Mathematical Programming 41 (1988) 1-27
North-Holland

SECOND-ORDER SENSITIVITY ANALYSIS IN FACTORABLE PROGRAMMING: THEORY AND APPLICATIONS*

Richard H.F. JACKSON

Center for Applied Mathematics, National Bureau of Standards, Gaithersburg, MD 20899, USA

Garth P. McCORMICK

Department of Operations Research, School of Engineering and Applied Science, George Washington University, Washington, DC 20052, USA

Received 14 January 1987
Revised manuscript received 22 June 1987

Second-order sensitivity analysis methods are developed for analyzing the behavior of a local solution to a constrained nonlinear optimization problem when the problem functions are perturbed slightly. Specifically, formulas involving third-order tensors are given to compute second derivatives of components of the local solution with respect to the problem parameters. When in addition, the problem functions are factorable, it is shown that the resulting tensors are polyadic in nature.

Key words: Second-order sensitivity analysis, high-order methods, *N*th-order derivatives, polyads, tensors, factorable functions, nonlinear optimization, Lagrangian analysis.

1. Introduction

In an earlier paper (Jackson and McCormick (1986)) the structure taken by N-dimensional arrays of Nth partial derivatives of the special class of *factorable* functions was examined. The N-dimensional arrays (or tensors as they are sometimes called) turn out to be computable naturally as the sum of generalized outer product matrices (polyads).

This natural polyadic structure has important computational implications for solving problems associated with nonlinear programming. It means for example that with some minor modification to existing software routines, high-order derivatives can be calculated efficiently, making previously intractable techniques that require them, again worthy of consideration. In the dissertation by Jackson (1983) from which most of the material in this paper is taken, these implications were pursued for second-order sensitivity analysis and high-order methods for solving the problem:

$$\begin{aligned} &\underset{x \in R^n}{\text{minimize}} && f(x), \\ &\text{subject to} && g_i(x) \geqslant 0, \end{aligned} \tag{1.1}$$

for $i = 1, \ldots, m$, when $f(x)$ and $g_i(x)$ are factorable functions.

The ability to compute third derivatives efficiently provides ready access to second-order nonlinear programming sensitivity information. In Section 3 of this

* Research sponsored by contract N00014-86-K-0052, US Office of Naval Research.

paper, second-order sensitivity analysis methods are developed for analyzing the behavior of a local solution to (1.1) when the problem functions are perturbed slightly. Section 3 begins by summarizing results from first-order sensitivity analysis which provide formulas for the first derivatives of the components of the local solution with respect to the problem parameters. Also developed are formulas, involving third-order tensors, for computing the second derivatives of the local solution with respect to the problem parameters. In addition, the polyadic structure of the tensors is investigated and displayed, and techniques for manipulating these three-dimensional arrays, capitalizing on this special structure, are developed. In general, this type of array manipulation is straightforward but time-consuming and requires significant computer storage. It is shown that these difficulties are ameliorated when the special structure of factorable functions is exploited. Examples of the use of these formulas for estimating the solution to perturbed problems using Taylor series approximations are also given.

The next section provides some definitions and mathematical background useful in understanding the results given in Section 3. Before proceeding, however, some comments on notation are required. There are unavoidable complications in the theory that follows that require subscripted subscripts. In some cases these are used. In other cases, subscripted subscripts are replaced with subscript functions. For example, $i_j \to i(j)$. The choice in each case was made on the basis of clarity of resulting formulae. Also in what follows, all vectors are assumed to be column vectors, and, where not otherwise stated, differentiation is with respect to the vector $x = (x_1, x_2, \ldots, x_n)^T$. Lastly, we use ∂ and ∇ to indicate partial differentiation, and d and D indicate total differentiation.

2. The polyadic structure of tensors of factorable functions

Loosely, a factorable function is a multivariable function that can be written as the last of a finite sequence of functions, in which the first n functions in the sequence are just the coordinate variables, and each function beyond the nth is a sum, a product, or a single-variable transformation of previous functions in the sequence. More rigorously, let $[f_1(x), f_2(x), \ldots, f_L(x)]$ be a finite sequence of functions such that $f_i: R^n \to R$, where each $f_i(x)$ is defined according to one of the following rules.

Rule 1. For $i = 1, \ldots, n$, $f_i(x)$ is the value of the ith Euclidean coordinate:

$$f_i(x) = x_i.$$

Rule 2. For $i = n+1, \ldots, L$, $f_i(x)$ is formed using *one* of the following compositions:

$$\begin{aligned} &\text{(a)} \quad f_i(x) = f_{j(i)}(x) + f_{k(i)}(x), \text{ or} \\ &\text{(b)} \quad f_i(x) = f_{j(i)}(x) \cdot f_{k(i)}(x), \text{ or} \\ &\text{(c)} \quad f_i(x) = T_i[f_{j(i)}(x)], \end{aligned} \tag{1.2}$$

where $j(i) < i$, $k(i) < i$, and T_i is a function of a single variable.

Then $f(x) = f_L(x)$ is a *factorable function* and $[f_1(x), f_2(x), \ldots, f_L(x)]$ is a *factored sequence.* Thus a function, $f(x)$, will be called factorable if it can be formed according to Rules 1 and 2, and the resulting sequence of functions will be called a factored sequence or, at times, the function written in factored form.

Although it is not always immediately grasped, the concept of a factorable function is actually a very natural one. In fact, it is just a formalization of the natural procedure one follows in evaluating a complicated function. See Jackson and McCormick (1986) for examples that illustrate this.

