{bmatrix} \tag{8}$$

where the (i,k)th component of it is defined by

$$H_{ik} = \frac{\partial^2 E}{\partial W_i \partial W_k} = \frac{\partial}{\partial W_k}\left(\frac{\partial E}{\partial W_i}\right). \tag{9}$$

The hessian matrix is of great importance in many fields. In optimization theory it is used in the second-order Newton's method of optimization [4] to define the direction $\mathbf{p}$ of search in n-dimensional space, $\mathbf{p} = -\mathbf{H}^{-1}\nabla E$. At the same time it is the measure of the curvature of the error surface in the optimized system, and as such, finds practical application in estimation of the sensitivity of the system to a given parameter. Recently it has been used as a measure of the saliency of weights in neural networks in the pruning process, called Optimal Brain Damage [13, 14].

As seen from relation (9), to determine the hessian we need to repeat the graph differentiation rule on each component of the gradient. However, in this case we should take into account that the excitation of the adjoint graph is also subject to differentiation. To avoid this inconvenience in hessian generation, we need to modify slightly the last step of the gradient calculation and apply directly relation (5). Each kth component of (5) should be calculated independently at unity excitation applied at the kth ($k = 1, 2, \ldots, M$) node of $\widehat{G}$; by using relation (3) the final result is now given in the form

$$\frac{\partial E}{\partial W_{ij}} = \sum_{k=1}^{M}(V_k - d_k)\frac{\partial V_k}{\partial W_{ij}} = \sum_{k=1}^{M}(V_k - d_k)V_j\widehat{V}_i^{(k)} \tag{10}$$

where this time $\widehat{V}_i^{(k)}$ is the ith node signal of $\widehat{G}$ with only the unity excitation applied at the kth node.

To simplify the notation let us assume that we consider only one output node ($M = 1$), denoted here as V_o, and that the weights W_i, W_k have the notations of the weights of graph branches, that is, $W_i = W_{ij}, W_k = W_{kl}$. By applying relation (10) for $M = 1$ we get

$$\frac{\partial^2 E}{\partial W_{ij}\partial W_{kl}} = \frac{\partial}{\partial W_{kl}}[\widehat{V}_i^{(o)}V_j(V_o - d)] \tag{11}$$

where V_j and V_o represent the solutions of the original graph G, $\widehat{V}_i^{(o)}$is the solution of the adjoint graph with unity excitation applied at the former output (o) node, and d represents the appropriate destination. The variables $\widehat{V}_i^{(o)}, V_j$, and V_o are the functions of the weights W_{ij} of the system. Differentiation in relation (11) leads to

$$\frac{\partial^2 E}{\partial W_{ij}\partial W_{kl}} = \frac{\partial \widehat{V}_i^{(o)}}{\partial W_{kl}}V_j(V_o - d) + \frac{\partial V_j}{\partial W_{kl}}\widehat{V}_i^{(o)}(V_o - d) + \frac{\partial V_o}{\partial W_{kl}}\widehat{V}_i^{(o)}V_j. \tag{12}$$

To get the final expression for the hessian on the basis of inspection of the graphs we should express the derivatives of $\hat{V}_i^{(o)}, V_j$, and V_o in the form of products of the appropriate variables of three graphs: the original one G, the graph $\hat{G}$ adjoint to the original and graph $\tilde{G}$ adjoint to $\hat{G}$. Observe that the internal structure of graph $\tilde{G}$ after reversing the branches of $\hat{G}$ is identical to G. The only difference between them may be in the excitations.

For clarity let us improve the notation further. By $\tilde{V}_j^{(e)}$ and $\hat{V}_j^{(e)}$ we denote the responses at the jth node of the graphs $\tilde{G}$ and $\hat{G}$, respectively, with the excitation placed at the eth node. These excitations are of unity values. The variables V_j of the original graph G will be denoted here without any superscript.

