Yugoslav Journal of Operations Research
8 (1998), Number 1, 75-92

Jovan Petrić memorial volume

A GENERAL NUMERICAL APPROACH TO SENSITIVITY ANALYSIS AND ERROR ANALYSIS WITH ADJOINT SYSTEMS

Masao IRI, Koichi KUBOTA
Department of Information and System Engineering,
Faculty of Science and Engineering, Chuo University,
Kasuga 1-13-27, Bunkyo-ku, Tokyo 112-8551, Japan

Dedicated to the late Professor Jovan Petrić

Abstract: The concept of *adjoint system* (sometimes, also called *conjugate system*, *transposed system*, etc.) has been defined and used in different ways in different fields of science and engineering for different purposes, and the theories and techniques in different fields have been developed more or less independently. In the present paper it will be illustrated that the concept can be somehow related to the numerical-computational technique of fast automatic differentiation in any field, and that its efficiency in theory and application can be better understood from the numerical-computational viewpoint.

Keywords: System analysis, adjoint system, sensitivity, error analysis, automatic differentiation

1. INTRODUCTION

The concept of *adjoint system* (sometimes, also called *conjugate system*, *transposed system*, etc.) has been defined and used in different ways in different fields of science and engineering for different purposes, i.e., for sensitivity analysis, for noise

analysis, for optimization, etc. However, the theories and techniques in different fields have been developed more or less independently, and it is not very easy to see how they are related to one another [1, 2, 6].

In the present paper it will be illustrated that the concept can be somehow related to the numerical-computational technique of fast automatic differentiation in any field. Moreover, its efficiency in theory and application can be better understood from the numerical-computational viewpoint than through the traditional physical-theoretical construction. For example, it is sometimes said that, for sensitivity analysis, we have to solve an adjoint system again from scratch after solving the original system [6], but, it is not really the case.

Moreover, in *practical* situations, we cannot but use the floating-point arithmetic operations in numerical system analysis, so that accumulation of rounding errors is inevitable. The adjoint system plays an important role in rounding error analysis, especially for implicit functions, which will be discussed in the following.

By the way, it should be noted that a lot of derivatives which appear in the following sections can be computed by means of automatic differentiation [4, 7, 8, 9, 17, 19] and that a number of *practical* tools for automatic differentiation are now available in the world [4, 18, 20, 22]. This paper is based on the series of papers [10, 11, 12, 13, 14] which the authors have been publishing for these five years.

2. A GENERAL FORMULATION OF NONLINEAR SYSTEMS

A nonlinear system may be described in a general setting as follows.

(a) The system has n *intensive variables* (or *through-variables*; such as electric currents, forces, moments, etc.) $\boldsymbol{x} = [\, x_1, x_2, \ldots, x_n \,]^T$, n *extensive variables* (or *across-variables*; such as voltages, displacements, bendings, etc.) $\boldsymbol{y} = [\, y_1, y_2, \ldots, y_n \,]^T$ which are conjugate or contragradient to $\boldsymbol{x}$, p variables $\boldsymbol{u} = [\, u_1, u_2, \ldots, u_p \,]^T$ representing *input* or *applied* quantities to the system and p' variables $\boldsymbol{v} = [\, v_1, v_2, \ldots, v_{p'} \,]^T$ representing the *responses* ($\boldsymbol{u}$ and $\boldsymbol{v}$ may not be conjugate or contragradient to each other).

(b) These $2n + p + p'$ variables $\boldsymbol{u}, \boldsymbol{v}, \boldsymbol{x}, \boldsymbol{y}$ satisfy a set of relations usually linear and called *structural equations* (such as Kirchihoff's current law and voltage law, equilibrium equations for forces and moments, compatibility conditions for deformations):

$$Q_1 \boldsymbol{u} + Q_2 \boldsymbol{v} + Q_3 \boldsymbol{x} + Q_4 \boldsymbol{y} = \boldsymbol{0}\,. \tag{1}$$

Each equation in (1) is a relation among intensive variables only or extensive variables only and variables of different kinds never appear in the same equation. In general, the rank of the structural equations is equal to n.

(c) The $2n$ variables $\boldsymbol{x}$ and $\boldsymbol{y}$ satisfy the relations generally nonlinear and usually called *constitutive equations*:

$$\boldsymbol{F}(\boldsymbol{x}, \boldsymbol{y}, \alpha) = \boldsymbol{0} \tag{2}$$

which represent the physical characteristics of elements of the system. The rank of (the Jacobian with respect to x and y of) $\boldsymbol{F}$ is usually equal to n, and, $\boldsymbol{F}$ is very sparse in most engineering systems, i.e., each equation represents a relation among a very few variables. The third argument $\alpha = [\ \alpha_1, \ldots, \alpha_q\ ]$ indicates parameters describing the characteristics of the elements.

3. SYSTEM ANALYSIS

Analyzing the system formulated in the preceding section means determining the (generally nonlinear) relation

$$\boldsymbol{v} = C(\boldsymbol{u}, \alpha) \tag{3}$$

between the input $\boldsymbol{u}$ and the response $\boldsymbol{v}$ by eliminating "internal" variables $\boldsymbol{x}$ and $\boldsymbol{y}$ using the structural equations (1) and the constitutive equations (2). Ordinarily, we compute the value of $\boldsymbol{v}$ from the given value of $\boldsymbol{u}$ with some numerical method, e.g., the Newton-Raphson method. For the existence of a unique solution $\boldsymbol{v}$ corresponding to the given $\boldsymbol{u}$, we have to assume that the matrix

$$\Phi = \begin{bmatrix} Q_2 & Q_3 & Q_4 \\ O & F_x & F_y \end{bmatrix} \tag{4}$$

is invertible, i.e., $\operatorname{rank} \Phi = 2n + p'$, in some neighbourhood of $\boldsymbol{x}$, $\boldsymbol{y}$ and α concerned.

During the Newton-Raphson iteration,

$$\begin{bmatrix} \boldsymbol{v}_{k+1} \\ \boldsymbol{x}_{k+1} \\ \boldsymbol{y}_{k+1} \end{bmatrix} = \begin{bmatrix} \boldsymbol{v}_k \\ \boldsymbol{x}_k \\ \boldsymbol{y}_k \end{bmatrix} - U^{-1} L^{-1} \begin{bmatrix} Q_1 \boldsymbol{u}_k + Q_2 \boldsymbol{v}_k + Q_3 \boldsymbol{x}_k + Q_4 \boldsymbol{y}_k \\ F(\boldsymbol{x}_k, \boldsymbol{y}_k, \alpha) \end{bmatrix} \tag{5}$$

$(k = 0, 1, 2, \ldots)$, to get the numerical solution of $\boldsymbol{v}$ starting from initial guesses $\boldsymbol{v}_0, \boldsymbol{x}_0$ and $\boldsymbol{y}_0$, we usually make use of the LU-factorization of Φ :

$$\Phi = LU \ , \tag{6}$$

where L and U are the lower triangular matrix and the upper triangular matrix, at each step of iteration. Furthermore, we have to choose an appropriate tolerance for terminating the iteration, where it is recommended in [8, 17] to make use of the information obtained from the estimates of rounding errors accumulated in the computed values of the components of (1) and (2). Note that most of the computational efforts are usually devoted to the factorization $\Phi = LU$.