In order to understand what follows, the concept of an outer product matrix must be introduced. An $(m \times n)$ matrix A is called an *outer product matrix* if there exists a scalar α, and $(m \times 1)$ vector a, and an $(n \times 1)$ vector b such that

$$A = a\alpha b^{\mathrm{T}}.$$

The expression $a\alpha b^{\mathrm{T}}$ is called an *outer product* or a *dyad.* Note that a dyad is conformable since the dimensions of the product are $(m \times 1)(1 \times 1)(1 \times n)$, which yields the $(m \times n)$ outer product matrix A as desired. A useful property of outer product matrices is that, if they are kept as dyads, matrix multiplication with them is simplified to inner products alone, saving the computations required to form the matrices involved. For example,

$$Ac = a\alpha[b^{\mathrm{T}}c],$$

$$d^{\mathrm{T}}A = [d^{\mathrm{T}}a]\alpha b^{\mathrm{T}}, \text{ and}$$

$$AF = a\alpha[b^{\mathrm{T}}F],$$

where c is $(n \times 1)$, d is $(m \times 1)$ and F is $(n \times p)$.

It is well-known (see McCormick (1983)) that factorable functions possess two very special properties that can be exploited to produce efficient (fast and accurate) algorithms: (i) once written in factorable form, their gradients and Hessians may be computed exactly, automatically, and efficiently; and (ii) their Hessians occur naturally as sums of dyads whose vector factors are gradients of terms in the factored sequence. The first of these properties has eased the task of providing the derivatives of a nonlinear programming problem to a computer code that solves it, and has the potential eventually to trivialize it. The second, as noted above, changes the way we look at matrix multiplication, which in many cases results in less computational effort.

There are factorable problems whose structure is such that the factorable approach results in more work: small, dense problems, for example. For these problems, the factorable approach can still be used for easy input, but some of the matrix techniques would be replaced by classical approaches.

Software packages have been written that perform the factoring automatically from natural language input. See Jackson and McCormick (1987) for a history of such efforts and more recently, Jackson, McCormick, and Sofer (1988). The latter describes a system that allows user input for nonlinear functions in a format similar

to FORTRAN, without any requirement on the user to understand the details of factorable functions.

As mentioned above, one fundamental value of factorable functions lies in the simple and computationally efficient forms that result for their Hessians. In fact factorable programming is based on the existence of, and the simplified operations that result from, these simple forms. The seminal result, (Fiacco and McCormick, 1968, pp. 184–188), is that the Hessian of a factorable function can be written as the sum of dyads, or outer products, of gradients of functions in the factored sequence. This basic result was generalized in Jackson and McCormick (1986). Before explaining the generalization, it is necessary to generalize the concepts of Hessian and dyad.

Let $A \in R^{(n_1 \times \cdots \times n_N)}$, and let $A_{i_1,\ldots,i_N}$ denote the $(i_1, \ldots, i_N)$th element of this array. For the purposes of this paper, A is called the *Nth-order tensor* of a multivariable function $f(x)$ if

$$A_{i_1,\ldots,i_N} = \partial^N f(x)/\partial x_{i_N} \cdots \partial x_{i_1}.$$

Note that gradients and Hessians are tensors of order 1 and 2 respectively.

An N-dimensional array A is called *generalized outer product matrix* if there exists a scalar α, and an ordered set of vectors $a_1, \ldots, a_N$ (where each a_k is $(n_k \times 1)$) such that each element of A is generated by the product of the scalar α and certain specific elements of the vectors $a_1, \ldots, a_N$ as follows

$$A_{i_1,\ldots,i_N} = \alpha * a_{1,i_1} * \cdots * a_{N,i_N},$$

for $i_1 = 1, \ldots, n_1; \ldots; i_N = 1, \ldots, n_N$, where a_{k,i_k} represents the (i_k)th element of the $(n_k \times 1)$ vector a_k.

The scalar and set of vectors which generate a generalized outer product matrix taken together are called a *polyad* and are written

$$(\alpha : a_1 \cdots a_N), \tag{2.1}$$

where order is important, i.e. the vector in position j is associated with the jth dimension. A polyad containing N vector factors is an *N-ad.* Also, an expression containing a sum of polyads is a *polyadic*, and an expression containing a sum of N-ads is an *N-adic.* (The actual addition here is performed as a sum of the associated generalized outer product matrices.) When vector factors in a polyad are repeated, exponential notation is used, as in the case of the symmetric N-ad, $(\alpha : [a]^N)$. Note that the representation of a generalized outer product matrix by a polyad is not unique. For example, $(\alpha/\gamma : [a_1\gamma] \cdots a_N)$ generates the same N-dimensional array of numbers as does (2.1) for any nonzero scalar γ. Finally, a *2-ad* of the form $(\alpha : ab)$ is equivalent to the more familiar dyad of the form $a\alpha b^T$, and the two will be used interchangeably.

The generalization mentioned above is given in the following theorem. It states that *all* tensors (that exist) of factorable functions possess a natural polyadic structure.

Theorem 1 (Polyadic tensors). *Let $f(x)$ be a factorable function in R^n, let $[f_1(x), f_2(x), \ldots, f_L(x)]$ be a factored sequence for $f(x)$, and assume that $f(x) \in C^N$ and $f_i(x) \in C^N$, for $i = 1, \ldots, L$. Then the Nth-order tensor of $f(x)$ can be written as the sum of generalized outer product matrices whose associated polyads have the form $(\alpha : a_1 \cdots a_N)$ where each a_k is the gradient of some function in the factored sequence, and the scalar α is a product of functions in the factored sequence and derivatives of the single-variable transformations used in defining the functions in the sequence. Only derivatives $\partial^k T[f]/\partial f^k$, for $1 \leq k \leq N$, are used.*

Proof. See Jackson and McCormick (1986).