Relation (12) has transformed the calculation of the hessian to the determinations of three sensitivities. Two of them are of the original graph and one of the adjoint graph. Taking into account that the direction of the branch W_{ij} in $\hat{G}$ is opposite to the general convention (it starts now from the ith and ends up in the jth node) and repeating the sensitivity calculations to obtain $\partial V_j/\partial W_{kl}, \partial V_o/\partial W_{kl}$, and $\partial \hat{V}_i^{(o)}/\partial W_{kl}$, we get the final expression for the component of the hessian in the following form:

$$\frac{\partial^2 E}{\partial W_{ij}\partial W_{kl}} = \hat{V}_k^{(o)}\tilde{V}_l^{(i)}V_j(V_o - d) + \hat{V}_k^{(j)}V_l\hat{V}_i^{(o)}(V_o - d) + \hat{V}_k^{(o)}V_l\hat{V}_i^{(o)}V_j. \tag{13}$$

To get one entry of the hessian matrix we have to analyze three graphs: the original graph with fixed original excitations V_{ex}, the adjoint graph $\hat{G}$ with unity excitation applied at the node (which appears in the superscript of the appropriate variable $\hat{V}$) and graph $\tilde{G}$ of the internal structure identical to G with unity excitation applied at the node indicated here by the superscript of the variable $\tilde{V}$. If we follow the procedure of exciting the systems, we find that the analysis of the original system G is just part of the analysis of system $\tilde{G}$. Thus the determination of the full $n \times n$ hessian matrix requires only the analysis of two graphs ($\tilde{G}$ and $\hat{G}$), irrespective of the size of the problem. The adjoint graphs $\hat{G}$ and $\tilde{G}$ are to be analyzed at the excitations applied at different points indicated by the superscripts of the appropriate variables in relation (13). For full hessian matrix determination we have to apply the unity excitations in $\hat{G}$ and $\tilde{G}$ at every internal node. Denoting the total number of nodes by K (including the M output nodes) we have to analyze each of the graphs, $\hat{G}$ and $\tilde{G}$, at most K times. Taking into account that the original graph analysis is included in the results of analyses of $\tilde{G}$, we find that the total number of analyses of the system equals $2K$.

It should be pointed out that the crucial point in the graph analysis of the linear system, using for example the Mason rule or any of its derivatives [2, 6], is the determination of the so-called principal determinant Δ, which is independent of the excitations and is the same for all graphs in our problem. From the mathematical point of view this means that we have to solve the system of algebraic equations $\mathbf{Ax} + \mathbf{b} = \mathbf{0}$ for graphs G and $\tilde{G}$, and $\mathbf{A}^t\hat{\mathbf{x}} +$

$\hat{\mathbf{b}} = \mathbf{0}$ for graph $\widehat{G}$ at the same matrix $\mathbf{A}$ and at different excitation vectors $\mathbf{b}$ and $\hat{\mathbf{b}}$. The LU decomposition or inverse of the matrix $\mathbf{A}$ should be done only once, and multiple solutions of G and $\widehat{G}$ are found by putting the value 1 in different positions of the zero excitation vectors $\mathbf{b}$ and $\hat{\mathbf{b}}$.

To illustrate the practical aspects of determining the hessian matrix let us consider the linear system presented by the graph of Figure 6(a). Let us assume one original excitation $V_{\text{ex}} = E$ applied at the node V_1 and the output V_o placed at node V_3. Let the energetic function E($\mathbf{W}$), where $\mathbf{W} = [W_{12}, W_{21}, W_{31}, W_{32}]$, be assumed as follows:

$$E(\mathbf{W}) = \frac{1}{2}(V_o - d)^2$$

where d is the required destination. The graphs $\widehat{G}$ and $\tilde{G}$ corresponding to the given graph of the system are presented in Figures 6(b) and 6(c), respectively. The excitations and output nodes are not denoted there; they will change their positions according to the number indicated in the superscript of the appropriate variable. From inspection of the corresponding graphs it is seen that the gradient vector can be expressed in the form

$$\nabla E = \left[\frac{\partial E}{\partial W_{12}} \quad \frac{\partial E}{\partial W_{21}} \quad \frac{\partial E}{\partial W_{31}} \quad \frac{\partial E}{\partial W_{32}} \right]^t = \begin{bmatrix} V_2 \widehat{V}_1^{(3)} (V_o - d) \\ V_1 \widehat{V}_2^{(3)} (V_o - d) \\ V_1 \widehat{V}_3^{(3)} (V_o - d) \\ V_2 \widehat{V}_3^{(3)} (V_o - d) \end{bmatrix}.$$