4. SENSITIVITY ANALYSIS

Now we consider two kinds of sensitivities:

$$\frac{\partial \boldsymbol{v}}{\partial \boldsymbol{\alpha}} = C_{\alpha}(\boldsymbol{u}, \boldsymbol{\alpha}) \tag{7}$$

and

$$\frac{\partial}{\partial \boldsymbol{\alpha}}\left(\frac{\partial \boldsymbol{v}}{\partial \boldsymbol{u}}\right) = \frac{\partial}{\partial \boldsymbol{\alpha}} C_{u}(\boldsymbol{u}, \boldsymbol{\alpha}) \tag{8}$$

of the system performance with respect to the parameters $\boldsymbol{\alpha}$ defining the characteristics of elements.

The former represents how much the responses $\boldsymbol{v}$ vary when the parameters $\boldsymbol{\alpha}$ are perturbed with the inputs $\boldsymbol{u}$ fixed constant, whereas the latter represents how the tangent linear relations $\partial \boldsymbol{v}/\partial \boldsymbol{u}$ between the inputs $\boldsymbol{u}$ and the responses $\boldsymbol{v}$ depend on the perturbation of the parameters $\boldsymbol{\alpha}$. The two essentially coincide with each other for linear systems since

$$C_{\alpha}(\boldsymbol{u}, \boldsymbol{\alpha}) = \frac{\partial C_{u}(\boldsymbol{u}, \boldsymbol{\alpha})}{\partial \boldsymbol{\alpha}} \cdot \boldsymbol{u} \ . \tag{9}$$

After solving the system for given inputs $\boldsymbol{u}$, i.e., after determining the numerical values $\boldsymbol{v}$, $\boldsymbol{x}$, $\boldsymbol{y}$ corresponding to given $\boldsymbol{u}$ and $\Phi = LU$, we compute the following equations by differentiating (1) and (2) with respect to $\boldsymbol{\alpha}$:

$$\begin{bmatrix} Q_2 \dfrac{\partial \boldsymbol{v}}{\partial \boldsymbol{\alpha}} + Q_3 \dfrac{\partial \boldsymbol{x}}{\partial \boldsymbol{\alpha}} + Q_4 \dfrac{\partial \boldsymbol{y}}{\partial \boldsymbol{\alpha}} \\ F_x \dfrac{\partial \boldsymbol{x}}{\partial \boldsymbol{\alpha}} + F_y \dfrac{\partial \boldsymbol{y}}{\partial \boldsymbol{\alpha}} \end{bmatrix} = \begin{bmatrix} O_{n+p',q} \\ -F_{\alpha} \end{bmatrix} \tag{10}$$

or

$$\begin{bmatrix} \dfrac{\partial \boldsymbol{v}}{\partial \boldsymbol{\alpha}} \\ \dfrac{\partial \boldsymbol{x}}{\partial \boldsymbol{\alpha}} \\ \dfrac{\partial \boldsymbol{y}}{\partial \boldsymbol{\alpha}} \end{bmatrix} = U^{-1}L^{-1}\begin{bmatrix} O_{n+p',q} \\ -F_{\alpha} \end{bmatrix} \tag{11}$$

to obtain

$$\frac{\partial \boldsymbol{v}}{\partial \boldsymbol{\alpha}} = -\Psi F_{\alpha}\,, \tag{12}$$

where

$$\Psi = [\ I_{p'}, O_{p',n}, O_{p',n}\ ]U^{-1}L^{-1}\begin{bmatrix} O_{n+p',n} \\ I_n \end{bmatrix}. \tag{13}$$

Here it is worth noting that, in (12) and (13), the order of multiplication of relevant matrices is crucial. In ordinary situations, p' is far smaller than n, so that the computation of the product $[\ I_{p'}, O_{p',n}, O_{p',n}\ ]U^{-1}L^{-1}$ must be performed *from left to right*, the remaining multiplication with $\begin{bmatrix} O_{n+p',n} \\ I_n \end{bmatrix}$ and F_{α} being substantially to pick up relevant elements of F_{α} to multiply with the corresponding elements of that product. (In actual computation, the multiplication by U^{-1} or L^{-1} is performed by solving the corresponding triangular systems of linear equations.)

Furthermore, it may be of interest to note that the right-hand side of (12) can be interpreted as the response of the perturbation linear system for the input $-F_{\alpha}$ so that the sensitivity $\partial \boldsymbol{v}/\partial \boldsymbol{\alpha}$ can be computed by solving that linear system, and that this is exactly the naive approach to sensitivity which is traditionally called "compensation theorem" in electric circuit theory. There we compute $\Phi^{-1}\begin{bmatrix} O_{n+p',n} \\ I_n \end{bmatrix}(=-F_{\alpha})$ first, and then pick up necessary components according to $[\ I_{p'}, O_{p',n}, O_{p',n}\ ]$, or, in terms of matrix multiplication, perform the multiplication *from right to left*.

For $\partial \boldsymbol{v}/\partial \boldsymbol{u}$, differentiation of (1) and (2) with respect to $\boldsymbol{u}$:

$$\begin{bmatrix} Q_2 \dfrac{\partial \boldsymbol{v}}{\partial \boldsymbol{u}} + Q_3 \dfrac{\partial \boldsymbol{x}}{\partial \boldsymbol{u}} + Q_4 \dfrac{\partial \boldsymbol{y}}{\partial \boldsymbol{u}} \\ F_x \dfrac{\partial \boldsymbol{x}}{\partial \boldsymbol{u}} + F_y \dfrac{\partial \boldsymbol{y}}{\partial \boldsymbol{u}} \end{bmatrix} = \begin{bmatrix} -Q_1 \\ O_{n,p} \end{bmatrix} \tag{14}$$

yields the expression:

$$\begin{bmatrix} \dfrac{\partial \boldsymbol{v}}{\partial \boldsymbol{u}} \\ \dfrac{\partial \boldsymbol{x}}{\partial \boldsymbol{u}} \\ \dfrac{\partial \boldsymbol{y}}{\partial \boldsymbol{u}} \end{bmatrix} = \Phi^{-1} \begin{bmatrix} -Q_1 \\ O_{n,p} \end{bmatrix}. \tag{15}$$