It was also shown in Jackson and McCormick (1986) that the vector factors that comprisse the monads of the gradient are the same vector factors which comprise the dyads of the Hessian, the triads of the third order tensor, and so on. This has important computational implications in mathematical programming. It means that once the gradient of a factorable function is computed, a major portion of the work involved in computing higher-order derivatives is already calculated. Consequently, high-order minimization techniques, previously considered computationally intractable, are once again worthy of consideration. See Jackson and McCormick (1986) for more.

It should be noted that, by their very nature, the tensors of factorable functions are ideally suited for computation on parallel processing and array processing computers. We know of few other such ideal applications in numerical optimization. Also, it has been shown (McCormick (1985)) that all factorable programming problems have an equivalent separable programming representation, and that efficient algorithms (Falk and Soland (1969), Falk (1973), Hoffman (1975), McCormick (1976), Leaver (1984)) exist for finding global solutions to these problems. Thus there exists the potential of finding global solutions to factorable programming problems fast and accurately.

3. Second-order sensitivity analysis in nonlinear programming

3.1. Basic first-order sensitivity results

One application of the results in Section 2 is in obtaining high-order sensitivity information for nonlinear programming problems, although only the second-order case is considered in this paper. Sensitivity analysis in nonlinear programming is concerned with analyzing the behavior of a local solution when the problem functions are perturbed slightly. This perturbation might be due to an inexactness with which certain parameter values in the problem are calculated or because the optimization model was parameterized and one is interested in the solution for a variety of values of the parameters. For additional information on this topic, see Armacost and Fiacco (1974), Armacost and Fiacco (1978), Fiacco (1980) and especially Fiacco (1983)

and its excellent bibliography. The parametric problem is written

$$\begin{aligned} P(\varepsilon): \quad & \underset{x \in R^n}{\text{minimize}} && f(x, \varepsilon), \\ & \text{subject to} && g_i(x, \varepsilon) \geq 0, \end{aligned} \tag{3.1}$$

for $i = 1, \ldots, m$, where ε is an $(r \times 1)$ vector of parameters. The more general version of the sensitivity problem includes equality constraints, but these are not included here for simplicity. The ideas and results presented in this section generalize readily to the equality-constrained problem.

The essence of sensitivity analysis in nonlinear programming is the application of the Implicit Function Theorem (see, e.g. Bliss (1946) to the Karush-Kuhn-Tucker (KKT) necessary conditions for the problem, $P(\varepsilon)$, in (3.1). First, define the Lagrangian for $P(\varepsilon)$ as

$$L(x, u, \varepsilon) = f(x, \varepsilon) - \sum_{i=1}^{m} u_i g_i(x, \varepsilon).$$

Then, assuming continuous differentiability in x of the problem functions, the KKT conditions for $P(\varepsilon)$ are that there exists a feasible point, x, for (3.1) and associated vector of Lagrange multipliers, u, such that

$$\begin{aligned} & \nabla L(x, u, \varepsilon) = 0, \\ & u_i g_i(x, \varepsilon) = 0, \\ & u_i \geq 0, \end{aligned} \tag{3.2}$$

for $i = 1, \ldots, m$.

The statement of the first-order sensitivity results given in Theorem 3 below also requires that the second-order sufficient conditions (SOSC) hold at a particular solution $\hat{x}$, of $P(\hat{\varepsilon})$ (which is just (3.1) for a specified vector of parameter values, $\hat{\varepsilon}$). These conditions may be written as follows.

Theorem 2 (SOSC). *Let $\hat{x}$ be a feasible point for $P(\hat{\varepsilon})$ and assume that the functions of $P(\hat{\varepsilon})$ are twice-continuously differentiable in x in a neighborhood of $\hat{x}$. Let $(\hat{x}, \hat{u}, \hat{\varepsilon})$ be a triple that satisfies the KKT conditions in* (3.2) *and define*

$$B = \{i | g_i(\hat{x}, \hat{\varepsilon}) = 0\} \quad \text{and} \quad D = \{i | \hat{u}_i > 0\}.$$

Further, suppose that

$$d^{\mathrm{T}} \nabla^2 L(\hat{x}, u, \hat{\varepsilon}) d > 0,$$

for all $d \neq 0$, such that

$$d^{\mathrm{T}} \nabla g_i(\hat{x}, \hat{\varepsilon}) \geq 0 \quad \textit{for all } i \in B,$$

and

$$d^{\mathrm{T}}\nabla g_i(\hat{x}, \hat{\varepsilon}) = 0 \quad \textit{for all } i \in D.$$

Then $\hat{x}$ is a strict local minimizer for $P(\hat{\varepsilon})$.

The earliest known reference to these conditions is Pennisi (1953), although it was almost 15 years (see McCormick (1967), and Fiacco and McCormick (1968)) before they were more fully developed and exploited.

The following theorem can be viewed as the basic result in nonlinear programming sensitivity analysis.