The hessian is the 4×4 symmetric matrix, hence we have to determine only 10 entries of it. From relation (13) it directly follows that

$$\frac{\partial^2 E}{\partial W_{12} \partial W_{21}} = \widehat{V}_2^{(3)} \tilde{V}_1^{(1)} V_2 (V_o - d) + \widehat{V}_2^{(2)} \widehat{V}_1^{(3)} V_1 (V_o - d) + \widehat{V}_2^{(3)} \widehat{V}_1^{(3)} V_1 V_2$$

$$\frac{\partial^2 E}{\partial W_{12} \partial W_{31}} = \widehat{V}_3^{(3)} \tilde{V}_1^{(1)} V_2 (V_o - d) + \widehat{V}_3^{(2)} \widehat{V}_1^{(3)} V_1 (V_o - d) + \widehat{V}_3^{(3)} \widehat{V}_1^{(3)} V_1 V_2$$

$$\frac{\partial^2 E}{\partial W_{12} \partial W_{32}} = \widehat{V}_3^{(3)} \tilde{V}_2^{(1)} V_2 (V_o - d) + \widehat{V}_3^{(2)} \widehat{V}_1^{(3)} V_2 (V_o - d) + \widehat{V}_3^{(3)} \widehat{V}_1^{(3)} V_2 V_2$$

$$\frac{\partial^2 E}{\partial W_{21} \partial W_{31}} = \widehat{V}_3^{(3)} \tilde{V}_1^{(2)} V_1 (V_o - d) + \widehat{V}_3^{(1)} \widehat{V}_2^{(3)} V_1 (V_o - d) + \widehat{V}_3^{(3)} \widehat{V}_2^{(3)} V_1 V_1$$

$$\frac{\partial^2 E}{\partial W_{21} \partial W_{32}} = \widehat{V}_3^{(3)} \tilde{V}_2^{(2)} V_1 (V_o - d) + \widehat{V}_3^{(1)} \widehat{V}_2^{(3)} V_2 (V_o - d) + \widehat{V}_3^{(3)} \widehat{V}_2^{(3)} V_1 V_2$$

$$\frac{\partial^2 E}{\partial W_{31} \partial W_{32}} = \widehat{V}_3^{(3)} \tilde{V}_2^{(3)} V_1 (V_o - d) + \widehat{V}_3^{(1)} \widehat{V}_3^{(3)} V_2 (V_o - d) + \widehat{V}_3^{(3)} \widehat{V}_3^{(3)} V_1 V_2$$

$$\frac{\partial^2 E}{\partial W_{12}^2} = \widehat{V}_1^{(3)} \tilde{V}_2^{(1)} V_2 (V_o - d) + \widehat{V}_1^{(2)} \widehat{V}_1^{(3)} V_2 (V_o - d) + \widehat{V}_1^{(3)} \widehat{V}_1^{(3)} V_2 V_2$$

$$\frac{\partial^2 E}{\partial W_{21}^2} = \widehat{V}_2^{(3)} \tilde{V}_1^{(2)} V_1 (V_o - d) + \widehat{V}_2^{(1)} \widehat{V}_2^{(3)} V_1 (V_o - d) + \widehat{V}_2^{(3)} \widehat{V}_2^{(3)} V_1 V_1$$

$$\frac{\partial^2 E}{\partial W_{31}^2} = \widehat{V}_3^{(3)} \tilde{V}_1^{(3)} V_1 (V_o - d) + \widehat{V}_3^{(1)} \widehat{V}_3^{(3)} V_1 (V_o - d) + \widehat{V}_3^{(3)} \widehat{V}_3^{(3)} V_1 V_1$$

$$\frac{\partial^2 E}{\partial W_{32}^2} = \hat{V}_3^{(3)}\tilde{V}_2^{(3)}V_2(V_o - d) + \hat{V}_3^{(2)}\hat{V}_3^{(3)}V_2(V_o - d) + \hat{V}_3^{(3)}\hat{V}_3^{(3)}V_2V_2.$$

As seen from these expressions the graphs $\widehat{G}$ and $\tilde{G}$ should be analyzed at the unity excitations applied to all three nodes. If the values of the weights of the graph are given in numerical form, the problem is equivalent to the direct solution of the appropriate matrix equations $\mathbf{Ax} + \mathbf{b} = \mathbf{0}$ and $\mathbf{A}^t\hat{\mathbf{x}} + \hat{\mathbf{b}} = \mathbf{0}$, where in this particular case we have

$$\mathbf{A} = \begin{bmatrix} -1 & W_{12} & 0 \\ W_{21} & -1 & 0 \\ W_{31} & W_{32} & -1 \end{bmatrix} \quad \mathbf{x} = \begin{bmatrix} V_1 \\ V_2 \\ V_3 \end{bmatrix}$$

while $\mathbf{b}$ and $\hat{\mathbf{b}}$ are the excitation vectors specially selected for each graph (as described above). In the case of the symbolic weights of the graph we can apply the Mason topological formula, from which we obtain