Then using the known general relation on the differentials of a matrix $A(t) = [\ a_{ij}(t)\ ]$ and its inverse A^{-1}:

$$\mathrm{d}A^{-1} = -A^{-1} \cdot \mathrm{d}A \cdot A^{-1}, \tag{16}$$

we obtain

$$\frac{\partial}{\partial \boldsymbol{\alpha}} \left(\frac{\partial \boldsymbol{v}}{\partial \boldsymbol{u}} \right) = [\ I_{p'}, O_{p',n}, O_{p',n}\ ] \Phi^{-1} \frac{\partial \Phi}{\partial \boldsymbol{\alpha}} \Phi^{-1} \begin{bmatrix} Q_1 \\ O_{n,p} \end{bmatrix}. \tag{17}$$

The expression (17) resembles (12) and (13) but differs from the latter in that $\partial \Phi / \partial \boldsymbol{\alpha}$ contains apparently higher-order tensors $\partial^2 F / \partial \boldsymbol{x} \partial \boldsymbol{\alpha}$ and $\partial^2 F / \partial \boldsymbol{y} \partial \boldsymbol{\alpha}$. However, those higher-order tensors are usually very sparse, so that the computation of (17) requires no substantial extra works if $\Phi^{-1} = U^{-1} L^{-1}$, which has already been computed, is available. Needless to say, in actual computation, we compute the product $[\ I_{p'}, O_{p',n}, O_{p',n}\ ] \Phi^{-1}$ by multiplication *from left to right* and $\Phi^{-1} \begin{bmatrix} Q_1 \\ O_{n,p} \end{bmatrix}$ *from right to left*. As will be clarified through comparison with the traditional approach in the following section, this procedure can be regarded as the numerical-computational approach to the essential part of the system sensitivity analysis, i.e., "solving the perturbation linear system (of the original system)" (the latter part) and then "solving the adjoint system" (the former part).

The computational procedures in the above will look very natural once the problem has been so formulated, but it is important to see the traditional approach naturally realized and extended to a more general setting by looking at the problem from the viewpoint of numerical computation. Furthermore, it should be noted that we

are led to essentially the same computational procedures if we "fast automatically differentiate" [8, 9] with respect to parameters α the procedure of analyzing original system making use of the technique of accelerating the part of matrix inversion [21].

From the viewpoint of computational complexity, important is to note that, once the original system has been solved, the extra amount of computation needed for the sensitivity analysis is of lower order (i.e., the LU-factorization of Φ is not necessary any more).

5. RECAPITULATION OF THE TRADITIONAL APPROACH TO SENSITIVITY ANALYSIS BY USING THE ADJOINT SYSTEM

In this section, we shall recapitulate the simplest case of traditional approach to sensitivity analysis by means of the adjoint system according basically to [6].

We shall confine ourselves to the case where $\boldsymbol{u}$ and $\boldsymbol{v}$ are contragradient quantities of each other without loss of generality ($p' = p$). There is a theorem called Tellegen's theorem (easy to prove) stating that, when $\{ \boldsymbol{u}, \boldsymbol{v}, \boldsymbol{x}, \boldsymbol{y} \}$ and $\{ \hat{\boldsymbol{u}}, \hat{\boldsymbol{v}}, \hat{\boldsymbol{x}}, \hat{\boldsymbol{y}} \}$ satisfy the structural equations independently of each other, we have

$$\sum_{i=1}^{p} \hat{u}_i v_i + \sum_{\kappa=1}^{n} \hat{x}_\kappa y_\kappa = 0 , \tag{18}$$

$$\sum_{i=1}^{p} \hat{v}_i u_i + \sum_{\kappa=1}^{n} \hat{y}_\kappa x_\kappa = 0 . \tag{19}$$

The adjoint system $\{ \hat{\boldsymbol{u}}, \hat{\boldsymbol{v}}, \hat{\boldsymbol{x}}, \hat{\boldsymbol{y}} \}$ of the original $\{ \boldsymbol{u}, \boldsymbol{v}, \boldsymbol{x}, \boldsymbol{y} \}$ is defined by the following (i) and (ii).

(i) $\{ \hat{\boldsymbol{u}}, \hat{\boldsymbol{v}}, \hat{\boldsymbol{x}}, \hat{\boldsymbol{y}} \}$ satisfy the same structural equations as that for the original system:

$$Q_1 \hat{\boldsymbol{u}} + Q_2 \hat{\boldsymbol{v}} + Q_3 \hat{\boldsymbol{x}} + Q_4 \hat{\boldsymbol{y}} = \boldsymbol{0} . \tag{20}$$

(ii) $\{ \hat{\boldsymbol{u}}, \hat{\boldsymbol{v}}, \hat{\boldsymbol{x}}, \hat{\boldsymbol{y}} \}$ satisfy the constitutive equations:

$$\sum_{\kappa=1}^{n} \hat{x}_\kappa \cdot \Delta y_\kappa - \sum_{\kappa=1}^{n} \hat{y}_\kappa \cdot \Delta x_\kappa = 0 , \tag{21}$$

which should hold valid for arbitrary perturbations Δy_κ and Δx_κ of the original constitutive equations with α fixed. Note here that the constitutive equations of the adjoint system depend in general on $\boldsymbol{u}$ if the original system is nonlinear.

In the following, we assume that there is a partition (K_1, K_2) of the index set $K = \{ 1, 2, \ldots, n \}$ of $\boldsymbol{x}$ and $\boldsymbol{y}$ such that the F in the constitutive equations (2) satisfies

$$\text{rank} \left[F_x|_{K_1}, F_y|_{K_2} \right] = n , \tag{22}$$

and that we have an explicit representation:

$$x_\kappa = X_\kappa(\{ y_\lambda \mid \lambda \in K_1 \}, \{ x_\mu \mid \mu \in K_2 \}) \quad (\kappa \in K_1), \tag{23}$$

$$y_\lambda = Y_\lambda(\{ y_\nu \mid \nu \in K_1 \}, \{ x_\kappa \mid \kappa \in K_2 \}) \quad (\lambda \in K_2). \tag{24}$$

For the sake of simplicity we assume that $K_1 = \emptyset$ and $K_2 = K$, i.e.,

$$y_\lambda = Y_\lambda(x_1, \ldots, x_n) \quad (\lambda = 1, \ldots, n). \tag{25}$$

(If we would not avoid complication of notation, the same arguments hold valid for the more general case.)

Let the solution for given $\boldsymbol{u}$ and $\boldsymbol{\alpha}$ be denoted by $\{ \boldsymbol{v}, \boldsymbol{x}, \boldsymbol{y} \}$, and the solution for another $\boldsymbol{\alpha} + \Delta\boldsymbol{\alpha}$ $(\Delta\mathbf{u} = 0)$ be denoted by $\{ \boldsymbol{v} + \Delta\boldsymbol{v}, \boldsymbol{x} + \Delta\boldsymbol{x}, \boldsymbol{y} + \Delta\boldsymbol{y} \}$. Then we have

$$\left.\begin{aligned} \sum_{i=1}^{p} \hat{u}_i \Delta v_i + \sum_{\kappa=1}^{n} \hat{x}_\kappa \Delta y_\kappa = 0, \\ \sum_{i=1}^{p} \hat{v}_i \Delta u_i + \sum_{\kappa=1}^{n} \hat{y}_\kappa \Delta x_\kappa = 0 \end{aligned}\right\} \tag{26}$$

for the structural equations, since $\{ \Delta\boldsymbol{u}, \Delta\boldsymbol{v}, \Delta\boldsymbol{x}, \Delta\boldsymbol{y} \}$ also satisfy (18), (19) and (20).