Theorem 3 (First-order sensitivity). *Let x be a feasible point for $P(\hat{\varepsilon})$ and assume that*

(i) *the functions in* (3.1) *are twice-continuously differentiable in x and the cross-partial derivatives exist and are jointly continuous in x and ε in a neighborhood of $(\hat{x}, \hat{\varepsilon})$;*

(ii) *the second-order sufficient conditions* (*Theorem* 2) *for a local minimum of $P(\hat{\varepsilon})$ hold at $\hat{x}$, with associated Lagrange multipliers $\hat{u}$;*

(iii) *the gradients of the binding constraints, i.e., $\nabla g_i(\hat{x}, \hat{\varepsilon})$, $i \in B$, are linearly independent; and*

(iv) *$\hat{u}_i > 0$ for all $i \in B$.*

Then, for ε in a sufficiently small neighborhood of $\hat{\varepsilon}$, there exists a unique, once-continuously differentiable vector function

$$y(\varepsilon) = [x(\varepsilon)^{\mathrm{T}}, u(\varepsilon)^{\mathrm{T}}]^{\mathrm{T}},$$

satisfying the KKT conditions for $P(\varepsilon)$, with

$$y(\hat{\varepsilon}) = [x(\hat{\varepsilon})^{\mathrm{T}}, u(\hat{\varepsilon})^{\mathrm{T}}]^{\mathrm{T}},$$

such that $x(\varepsilon)$ is a locally unique isolated minimizer for $P(\varepsilon)$. Furthermore, the first partial derivatives of $x(\varepsilon)$ and $u(\varepsilon)$ with respect to ε can be obtained from the equation

$$Y = -M^{-1}N, \tag{3.3}$$

where

$$Y = \nabla_{\varepsilon} y = \begin{bmatrix} \dfrac{\partial x_1}{\partial \varepsilon_1} & \cdots & \dfrac{\partial x_1}{\partial \varepsilon_r} \\ \vdots & \ddots & \vdots \\ \dfrac{\partial x_n}{\partial \varepsilon_1} & \cdots & \dfrac{\partial x_n}{\partial \varepsilon_r} \\ \dfrac{\partial u_1}{\partial \varepsilon_1} & \cdots & \dfrac{\partial u_1}{\partial \varepsilon_r} \\ \vdots & \ddots & \vdots \\ \dfrac{\partial u_m}{\partial \varepsilon_1} & \cdots & \dfrac{\partial u_m}{\partial \varepsilon_r} \end{bmatrix},$$

$$M = \begin{bmatrix} \nabla^2 L & -\nabla g_1 & -\nabla g_2 & \cdots & -\nabla g_m \\ u_1 \nabla g_1^{\mathrm{T}} & g_1 & 0 & \cdots & 0 \\ u_2 \nabla g_2^{\mathrm{T}} & 0 & g_2 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ u_m \nabla g_m^{\mathrm{T}} & 0 & 0 & \cdots & g_m \end{bmatrix}$$

and

$$N = \begin{bmatrix} \nabla^2_{\varepsilon x} L \\ u_1 \nabla_\varepsilon g_1^{\mathrm{T}} \\ u_2 \nabla_\varepsilon g_2^{\mathrm{T}} \\ \vdots \\ u_m \nabla_\varepsilon g_m^{\mathrm{T}} \end{bmatrix}$$

with all quantities in M and N evaluated at $(\hat{x}^{\mathrm{T}}, \hat{u}^{\mathrm{T}})^{\mathrm{T}}$.

Proof. See Fiacco (1983).

3.2. Development of the second-order equation

The result in (3.3) provides a direct way of calculating first-order sensitivity information and is the point of departure for the work given here, which begins with the following theorem due to Fiacco (1983).

Theorem 4 (Higher-order sensitivity). *Let* $\hat{x}$ *be a feasible point for* $P(\hat{\varepsilon})$ *and assume conditions* (i) *through* (iv) *in Theorem* 3. *Assume also that all* $(k+1)$*st-order partial derivatives in x and all* $(k+1)$*st-order cross partial derivatives in x and* ε *exist and are jointly continuous in x and* ε *in a neighborhood of* $(\hat{x}, \hat{\varepsilon})$. *Then in a sufficiently small neighborhood of* $\hat{\varepsilon}$, *the vector function* $y(\varepsilon)$ *is k times continuously differentiable.*

Proof. The proof follows directly from the fact that if the Jacobian, M, of the KKT conditions in (3.2) is nonsingular, and if the functions involved posses the appropriate degree of differentiability, the Implicit Function Theorem (see Bliss, 1946, p. 270) guarantees the existence of the higher-order partial derivatives. Nonsingularity of M for (3.2) was shown in Fiacco and McCormick (1968). Thus the theorem is proved by direct application of the Implicit Function Theorem.

If this high-order sensitivity information is to be used, it is necessary also to develop a convenient mechanism for calculating it. This is done next for the case when $k=2$. Consider the matrix equation in (3.3) and rewrite it as

$$MY = -N.$$

The (i, j)th element of this matricial equation is

$$\sum_s M_{is} Y_{sj} = -N_{ij},$$

and, by straightforward applications of the product-rule and the chain rule, the total derivative of this equation is written

$$-\sum_s M_{is} \frac{\partial Y_{sj}}{\partial \varepsilon_k} = \sum_s \left[\sum_t \frac{\partial M_{is}}{\partial y_t} Y_{tk} + \frac{\partial M_{is}}{\partial \varepsilon_k} \right] Y_{sj} + \sum_t \frac{\partial N_{ij}}{\partial y_t} Y_{tk} + \frac{\partial N_{ij}}{\partial \varepsilon_k}. \tag{3.4}$$

Now consider the left-hand-side (lhs) in (3.4) more closely. It can be written

$$-\sum_s M_{is} \frac{\partial^2 y_s}{\partial \varepsilon_k \, \partial \varepsilon_j},$$

which is of the form

$$-\sum_s M_{is} A_{sjk},$$

which in turn gives the (i, j, k)th element of the three-dimensional array that results from the matrix multiplication of M and the kth (counting from front to rear) two-dimensional matrix of A that is parallel to the xz-plane in R^3.