$$\begin{aligned} D &= 1 - W_{12}W_{21} \\ V_1 &= E/D \\ V_2 &= EW_{21}/D \\ V_3 &= E(W_{31} + W_{21}W_{32})/D \\ V_o &= V_3 \end{aligned}$$

and

$$\begin{aligned} \hat{V}_1^{(1)} &= 1/D \\ \hat{V}_2^{(1)} &= W_{12}/D \\ \hat{V}_3^{(1)} &= 0 \\ \hat{V}_1^{(2)} &= W_{21}/D \\ \hat{V}_2^{(2)} &= 1/D \\ \hat{V}_3^{(2)} &= 0 \\ \hat{V}_1^{(3)} &= (W_{31} + W_{21}W_{32})/D \\ \hat{V}_2^{(3)} &= (W_{32} + W_{12}W_{31})/D \\ \hat{V}_3^{(3)} &= 1 \end{aligned}$$

and

$$\begin{aligned} \tilde{V}_1^{(1)} &= 1/D \\ \tilde{V}_2^{(1)} &= W_{21}/D \\ \tilde{V}_3^{(1)} &= (W_{31} + W_{21}W_{32})/D \\ \tilde{V}_1^{(2)} &= W_{12}/D \\ \tilde{V}_2^{(2)} &= 1/D \\ \tilde{V}_3^{(2)} &= (W_{32} + W_{12}W_{31})/D \end{aligned}$$

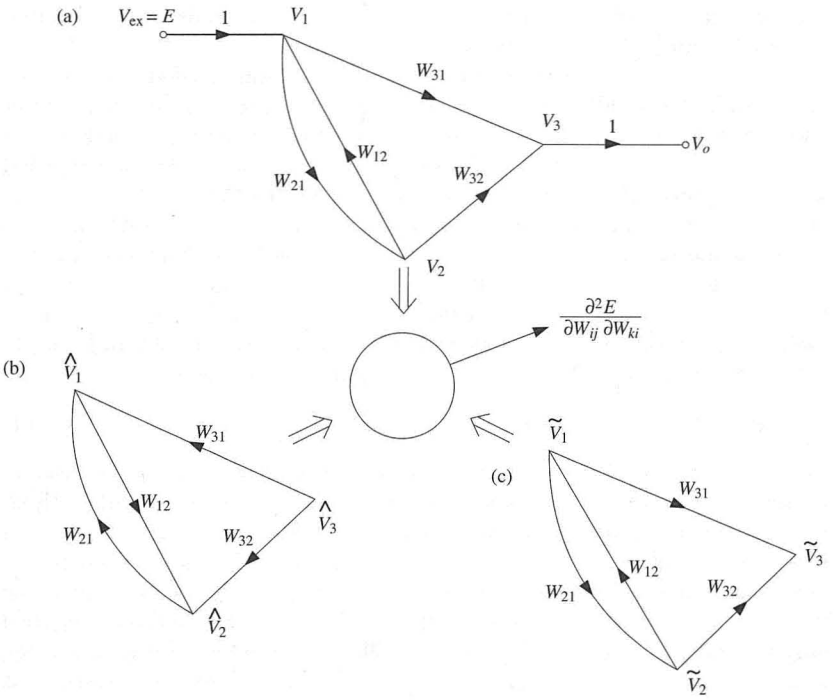

Figure 6: The signal flow graphs for hessian determination of the example. (a) Original graph G. (b) Adjoint graph $\hat{G}$. (c) Graph $\tilde{G}$ adjoint to $\hat{G}$.

$$\tilde{V}_1^{(3)} = 0$$
$$\tilde{V}_2^{(3)} = 0$$
$$\tilde{V}_3^{(3)} = 1.$$

The significant feature of the obtained results is the determination of exact values of the components of the gradient and hessian in a form very convenient from an algorithmic point of view. Instead of lengthy analytic calculations of derivatives, it is enough to analyze the system a few times at differently placed excitations.

The important advantage of the proposed hessian formula is a significant savings in memory requirements. For large-scale circuits, like neural networks, the number of weights is very high, whereas the number of nodes (neurons) is much smaller. This difference is typically of at least one or two orders. If we take, for example, the network of $K = 100$ neurons and $n = 5000$ weights, we would need $n^2/2 = 1.25 \times 10^7$ units of memory (the hessian matrix is symmetric and it is enough to memorize half of its entries). At the same time all the information needed to reproduce the full hessian using our formulation requires only $2K^2$ units to keep the solutions of our

three graphs. For our numerical example the memory requirement is limited to approximately 2×10^4 units of memory.