For the constitutive equations, using the relation $y_\kappa = Y_\kappa(x_\lambda \mid \lambda \in K_2)$, we have

$$\Delta y_\kappa = \sum_{\lambda=1}^{n} \frac{\partial Y_\kappa}{\partial x_\lambda} \Delta x_\lambda + \sum_{j=1}^{q} \frac{\partial Y_\kappa}{\partial \alpha_j} \Delta \alpha_j . \tag{27}$$

When $\Delta\alpha_j = 0$, we obtain

$$\sum_{\kappa=1}^{n} \hat{x}_\kappa \Delta y_\kappa - \sum_{\kappa=1}^{n} \hat{y}_\kappa \Delta x_\kappa = \sum_{\lambda=1}^{n} (\sum_{\kappa=1}^{n} \hat{x}_\kappa \frac{\partial Y_\kappa}{\partial x_\lambda} - \hat{y}_\lambda) \Delta x_\lambda = 0 . \tag{28}$$

Since this should hold for any Δx_λ, the constitutive equations of the adjoint system can be represented as

$$\hat{y}_\lambda = \sum_{\kappa=1}^{n} \hat{x}_\kappa \frac{\partial Y_\kappa}{\partial x_\lambda} \quad (\lambda = 1, \ldots, n) . \tag{29}$$

Using (26) and $\Delta u_i = 0$, and solving the adjoint system (29) once, we have

$$\sum_{i=1}^{p} \hat{u}_i \Delta v_i + \sum_{\kappa=1}^{n} \hat{x}_\kappa \Delta y_\kappa - \sum_{\kappa=1}^{n} \hat{y}_\kappa \Delta x_\kappa$$

$$= \sum_{i=1}^{p} \hat{u}_i \Delta v_i + \sum_{\lambda=1}^{n} \Big(\sum_{\kappa=1}^{n} \hat{x}_\kappa \frac{\partial Y_\kappa}{\partial x_\lambda} - \hat{y}_\lambda \Big) \Delta x_\lambda + \sum_{j=1}^{q} \sum_{\kappa=1}^{n} \hat{x}_\kappa \frac{\partial Y_\kappa}{\partial \alpha_j} \Delta \alpha_j \tag{30}$$

$$= \sum_{i=1}^{p} \hat{u}_i \Delta v_i + \sum_{j=1}^{q} \Big(\sum_{\kappa=1}^{n} \hat{x}_\kappa \frac{\partial Y_\kappa}{\partial \alpha_j} \Big) \Delta \alpha_j = 0, \tag{31}$$

since the second term of (30) is equal to zero. Thus we obtain

$$\sum_{i=1}^{p} \hat{u}_i \Delta v_i = - \sum_{j=1}^{q} \Big(\sum_{\kappa=1}^{n} \hat{x}_\kappa \frac{\partial Y_\kappa}{\partial \alpha_j} \Big) \Delta \alpha_j . \tag{32}$$

In order to compute $\partial v_i / \partial \alpha_j$ according to the traditional method, we may solve the adjoint system $\{ \hat{\boldsymbol{v}}, \hat{\boldsymbol{x}}, \hat{\boldsymbol{y}} \}$ with the conditions

$$\hat{u}_i = -1 , \ \hat{u}_k = 0 \quad (k \neq i) , \tag{33}$$

and then, with $\hat{\boldsymbol{x}}$ thus obtained, compute

$$\frac{\partial v_i}{\partial \alpha_j} = \sum_{\kappa=1}^{n} \hat{x}_\kappa \frac{\partial Y_\kappa}{\partial \alpha_j} . \tag{34}$$

The above arguments make the essence of the traditional approach to sensitivity analysis by means of the adjoint system. It will be easy to see that this is a special case of our general numerical approach developed in the previous sections, and that, contrarily to the tradition, we need not "solve the adjoint system anew" but can make use of the LU-factorization which we have had during the solution of the original system.

6. ERROR ANALYSIS BY MEANS OF THE ADJOINT SYSTEM

6.1. Error analysis for an implicit function

First, let us consider the simplest case of a scalar implicit function z of w defined by

$$g(z,w) = 0\,, \tag{35}$$

where g is continuously differentiable with respect to the two variables z and w. We assume that we have an explicit procedure to compute the value of g for given values of z and w and that we are to compute the value of z corresponding to a given value of w by means of the Newton-Raphson method. We will denote with an overline those values of variables which we actually have by finite-precision computations, e.g., $\overline{w}$ and $\overline{z}$.

If we were able to infinitely iterate the Newton-Raphson method with the infinite-precision arithmetic operations, we could compute z that satisfies $g(z,\overline{w}) = 0$. But in actual circumstances, we have to take into consideration the following two types of errors with finite-precision arithmetic operations. One is the rounding error δ_g accumulated in the value of g and the other is the truncation errors due to terminating the Newton-Raphson iteration after a finite number of steps. What we know at the end of the Newton-Raphson iteration is only the value $\overline{\eta}$ of the right-hand side of

$$g(\overline{z},\overline{w}) + \delta_g = \overline{\eta}\,, \tag{36}$$

where the termination criterion is $\left|\,\overline{\eta}\,\right| \le \overline{\varepsilon}$ with a given tolerance $\overline{\varepsilon}$.

Now, in order to evaluate δ_g, we represent the process of the computation of the function g as a sequence of equations called the *computational process* [8, 24, 25]:

$$\begin{aligned} v_1 &= \varphi_1(u_{11},u_{12}) \\ &\vdots \\ v_k &= \varphi_k(u_{k1},u_{k2}) \\ &\vdots \\ g(z,w) = v_r &= \varphi_r(u_{r1},u_{r2}) \end{aligned}$$

where v_k is an intermediate variable that keeps the intermediate result, φ_k is one of the *basic operations* (which may be either unary or binary), and u_{k1} and u_{k2} are formal parameters, the actual parameter corresponding to each of which is one of constants, the input w and the intermediate results $v_1,\ldots,v_{k-1}$ $(k=1,\ldots,r)$. If we add perturbations $\delta_1,\ldots,\delta_r$ to the intermediate results, the final result may be a function of the perturbations, which we will denote by $G(z,w;\delta_1,\ldots,\delta_r)$:

$$
\begin{aligned}
v_1 &= \varphi_1(u_{11},u_{12}) + \delta_1 \\
&\vdots \\
v_k &= \varphi_k(u_{k1},u_{k2}) + \delta_k \\
&\vdots \\
G(z,w;\delta_1,\ldots,\delta_r) \equiv v_r &= \varphi_r(u_{r1},u_{r2}) + \delta_r .
\end{aligned}
$$