Observe that $\partial^2 y_i / \partial \varepsilon_k \partial \varepsilon_j$ can be obtained by premultiplying both sides of (3.4) by $H = M^{-1}$. Then

$$\frac{\partial^2 y_i}{\partial \varepsilon_j \, \partial \varepsilon_k} = -\sum_r H_{ir} \left\{ \sum_s \left[\sum_t \frac{\partial M_{rs}}{\partial y_t} \frac{\partial y_t}{\partial \varepsilon_k} + \frac{\partial M_{rs}}{\partial \varepsilon_k} \right] \frac{\partial y_s}{\partial \varepsilon_j} + \sum_t \frac{\partial N_{rj}}{\partial y_t} \frac{\partial y_t}{\partial \varepsilon_k} + \frac{\partial N_{rj}}{\partial \varepsilon_k} \right\}, \tag{3.5}$$

which is the (i, j, k)th element of the three-dimensional array of second partial derivatives of $y = [x(\varepsilon)^{\mathrm{T}}, u(\varepsilon)^{\mathrm{T}}]^{\mathrm{T}}$ with respect to ε.

It is desired next to write (3.5) using array notation. Because these operations are performed in three-space, however, some new notation must be developed before this is rewritten. In order to motivate the new notation, consider a multivariable function $f(x)$. It is perhaps clear in this case what is meant by ∇f, $\nabla^2 f$ and $\nabla^3 f$; i.e., ∇f is a vector that is written down the page, $\nabla^2 f$ requires taking the gradient of each element of ∇f and writing that result across the page to form a two-dimensional matrix, and thus to form $\nabla^3 f$ one would take the gradient of each element of $\nabla^2 f$ and write that result into the third dimension (into the page, say). Hence if M is a matrix, it should be clear what is meant by ∇M; i.e., take the gradient of each element of M and write the result into the third dimension.

But, if y is a vector in R^2, there are three possible orientations parallel to the coordinate planes in R^3 for the matrix usually notated as ∇y. These are shown[1] in Figure 1. In order clearly to differentiate among these, the notation "$\nabla\{\cdot\}$" will be

[1] All graphs were produced using DATAPLOT as described in Filliben (1981).

used. Thus $\nabla\{\cdot\}$ operates on the argument by taking its gradient into the third dimension. Note that

$$\nabla y \neq \nabla\{y\} \neq \nabla\{y^{\mathrm{T}}\}.$$

Whereas the elements of these vectors are the same, their orientations in three-space are not. These, too, are shown in Fig. 1. The same comments apply with regard to total differentiation, for which the notation $D\{\cdot\}$ will be used. This notation, of course, is only used for an operation from R^2 to R^3; analogs exist for higher dimensions.

Now with this notation, the matrix form of (3.5) is written:

$$\nabla^2_{\varepsilon\varepsilon} y = -M^{-1}[\nabla_y M \nabla_\varepsilon\{y\}\nabla_\varepsilon y + \nabla_\varepsilon M \nabla_\varepsilon y + \nabla_y N \nabla_\varepsilon\{y\} + \nabla_\varepsilon N], \tag{3.6}$$

where, letting $p = (n+m)$,

$\nabla^2_{\varepsilon\varepsilon} y$ is $(p \times r \times r)$, $\nabla_\varepsilon y$ is $(p \times r \times 1)$,

M^{-1} is $(p \times p \times 1)$, $\nabla_\varepsilon M$ is $(p \times p \times r)$,

$\nabla_y M$ is $(p \times p \times p)$, $\nabla_y N$ is $(p \times r \times p)$,

$\nabla_\varepsilon\{y\}$ is $(p \times 1 \times r)$, $\nabla_\varepsilon N$ is $(p \times r \times r)$,

and the multiplication in three-space is carried out so that each array within the brackets on the rhs of (3.6) is $(p \times r \times r)$. This $(p \times r \times r)$ array must then be premultiplied by M^{-1}, a $(p \times p \times 1)$ array. This multiplication is effected by premultiplying each of the r matrices of dimension $(p \times r)$ by M^{-1}, resulting in r matrices of dimension $(p \times r)$, or a $(p \times r \times r)$ array again, as required by $\nabla^2_{\varepsilon\varepsilon} y$. More on three-dimensional array multiplication (including graphical depictions) is given in Section 3.5.

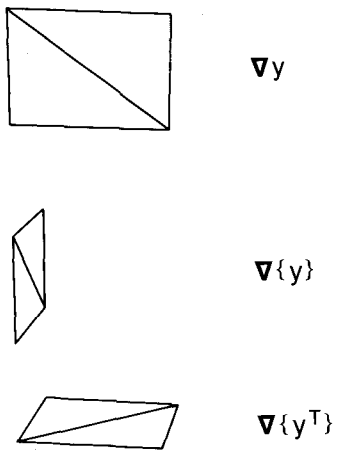

Fig. 1. Possible orientations of a matrix in three-space.