Much more striking is the comparison of calculations needed to obtain the full hessian. To get all entries of the hessian we need to perform $2K$ analyses and $3n^2$ multiplications for each output unit of the system. All analyses are performed on the same matrix $\mathbf{A}$, so the number of multiplications needed for them is proportional to $(K^3 + 3K^2)$. This means that the total number of multiplications in this case is proportional to $(3n^2M + K^3 + 3K^2)$.

In the most widely used general approximate perturbation method of the hessian calculation, we first need to perform $(n + 1)$ analyses of the system to get the gradient. In completely dense cases of the hessian, an additional n evaluations of gradients along n directions $\mathbf{d}^i$ are necessary to obtain $\frac{1}{2}n(n+1)$ entries of the hessian. This is due to the approximation formula

$$\mathbf{H}(\mathbf{x})\mathbf{d}^i = \nabla E(\mathbf{x} - \mathbf{d}^i) - \nabla E(\mathbf{x}) \tag{14}$$

for $i = 1, 2, \ldots, n$. All together this makes $n(n + 1)$ analyses of the system. Comparing this to $2K$ analyses required generally in our proposed method, we find that the rate of improvement is proportional to $n^2/2K$, in which n is the number of weights and K is the total number of neurons (hidden plus output). It should be noted that the number of operations to be done at the full hessian approximation is prohibitively high, and this method is applied only for small or sparse problems [15]. Different methods for approximating hessian generations are used, such as Gauss-Newton or Levenberg-Marquardt regularization and their derivatives [13, 15, 16, 17]. Those methods use information from the gradient and the last iteration results to reproduce the hessian at each cycle. Even then application of the gradient formula presented in this paper may greatly reduce the number of computations (two analyses of the system instead of $N + 1$ to obtain the value of the gradient vector). Application of our method has the additional advantage that the gradient is an exact value and the problem of approximation of a badly conditioned system is partly avoided.

Recently the original method of exact hessian determination for multilayer perceptron system was reported [18]. However, it applies only to feedforward systems and makes use of many simplifications that arise from this fact. As reported by the author, the number of analyses of the system (per training pattern) is at most equal to twice the number of hidden K_h and output M neurons, and this corresponds to $2K$ (in our notation $K = K_h + M$). The total number of mathematical operations scales like the square of the number of weights (n^2) in the system, where for the full feedforward network $n = K_h(M + K_i)$, in which K_i denotes the number of input nodes.

For the purpose of comparison we consider the same case using our method. At K_i input nodes, K_h hidden neurons, and M output neurons we need to perform $2(K_h + M)$ analyses of the system, and one analysis of the feedforward system requires a number of multiplications that is proportional to $K_h(K_i + M)$. That makes $2K_h(K_h + M)(K_i + M)$ total multiplications nedeed for the analyses of the system. The simplifications that follow from

the nature of the feedforward system result in a reduction of the different nonzero entries of the hessian to $K_h M$ only. Determination of each entry requires an additional six multiplications. In this case the number of multiplications in our method scales like $2K_h(K_h + M)(K_i + M) + 6K_h M$. If we examine different figures for the number of nodes we find that this number is smaller than the $[K_h(M + K_i)]^2$ needed by the Bishop method. Moreover, it should be noted that our approach is more general and applicable to the recurrent system as well.

4. Conclusions

This paper has presented the graph application to the exact calculation of the gradient vector and hessian matrix. By introducing the concept of adjoint graphs, simple formulas for defining the gradient vector and then the hessian matrix have been obtained. It was shown that all information about the first- and second-order derivatives of the energetic function is contained in the solution of the normal graph G and the graph $\widehat{G}$ adjoint to it. To get the full hessian matrix we need to solve the graph G once with originally given excitations, and K times with the unity excitation applied at each of K internal nodes of the graph. We also need to repeat these K solutions for the adjoint graph $\widehat{G}$. The $(2K + 1)$ analyses of the graphs are equivalent to the solutions of two matrix equations $\mathbf{A}\mathbf{x} + \mathbf{b} = \mathbf{0}$ and $\mathbf{A}^t\hat{\mathbf{x}} + \hat{\mathbf{b}} = \mathbf{0}$, with the same matrix $\mathbf{A}$ representing the weights of the graph and different vectors $\mathbf{b}$ and $\hat{\mathbf{b}}$ that contain all zeros except one position equal to unity (for graphs $\tilde{G}$ and $\widehat{G}$), or representing the given excitation of the original system (for a given graph G).