With finite-precision operations, these perturbations are caused by the *rounding errors* at executions of basic operations, i.e., $\delta_1,\ldots,\delta_r$ are the rounding errors locally generated at computation of $\varphi_1,\ldots,\varphi_r$. Thus, by virtue of the mean value theorem, we have

$$\delta_g = G(z,w;\delta_1,\ldots,\delta_r) - G(z,w;0,\ldots,0) \tag{37}$$

$$= \sum_{k=1}^{r} \frac{\partial G}{\partial \delta_k}(z,w;\xi_1,\ldots,\xi_r)\cdot\delta_k , \tag{38}$$

for δ_g in (36), where $\xi_k = \theta\delta_k$ $(0<\theta<1;\ k=1,\ldots,r)$. The magnitudes of rounding errors δ_k's themselves are unknown, but we can estimate the upper bound of $|\ \delta_k\ |$ as $|\ v_k\ |\cdot\varepsilon_M$ according to the IEEE standard arithmetics, where ε_M is the so-called machine epsilon, i.e., we may set $|\ \delta_k\ | \le |\ v_k\ |\cdot\varepsilon_M$. Moreover, if we resort to interval arithmetics, we can determine intervals that contain $\partial G/\partial\delta_k$'s and δ_k's, to get a rigorous upper bound of δ_g. Note that $\partial G/\partial\delta_k$ (either as a "point" or as an "interval") can be computed straightforward with fast automatic differentiation (or reverse mode automatic differentiation) [17, 23, 25]. In many real engineering applications, an approximate upper bound or a probabilistic estimation (when *round to the nearest* is employed) of δ_g is sufficiently useful:

$$\Delta_{\mathrm{A}}[\ g\ ] \equiv \Big(\sum_{k=1}^{r} \Big|\ \frac{\partial G}{\partial\delta_k}\cdot v_k\ \Big|\Big)\cdot\varepsilon_M \quad\text{or}\quad \Delta_{\mathrm{P}}[\ g\ ] \equiv \sqrt{\frac{1}{3}\sum_{k=1}^{r}\Big(\frac{\partial G}{\partial\delta_k}\cdot v_k\Big)^2}\cdot\varepsilon_M , \tag{39}$$

with $\partial G/\partial\delta_k\,(z,w;0,\ldots,0)$ as $\partial G/\partial\delta_k$ $(k=1,\ldots,r)$ [8, 9, 17]. Using these estimates, we have

$$|\ \delta_g\ | \le \Delta_{\mathrm{A}}[\ g\ ] \quad\text{or}\quad |\ \delta_g\ | \cong \Delta_{\mathrm{P}}[\ g\ ]. \tag{40}$$

Let us proceed to the evaluation of the accumulated error in the computed value $\bar{z}$ corresponding to a given $\bar{w}$ defined by (35). Subtracting (35) from (36), we have

$$g(\bar{z},\bar{w}) - g(z,w) = \bar{\eta} - \delta_g . \tag{41}$$

Since $\overline{z}$, $\overline{w}$ and $\overline{\eta}$ are computed values, and an interval containing $\overline{w} - w$ as well as that containing δ_g can be estimated, an interval containing z may be determined by means of an established method of interval computation [3].

In the case of vector-valued function $\boldsymbol{g} : \mathbf{R}^{m+n} \to \mathbf{R}^m$, $\boldsymbol{z} \in \mathbf{R}^m$ is implicitly defined by

$$\boldsymbol{g}(\boldsymbol{z}, \boldsymbol{w}) = \mathbf{0} \tag{42}$$

to be a function of $\boldsymbol{w} \in \mathbf{R}^n$, where we assume that $\boldsymbol{g}$ is smooth enough and $\partial \boldsymbol{g}/\partial \boldsymbol{z}$ is non-singular.

On termination of the Newton-Raphson iteration, say, we have

$$\boldsymbol{g}(\overline{\boldsymbol{z}}, \overline{\boldsymbol{w}}) + \delta_g = \overline{\eta}\ , \tag{43}$$

where $\overline{\eta}$ is the computed vector of residuals and δ_g indicates a vector whose components are the rounding errors in the corresponding components of the computed value of g. For the ith component, we have

$$g_i(\overline{\boldsymbol{z}}, \overline{\boldsymbol{w}}) - g_i(\boldsymbol{z}, \boldsymbol{w}) = (\overline{\eta})_i - (\delta_g)_i \tag{44}$$

and

$$\frac{\partial g_i}{\partial \boldsymbol{z}}(\xi_z^i, \xi_w^i)(\overline{\boldsymbol{z}} - \boldsymbol{z}) + \frac{\partial g_i}{\partial \boldsymbol{w}}(\xi_z^i, \xi_w^i)(\overline{\boldsymbol{w}} - \boldsymbol{w}) = (\overline{\eta})_i - (\delta_g)_i\ , \tag{45}$$

where $\xi_z^i = (1 - \theta_i)\overline{\boldsymbol{z}} - \theta_i \boldsymbol{z}$, $\xi_w^i = (1 - \theta_i)\overline{\boldsymbol{w}} - \theta_i \boldsymbol{w}$ and $0 < \theta_i < 1$ for each i. Defining

$$\tilde{J}_z \equiv \left[\frac{\partial g_i}{\partial z_j}(\xi_z^i, \xi_w^i)\right] \quad \text{and} \quad \tilde{J}_w \equiv \left[\frac{\partial g_i}{\partial w_j}(\xi_z^i, \xi_w^i)\right], \tag{46}$$

we have

$$\overline{\boldsymbol{z}} - \boldsymbol{z} = \tilde{J}_z^{-1}(\overline{\eta} - \delta_g - \tilde{J}_w(\overline{\boldsymbol{w}} - \boldsymbol{w})) \tag{47}$$

and we can express the rounding error δ_g in terms of function $\boldsymbol{G}\ (= (G_i))$:

$$(\delta_g)_i = \sum_{k=1}^{r} \frac{\partial G_i}{\partial \delta_k}(\boldsymbol{z}, \boldsymbol{w}; \xi_1, \ldots, \xi_r) \cdot \delta_k\ , \tag{48}$$

which is defined by the computational process for $\boldsymbol{g}$:

$$v_1 = \varphi_1(u_{11}, u_{12}) + \delta_1$$
$$\vdots$$
$$v_k = \varphi_k(u_{k1}, u_{k2}) + \delta_k$$
$$\vdots$$
$$G_1(z, w; \delta_1, \ldots, \delta_r) \equiv v_{r-m+1} = \varphi_{r-m+1}(u_{r-m+1,1}, u_{r-m+1,2}) + \delta_{r-m+1}$$
$$\vdots$$
$$G_m(z, w; \delta_1, \ldots, \delta_r) \equiv v_r = \varphi_r(u_{r1}, u_{r2}) + \delta_r ,$$