Observe that (3.3) could have been differentiated directly using the techniques for differentiation of matrices as given in Marlow (1978). He uses Kronecker products and a special two-dimensional representation of three-dimensional arrays for performing this kind of differentiation. A disadvantage of this approach is that whatever special structure that exists in the three-dimensional arrays is lost in the process. This "structural integrity" is maintained in the approach given here, allowing the more detailed analysis given in the next section.

There has been some previous work in second-order sensitivity analysis. Dembo (1982) derived a computationally efficient technique for getting an approximation to the second derivative with respect to ε of the vector function y, correct to terms of order two. By contrast, (3.6) is exact. Also, a result for the geometric programming problem for the case where ε is a scalar was provided by Kyparisis (1983).

The material in the remainder of this section addresses the issue of computational efficiency when the problem functions are factorable. Sections 3.3 and 3.4 are admittedly rather detailed. The reason for including them is twofold: to illustrate the special polyadic structure of the tensors in nonlinear programming sensitivity analysis, and to stimulate more research in these areas. It is not necessary to wade through this material for each second-order sensitivity calculation. Ultimately this will be performed automatically by the Factorable Programming system of programs being developed at the National Bureau of Standards.

3.3. *Structure of the three-dimensional arrays*

While the formula in (3.6) may be mathematically succinct, it may not be obvious how the polyadic structure of derivatives of factorable functions can assist in its calculation. To understand how these calculations are performed, it is necessary to investigate further the structure of the three-dimensional arrays involved. These are: $\nabla_y M$, $\nabla_\varepsilon M$, $\nabla_y N$, and $\nabla_\varepsilon N$. Since $\nabla_\varepsilon N$ is the simplest of these, it is considered first. Strictly speaking, the (i, j, k)th element of $\nabla_\varepsilon N$ is

$$(\nabla_\varepsilon N)_{ijk} = \begin{cases} \dfrac{\partial^3 L}{\partial \varepsilon_k \, \partial \varepsilon_j \, \partial x_i}, & i \leqslant n, \\ u_{i-n} \dfrac{\partial^2 g_{i-n}}{\partial \varepsilon_k \, \partial \varepsilon_j}, & i > n. \end{cases}$$

This can be thought of as taking the gradient with respect to ε of each element of N into the third dimension, and thus can be pictured as a partitioned rectangular parallelepiped as shown in Fig. 2. The partitioning, which is due to the parts in y, separates $\nabla_\varepsilon N$ into an "upper" part which is $\nabla^3_{\varepsilon\varepsilon x} L$ and a "lower" part which is just a stack of constraint Hessians with respect to ε. This more detailed structure is shown in the exploded view of $\nabla \varepsilon N$ given in Fig. 3.

The next simplest array to portray is $\nabla_y N$. This is a three-dimensional array that has four parts, again a result of the two parts in y, arranged as shown in Fig. 4.

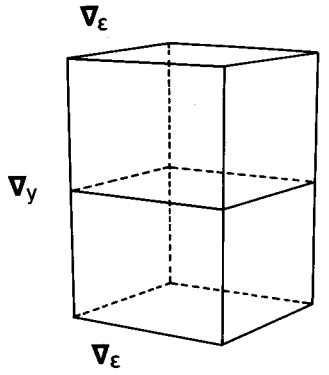

Fig. 2. Basic partitioned structure of $\nabla_\varepsilon N$.

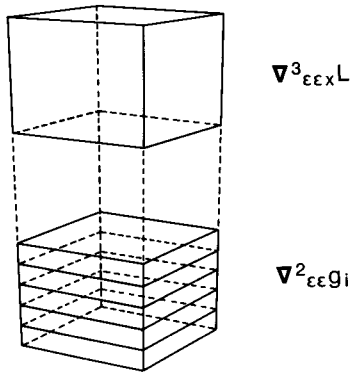

Fig. 3. Exploded view of structural details of $\nabla_\varepsilon N$.

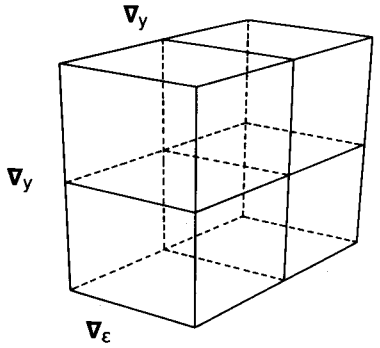

Fig. 4. Basic partitioned structure of $\nabla_y N$.

These partitions are described mathematically as:

$$(\nabla_y N)_{ijk} = \begin{cases} \dfrac{\partial^3 L}{\partial x_k\, \partial \varepsilon_j\, \partial x_i}, & i \leq n,\ k \leq n, \\ u_{i-n} \dfrac{\partial^2 g_{i-n}}{\partial x_k\, \partial \varepsilon_j}, & i > n,\ k \leq n, \\ -\dfrac{\partial^2 g_{k-n}}{\partial \varepsilon_j\, \partial x_i}, & i \leq n,\ k > n, \\ \dfrac{\partial g_{i-n}}{\partial x_j}, & i > n,\ k > n,\ i = k, \\ 0, & \text{otherwise.} \end{cases}$$

These are depicted graphically in the exploded view in Fig. 5, where the different orientations of the various matrices and vectors in three-space are more easily grasped.