Instead of lengthy mathematical manipulations to determine the derivatives—which are extremely tedious and susceptible to errors, especially for the recurrent systems—we have the simple mathematical formulas based on the direct solution of two systems: the original and the adjoint. The method is algorithmic in its nature and is very easy in computer implementation. It may be applied in either numeric form (solution of two systems of linear algebraic equations) or in symbolic form using the Mason gain formula.

The method provides a significant reduction in memory requirements of the computer system and at the same time reduces the complexity of the calculations proportionally to the squared number of the nodes of the system.

The method may find practical application anywhere, where the curvature of the error surface and the information of the second-order derivatives of the energetic function is important—for example in the variety of second-order Newton's methods of optimization [3, 4], in the area of neural networks to develop new generation learning strategies based on second-order optimization algorithms, or on pruning of such networks [13, 14]. The method is applicable to both feedforward and feedback systems, presenting a new and efficient look at these old and still important problems.

Acknowledgments

This paper was completed while the first author was with the Laboratoire TIRF, INPG, Grenoble, France, financed by the EEC mobility grant.

References

[1] L. O. Chua and P. M. Lin, *Computer Aided Analysis of Electronic Circuits* (Englewood Cliffs, NJ: Prentice Hall, 1975).

[2] S. Mason and H. Zimmerman, *Electronic Circuits, Signals and Systems* (New York: Wiley, 1960).

[3] G. Golub and C. Van Loan, *Matrix Computation*, (North Oxford Academic, 1990).

[4] P. Gill, W. Murray, and M. Wright, *Practical Optimization*, (New York: Academic Press, 1981).

[5] A. Y. Lee, "Signal Flow Graphs—Computer Aided System Analysis and Sensitivity Calculation," *IEEE Transactions on Circuits and Systems*, **14** (1974) 227–228.

[6] C. Acar, "New Transformations in Signal Flow Graphs," *Electronics Letters*, **7** (1971) 27–28.

[7] L. Almeida, "Backpropagation in Perceptrons with Feedback," pages 199–208 in *Neural Computers*, edited by R. Eckmiller and C. Malsburg, (1987).

[8] A. Cichocki and R. Unbehauen, "Neural Networks for Computing Eigenvalues and Eigenvectors," *Biological Cybernetics*, **68** (1992) 155–164.

[9] J. Herault, C. Jutten, and B. Ans, "Detection de Grandeurs Primitives dans un Message Composite par une Architecture de Calcul Neuromimetique en Apprentissage non Supervise," *Actes du X'eme Colloque GRETSI*, Nice, **2** (1985) 1017–1022.

[10] C. Jutten and J. Herault J, "Blind Separation of Sources, Part I: An Algorithm Based on a Neuromimetic Architecture," *Signal Processing*, **24** (1991) 1–29.

[11] S. Osowski, "Signal Flow Graph Approach to the Learning Rule of Neural Networks," *Workshop on Massively Parallel Computations*, Leysin, (March 1992) 13–24.

[12] D. Hush and B. Horne, "Progress in Supervised Neural Networks," *IEEE Signal Processing Magazine*, **1** (1993) 8–39.

[13] Y. Le Cun, J. Denker, and S. Solla, "Optimal Brain Damage," pages 598–605 in *Advances in Neural Information Processing Systems 2*, edited by D. Touretzky (San Mateo, CA: Morgan Kauffman, 1990).

[14] B. Hassibi and D. Stork, "Second Order Derivatives for Network Pruning: Optimal Brain Surgeon," in *Advances in Neural Information Processing Systems 5* (San Mateo, CA: Morgan Kauffman, 1993).

[15] T. Coleman and J. More, "Estimation of Sparse Hessian Matrices and Graph Coloring Problems," *Mathematical Programming*, **228** (1984) 243–270.

[16] M. Bartholomew-Biggs, "The Estimation of the Hessian Matrix in Nonlinear Least Squares Problems with Nonzero Residuals," *Mathematical Programming*, **12** (1977) 67–80.

[17] T. McCormick, "Optimal Approximation of Sparse Hessian and Its Equivalence to a Graph Coloring Problem," *Mathematical Programming*, **26** (1983) 153–171.

[18] C. Bishop, "Exact Calculation of the Hessian Matrix for the Multilayer Perceptron," *Neural Computation*, **4** (1992) 494–501.