$\xi_1, \ldots, \xi_r$ being as defined in (38), i.e., we may assume that the last m intermediate results in the process correspond to the components of the computed value of $\boldsymbol{g}$. Denoting by $\boldsymbol{e}_i$ the ith unit vector, we have

$$(\bar{\boldsymbol{z}} - \boldsymbol{z})_i = \boldsymbol{e}_i^T (\bar{\boldsymbol{z}} - \boldsymbol{z}) = \boldsymbol{e}_i^T \tilde{J}_z^{-1} (\bar{\eta} - \delta_g - \tilde{J}_w (\bar{\boldsymbol{w}} - \boldsymbol{w})) \tag{49}$$
$$= \boldsymbol{b}^T (\bar{\eta} - \tilde{J}_w (\bar{\boldsymbol{w}} - \boldsymbol{w})) - \boldsymbol{b}^T \delta_g , \tag{50}$$

where $\boldsymbol{b}^T = \boldsymbol{e}_i^T \tilde{J}_z^{-1}$. The last term of (50), $\boldsymbol{b}^T \delta_g$, corresponds to the expansion of errors of a scalar-valued function $h(\boldsymbol{z}, \boldsymbol{w}) \equiv \boldsymbol{b}^T \boldsymbol{g}(\boldsymbol{z}, \boldsymbol{w})$. Consequently we can compute $\boldsymbol{b}^T \delta_g$ by applying fast automatic differentiation (or reverse mode automatic differentiation) to the function h. (Note that we can replace $\boldsymbol{e}_i$ with arbitrary constant vector $\boldsymbol{c}$ for expanding $\boldsymbol{c}^T (\bar{\boldsymbol{z}} - \boldsymbol{z})$.)

For example, let us consider the implicit function $\boldsymbol{v}$, $\boldsymbol{x}$, $\boldsymbol{y}$ of $\boldsymbol{u}$ in the system described by (1) and (2):

$$\boldsymbol{g}(\boldsymbol{u}, \boldsymbol{v}, \boldsymbol{x}, \boldsymbol{y}) \equiv \begin{bmatrix} Q_1 \boldsymbol{u} + Q_2 \boldsymbol{v} + Q_3 \boldsymbol{x} + Q_4 \boldsymbol{y} \\ \boldsymbol{F}(\boldsymbol{x}, \boldsymbol{y}, \alpha) \end{bmatrix} = \boldsymbol{0} . \tag{51}$$

In order to solve the system described by (51), we must compute approximate values $\bar{\boldsymbol{v}}, \bar{\boldsymbol{x}}, \bar{\boldsymbol{y}}$ numerically with a given value $\bar{\boldsymbol{u}}$. At the final step of the Newton-Raphson iterations, we have

$$Q_1 \bar{\boldsymbol{u}} + Q_2 \bar{\boldsymbol{v}} + Q_3 \bar{\boldsymbol{x}} + Q_4 \bar{\boldsymbol{y}} + \delta_1 = \bar{\eta}_1 , \tag{52}$$
$$\boldsymbol{F}(\bar{\boldsymbol{x}}, \bar{\boldsymbol{y}}; \alpha) + \delta_2 = \bar{\eta}_2 , \tag{53}$$

where δ_1 and δ_2 denote the accumulated rounding errors, and $\bar{\eta}_1$ and $\bar{\eta}_2$ denote the rounding errors finally computed in the Newton-Raphson iteration.

With the mean value theorem, the differences between the known values $(\overline{\boldsymbol{u}}(=\boldsymbol{u}), \overline{\boldsymbol{v}}, \overline{\boldsymbol{x}}, \overline{\boldsymbol{y}}, \overline{\eta}_1$ and $\overline{\eta}_2)$ and the unknown values $(\boldsymbol{v}, \boldsymbol{x}, \boldsymbol{y}, \delta_1$ and $\delta_2)$ satisfy the following equations:

$$Q_1(\overline{\boldsymbol{u}} - \boldsymbol{u}) + Q_2(\overline{\boldsymbol{v}} - \boldsymbol{v}) + Q_3(\overline{\boldsymbol{x}} - \boldsymbol{x}) + Q_4(\overline{\boldsymbol{y}} - \boldsymbol{y}) = \overline{\eta}_1 - \delta_1, \tag{54}$$

$$\widetilde{\boldsymbol{F}}_x(\overline{\boldsymbol{x}} - \boldsymbol{x}) + \widetilde{\boldsymbol{F}}_y(\overline{\boldsymbol{y}} - \boldsymbol{y}) = \overline{\eta}_2 - \delta_2, \tag{55}$$

where

$$(\widetilde{\boldsymbol{F}}_x)_{i,j} = \frac{\partial(\boldsymbol{F})_i}{\partial x_j}(\xi_i, \zeta_i; \boldsymbol{\alpha}), \tag{56}$$

$$(\widetilde{\boldsymbol{F}}_y)_{i,j} = \frac{\partial(\boldsymbol{F})_i}{\partial y_j}(\xi_i, \zeta_i; \boldsymbol{\alpha}), \tag{57}$$

$$\xi_i = (1 - \theta_i)\overline{\boldsymbol{x}} + \theta_i \boldsymbol{x}, \tag{58}$$

$$\zeta_i = (1 - \theta_i)\overline{\boldsymbol{y}} + \theta_i \boldsymbol{y} \tag{59}$$

$(i, j = 1, \ldots, n)$. θ_i's are also unknown except that $0 < \theta_i < 1$. Note that the above relations hold exactly (with no approximation).

Thus, defining $\widetilde{\Phi}$ as

$$\widetilde{\Phi} = \begin{bmatrix} Q_2 & Q_3 & Q_4 \\ O & \widetilde{F}_x & \widetilde{F}_y \end{bmatrix}, \tag{60}$$

we obtain

$$\begin{bmatrix} \overline{\boldsymbol{v}} - \boldsymbol{v} \\ \overline{\boldsymbol{x}} - \boldsymbol{x} \\ \overline{\boldsymbol{y}} - \boldsymbol{y} \end{bmatrix} = -\widetilde{\Phi}^{-1} \begin{bmatrix} Q_1 \\ O_{n,p} \end{bmatrix} (\overline{\boldsymbol{u}} - \boldsymbol{u}) + \widetilde{\Phi}^{-1} \begin{bmatrix} \overline{\eta}_1 - \delta_1 \\ \overline{\eta}_2 - \delta_2 \end{bmatrix}. \tag{61}$$

When we assume that there are no input errors, i.e. $\overline{\boldsymbol{u}} - \boldsymbol{u} = \boldsymbol{0}$, we obtain

$$\overline{\boldsymbol{v}} - \boldsymbol{v} = \begin{bmatrix} I_{p'}, O_{p',n}, O_{p',n} \end{bmatrix} \widetilde{\Phi}^{-1} \begin{bmatrix} \overline{\eta}_1 - \delta_1 \\ \overline{\eta}_2 - \delta_2 \end{bmatrix}. \tag{62}$$

Note that, for computation of estimates of rounding errors accumulated in the computed value of ith component $\overline{v}_i$ of $\overline{\boldsymbol{v}}$, we must expand each component of δ_1 and δ_2 according to (38). Apparently this computation looks complex but it is quite simple because it corresponds to differentiation of a scalar-valued function defined by the product of a row vector $\left(\begin{bmatrix} I_{p'}, O_{p',n}, O_{p',n} \end{bmatrix}\right)_i \widetilde{\Phi}^{-1}$ and the vector valued function $\boldsymbol{g}$ with the fast automatic differentiation (or reverse mode automatic differentiation).