The next three-dimensional array to consider is $\nabla_\varepsilon M$. This too is partitioned into four smaller three-dimensional arrays, but with an orientation that differs from $\nabla_y N$. The basic structure of the parts is shown in Fig. 6, and each element of the array

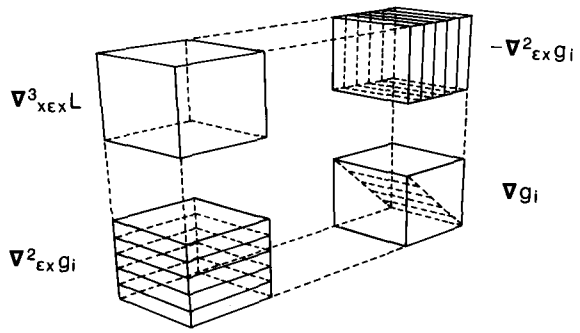

Fig. 5. Exploded view of structural details of $\nabla_y N$.

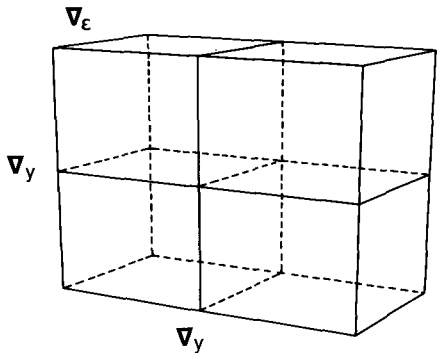

Fig. 6. Basic partitioned structure of $\nabla_\varepsilon M$.

is defined in the following equations:

$$(\nabla_\varepsilon M)_{ijk} = \begin{cases} \dfrac{\partial^3 L}{\partial \varepsilon_k \, \partial x_j \, \partial x_i}, & i \leq n, \ j \leq n, \\ -\dfrac{\partial^2 g_{j-n}}{\partial \varepsilon_k \partial x_i}, & i \leq n, j > n, \\ u_{i-n} \dfrac{\partial^2 g_{i-n}}{\partial \varepsilon_k \, \partial x_j}, & i > n, \ j \leq n, \\ \dfrac{\partial g_{i-n}}{\partial \varepsilon_k}, & i > n, \ j > n, \ i = j, \\ 0, & \text{otherwise.} \end{cases}$$

The three-dimensional orientation of these matrices and vectors is shown in more detail in Fig. 7.

The last three-dimensional array to be depicted is $\nabla_y M$, the most complicated, with eight parts that result from differentiating three times with respect to the partitioned vector y. The eight parts are shown in Fig. 8, and the mathematical

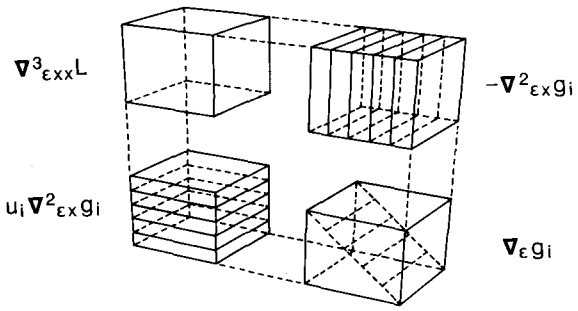

Fig. 7. Exploded view of structural details of $\nabla_\varepsilon M$.

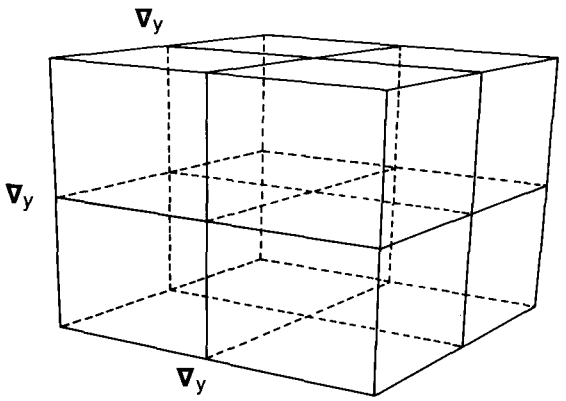

Fig. 8. Basic partitioned structure of $\nabla_y M$.

statement of the (i, j, k)th element for each case is given below.

$$(\nabla_y M)_{ijk} = \begin{cases} \dfrac{\partial^3 L}{\partial x_k\, \partial x_j\, \partial x_i}, & i \leq n,\ j \leq n,\ k \leq n, \\ -\dfrac{\partial^2 g_{j-n}}{\partial x_k\, \partial x_i}, & i \leq n,\ j > n,\ k \leq n, \\ u_{i-n} \dfrac{\partial^2 g_{i-n}}{\partial x_k\, \partial x_j}, & i > n,\ j \leq n,\ k \leq n, \\ \dfrac{\partial_{i-n}}{\partial x_k}, & i > n,\ j > n,\ k \leq n,\ i = j, \\ -\dfrac{\partial^2 g_{k-n}}{\partial x_j\, \partial x_i}, & i \leq n,\ j \leq n,\ k > n, \\ 0, & i \leq n,\ j > n,\ k > n, \\ \dfrac{\partial g_{i-n}}{\partial x_j}, & i > n,\ j \leq n,\ k > n,\ i = k, \\ 0, & i > n,\ j > n,\ k > n, \\ 0, & \text{otherwise.} \end{cases}$$

And finally, these structures are shown in detail in Fig. 9.