6.2. Extended system

Since large-scale systems are usually constituted from many subsystems, their mathematical models are also structured accordingly. A prototype of such a system is described as the following three-step block-computation:

$$\boldsymbol{u} = \boldsymbol{\varphi}_0(\boldsymbol{w}), \tag{63}$$

$$\begin{cases} Q_1\boldsymbol{u} + Q_2\boldsymbol{v} + Q_3\boldsymbol{x} + Q_4\boldsymbol{y} = \boldsymbol{0}, \\ F(\boldsymbol{x}, \boldsymbol{y}; \alpha) = \boldsymbol{0}, \end{cases} \tag{64}$$

$$\boldsymbol{z} = \boldsymbol{\varphi}_3(\boldsymbol{v}), \tag{65}$$

where $\boldsymbol{\varphi}_0 : \mathbf{R}^{n_0} \to \mathbf{R}^{m_0}$ and $\boldsymbol{\varphi}_3 : \mathbf{R}^{n_3} \to \mathbf{R}^{m_3}$. We regard $\boldsymbol{w}$ and $\boldsymbol{z}$ as the input and the output, respectively. Note that (64) defines $\boldsymbol{v}$ implicitly.

With floating-point computation, we have to rewrite the above equation:

$$\overline{\boldsymbol{u}} = \boldsymbol{\varphi}_0(\overline{\boldsymbol{w}}) + \boldsymbol{\delta}_0, \tag{66}$$

$$\begin{cases} Q_1\overline{\boldsymbol{u}} + Q_2\overline{\boldsymbol{v}} + Q_3\overline{\boldsymbol{x}} + Q_4\overline{\boldsymbol{y}} = \overline{\boldsymbol{\eta}}_1 - \boldsymbol{\delta}_1, \\ F(\overline{\boldsymbol{x}}, \overline{\boldsymbol{y}}; \alpha) = \overline{\boldsymbol{\eta}}_2 - \boldsymbol{\delta}_2, \end{cases} \tag{67}$$

$$\overline{\boldsymbol{z}} = \boldsymbol{\varphi}_3(\overline{\boldsymbol{v}}) + \boldsymbol{\delta}_3, \tag{68}$$

where (67) indicates the final step in the Newton-Raphson iteration.

Thereby, the accumulated computational error in the final output $\overline{\boldsymbol{z}} - \boldsymbol{z}$ is expanded as follows.

$$(\overline{\boldsymbol{z}} - \boldsymbol{z})_j = (\boldsymbol{\varphi}_3(\overline{\boldsymbol{v}}) - \boldsymbol{\varphi}_3(\boldsymbol{v}))_j + (\boldsymbol{\delta}_3)_j \tag{69}$$

$$= \sum_{l=1}^{n_3} \frac{\partial(\boldsymbol{\varphi}_3)_j}{\partial v_l}(\xi_{3j}) \cdot (\overline{\boldsymbol{v}} - \boldsymbol{v})_l + (\boldsymbol{\delta}_3)_j \quad (j = 1, \ldots, m_3), \tag{70}$$

$$\overline{\boldsymbol{v}} - \boldsymbol{v} = \begin{bmatrix} I_{p'}, O, O \end{bmatrix} \tilde{\Phi}^{-1} \left(- \begin{bmatrix} Q_1 \\ O_{n,p} \end{bmatrix} (\overline{\boldsymbol{u}} - \boldsymbol{u}) + \begin{bmatrix} \overline{\boldsymbol{\eta}}_1 - \boldsymbol{\delta}_1 \\ \overline{\boldsymbol{\eta}}_2 - \boldsymbol{\delta}_2 \end{bmatrix} \right), \tag{71}$$

$$(\overline{\boldsymbol{u}} - \boldsymbol{u})_j = (\boldsymbol{\varphi}_0(\overline{\boldsymbol{w}}) - \boldsymbol{\varphi}_0(\boldsymbol{w}))_j + (\boldsymbol{\delta}_0)_j \tag{72}$$

$$= \sum_{l=1}^{n_0} \frac{\partial(\boldsymbol{\varphi}_0)_j}{\partial w_l}(\xi_{0j}) \cdot (\overline{\boldsymbol{w}} - \boldsymbol{w})_l + (\boldsymbol{\delta}_0)_j \quad (j = 1, \ldots, m_0), \tag{73}$$

where $\xi_{3j} = (1 - \theta_{3j})\overline{\boldsymbol{v}} - \theta_{3j}\boldsymbol{v}$, $\xi_{0j} = (1 - \theta_{0j})\overline{\boldsymbol{w}} - \theta_{0j}\boldsymbol{w}$ $(0 < \theta_{kj} < 1$, for all k and $j)$. Note again that the above equations are exact. In evaluating the values of estimates

$|\,(\bar{z}-z)_k\,|$, we may practically assume that $\theta_{lj}=0$ for all l and j, and $\bar{\delta}_i$'s are expanded according to (38), so that we can compute the estimates of the rounding errors accumulated in the kth component $\bar{z}_k$ of the final result $\bar{z}$ $(k=1,\ldots,m_3)$ [5]. In order to compute *rigorous* upper bounds of the rounding errors accumulated in $\bar{z}$ we may resort to interval arithmetics [3, 23].

7. CONCLUDING REMARKS

(i) Most of the computational efforts are devoted to the factorization of the Jacobian matrix $\Phi = LU$. It must be noted that the Jacobian matrix at the last iteration step only is required for the sensitivity analysis. With this factorization, we can construct the solution of the adjoint system so that it is not necessary to solve the adjoint system explicitly again although it is sometimes said that it is necessary to do so after solving the original system [6].

(ii) In the above formulation, we can transparently see that the sensitivity is computed with (12) and (13). Since the number of rows of $[\,I_{p'},O_{p',n},O_{p',n}\,]$ is less than that of the columns, the product of the matrices $[\,I_{p'},O_{p',n},O_{p',n}\,]U^{-1}L^{-1}$ ought to be multiplied from left to right.

(iii) The reason why the adjoint system is sometimes called the transposed system may be that the coefficient matrix of the constitutive equations of the adjoint system is equal to the transpose of that of the linearized constitutive equations of the original system. However, in this respect, there are more penetrating arguments (see, e.g., [16]).

(iv) In most cases, the matrix Ψ admits some physical interpretation in the adjoint system when (22), (23) and (24) hold in the original system, but the relation between the extensions of the adjoint system discussed in [16] and the sensitivity analysis is not clear.

(v) Our formulation can be interpreted as a kind of fast automatic differentiation of the computational process where the inner product is regarded as a basic operation during the LU-factorization [21].

(vi) A system of which the adjoint system is the same as the original is said to be *reciprocal*. Even for reciprocal systems, we should distinguish the adjoint system from the original although the numerical treatment of reciprocal systems is easier than the nonreciprocal.

(vii) Error analysis requires the same amount of computational labour as those required for sensitivity analysis with the adjoint system. We can compute

rigorous upper bounds on errors by combining interval arithmetics and fast automatic differentiation [23].

(viii) There are many systems described in terms of partial/ordinary differential equations and/or integrals on multi-dimensional regions of complicated shapes. Unless we can solve the systems analytically, we have to discretize them and solve the discretized approximate systems. In such a case, there are two possible ways for analyzing sensitivity: one is to compute the discretized system of the adjoint system of the original system, and the other is to compute the adjoint system of the discretized system of the original system. The difference is rather delicate but the authors consider that the latter is better and more consistent than the former approach. Discussions on such questions are formed in [15, 26].

REFERENCES

[1] Adelman, H. M., and Haftka, R. T., "Sensitivity analysis for discrete structural systems", *AIAA Journal*, 24 (1986) 823-832.

[2] Adelman, H. M., and Haftka, R. T., "Recent developments in structural sensitivity analysis", *Structural Optimization*, 1 (1989) 137-151.

[3] Alefeld, G., and Herzberger, J., *Introduction to Interval Computations*, Academic Press, New York, 1983.

[4] Berz, M., Bischof, C., Corliss, G., and Griewank, A. (eds.), *Computational Differentiation: Techniques, Applications, and Tools*, SIAM, Philadelphia, Pennsylvania, 1996.

[5] Christianson, B., "Reverse accumulation and implicit functions", Technical Report, Faculty of Information Sciences, University of Hertfordshire, Hatfield, England, 1997.

[6] Chua, L. O., and Lin, P.-M., *Computer-Aided Analysis of Electronic Circuits*, Prentice Hall, Inc., Englewood Cliffs, New Jersey, 1975.

[7] Griewank, A., and Corliss, G. F. (eds.), *Automatic Differentiation of Algorithms: Theory, Implementation, and Application*, SIAM, Philadelphia, Pennsylvania, 1991.

[8] Iri, M., "Simultaneous computation of functions, partial derivatives and estimates of rounding errors – Complexity and practicality", *Japan Journal of Applied Mathematics*, 1 (1984) 223-252.

[9] Iri, M., "History of automatic differentiation and rounding error estimation", in: Griewank, A., and Corliss, G. F. (eds.), *Automatic Differentiation of Algorithms: Theory, Implementation, and Application*, SIAM, Philadelphia, Pennsylvania, 1991, 3-16.

[10] Iri, M., "Computational methods for the sensitivity analysis of linear systems and the re-evaluation of the role of the 'adjoint system' from the viewpoint of fast automatic differentiation", *Proceedings of the 22nd Symposium on Numerical Analysis*, 1993, 102-106 (in Japanese).

[11] Iri, M., "A unifying approach to the sensitivity analysis of large systems containing nonlinear elements", *Proceedings of the 23rd Symposium on Numerical Analysis*, 1994, 47-49 (in Japanese).

[12] Iri, M., "A numerical-computational approach to the concept of adjoint system", *Proceedings of the 24th Symposium on Numerical Analysis*, 1995, 36-40 (in Japanese).

[13] Iri, M., "Roles of automatic differentiation in nonlinear analysis and high-quality computation", *Nonlinear Analysis*, 30 (1997) 4317-4328.

[14] Iri, M., and Kubota, K., "Remarks on matrix inversion, sensitivity analysis, etc. from the standpoint of fast automatic differentiation", *Proceedings of the 21st Symposium on Numerical Analysis*, 1992, 134-139 (in Japanese).

[15] Iri, M., Kubota, K., and Murota, K., "Geometrical/geographical optimization and fast automatic differentiation", *Yugoslav Journal of Operations Research*, 1 (1991) 121-134.

[16] Iri, M., and Recski, A., "What does duality really mean?", *Circuit Theory and Applications*, 8 (1980) 317-324.

[17] Iri, M., Tsuchiya, T., and Hoshi, M., "Automatic computation of partial derivatives and rounding error estimates with applications to large-scale systems of nonlinear equations", *Journal of Computational and Applied Mathematics*, 24 (1988) 365-392.

[18] Jerrell, M. E., "Function minimization and automatic differentiation using C++", *OOPSLA'89 Conference Proceedings* (*Special Issue of SIGPLAN Notices*), 24 (1989) 169-173.

[19] Kim, K. V., Nesterov, Yu. E., Skokov, V. A., and Cherkasskiĭ, B. V., "Éffektivnyĭ algoritm vychisleniya proizvodnykh i èkstremal'nye zadachi", *Ékonomika i Matematicheskie Metody*, 20 (1984) 309-318.

[20] Klatte, R., Kulisch, U., Wiethoff, A., Lawo, C., and Rauch, M., *C-XSC: A C++ Class Library for Extended Scientific Computing*, Springer-Verlag, Berlin, 1993.

[21] Kubota, K., "Matrix inversion algorithms by means of automatic differentiation", *Applied Mathematics Letters*, 7 (1994) 19-22.

[22] Kubota, K., "PADRE2 – FORTRAN precompiler for automatic differentiation and estimates of rounding errors", in: Berz, M., Bischof, C., Corliss, G., and Griewank, A. (eds.), *Computational Differentiation: Techniques, Applications, and Tools*, SIAM, Philadelphia, Pennsylvania, 1996, 367-374.

[23] Kubota, K., and Iri, M., "Estimates of rounding errors with fast automatic differentiation and interval analysis", *Journal of Information Processing*, 14 (1991) 508-515.

[24] Linnainmaa, S., "Taylor expansion of the accumulated rounding error", *BIT*, 16 (1976) 146-160.

[25] Miller, W., and Wrathall, C., *Software for Roundoff Analysis of Matrix Algorithms*, Academic Press, New York, 1980.

[26] Stetter, H. J., "Validated solution of initial value problems for ODE", in: C. Ullrich (ed.), *Computer Arithmetic and Self-Validating Numerical Methods*, Academic Press, Inc., 1990, 171-187.