3.4. *Polyadics in second-order sensitivity analysis*

3.4.1. *The dyadics in the second-order terms*

Although the material in the previous section provided insight into the three-dimensional structure of the second-order sensitivity analysis formula in (3.6), it still may not be clear how the natural polyadic structure of tensors of factorable functions can assist in the evaluation of the formula. To see this, it is first necessary to show that each substructure exhibited in Figs 3, 5, 7, and 9, is a polyadic, and

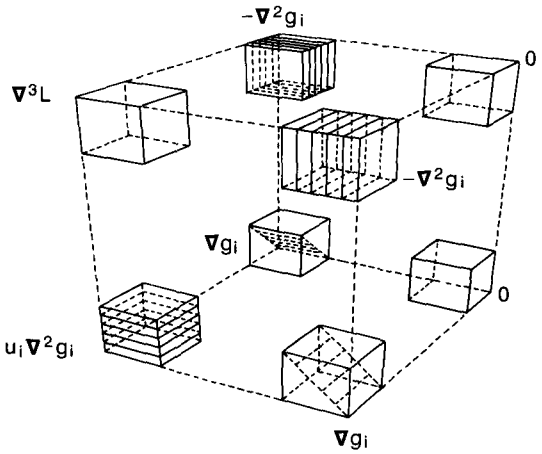

Fig. 9. Exploded view of structural details of $\nabla_y M$.

then to demonstrate how the multiplication with $\nabla_\varepsilon y$ (or $\nabla_\varepsilon\{y\}$) is to be carried out. This section addresses the former of these activities by providing proofs that factorable functions of (x, ε) have polyadic derivatives. Section 3.5 addresses the latter.

First notice that if the functions of the problem given in (3.1) are factorable in x and ε, so too is the Lagrangian of that problem factorable in x and ε. Then the task of this section reduces to considering some function $f(x, \varepsilon) = f_L(x, \varepsilon)$, with factored sequence $[f_1(x, \varepsilon), f_2(x, \varepsilon), \ldots, f_L(x, \varepsilon)]$ formed using the following modification to the rules given in (1.2).

Rule 1. For $i \leq n$,

$$f_i(x, \varepsilon) = x_i.$$

Rule 2. For $i > n$, either

$$\begin{aligned} &\text{(a)} \quad f_i(x, \varepsilon) = f_{j(i)}(x, \varepsilon) + f_{k(i)}(x, \varepsilon), \text{ or} \\ &\text{(b)} \quad f_i(x, \varepsilon) = f_{j(i)}(x, \varepsilon) \cdot f_{k(i)}(x, \varepsilon), \text{ or} \\ &\text{(c)} \quad f_i(x, \varepsilon) = T_i[f_{j(i)}(x, \varepsilon), \varepsilon]; \end{aligned} \tag{3.7}$$

where $j(i) < i$, and $k(i) < i$, and the derivatives with respect to ε of the T_i are themselves factorable functions in ε.

For convenience, in the rest of this section, let $j(i) = j$, $k(i) = k$, and drop the arguments (x, ε) and $[f_{j(i)}(x, \varepsilon)]$. Thus, e.g., $T_i[f_{j(i)}(x, \varepsilon), \varepsilon] \to T_i$.

Calculation of the Hessians with respect to x of the forms in (3.7) is straightforward and shown in Table 1. Notice that these forms are uncomplicated by derivatives with respect to ε. Therefore Theorem 1 can be invoked here to show that it is possible to write $\nabla^2 f(x, \varepsilon)$ as a sum of dyads of the appropriate form.

Table 1
Hessians of factorable function forms in sensitivity analysis

Rule	f_i	$\nabla^2 f_i$
1	x_i	$0_{n\times n}$
2a	$f_j + f_k$	$\nabla^2 f_j + \nabla^2 f_k$
2b	$f_j \cdot f_k$	$f_j \nabla^2 f_k + \nabla f_k \nabla f_j^{\mathrm{T}} + f_k \nabla^2 f_j + \nabla f_j \nabla f_k^{\mathrm{T}}$
2c	T_i	$\dot{T}_i \nabla^2 f_j + \nabla f_j \ddot{T}_i \nabla f_j^{\mathrm{T}}$

The calculation of the matrix of second partial derivatives of $f_i(x, \varepsilon)$ with respect to x and ε is slightly more complicated by the fact that T_i is a composite function of $f(x, \varepsilon)$ and ε. This requires the chain rule to obtain the second derivative matrix. Hence,

$$\mathrm{D}_{\varepsilon x} T_i = \mathrm{D}_\varepsilon[\dot{T}_i \nabla f_j] = \dot{T}_i \nabla^2_{\varepsilon x} f_j + \nabla f_j \ddot{T}_i \nabla_\varepsilon f_j^{\mathrm{T}} + \nabla f_j \nabla_\varepsilon \dot{T}_i^{\mathrm{T}},$$

where $\dot{T}_i \equiv \partial T/\partial f$ and $\ddot{T}_i = \partial^2 T/\partial f^2$. With this, the formulae for the matrix of second partials of f_i with respect to x and ε can be written as in Table 2. The first appearance of these formulae was in de Silva and McCormick (1978).

Using the results displayed in Table 4, an inductive argument paralleling that used in the proof of Theorem 2 in Jackson and McCormick (1986) will yield the fact that $\nabla^2_{\varepsilon x} f$ can also be written as the sum of dyads. The difference is that for $\nabla^2_{\varepsilon x} f$, the first vector in the dyads is a gradient with respect to x of a factored-sequence function, and the second vector is a gradient with respect to ε of a factored-sequence function *or* a derivative of a single-variable transformation.

This result implies that the submatrices of *second*-order derivatives which appear in the arrays $\nabla_y M$, $\nabla_\varepsilon M$, $\nabla_y N$, and $\nabla_\varepsilon N$ on the rhs of (3.6) and in Figs 3, 5, 7, and 9, are all dyadics. What remains is to show that the subarrays of *third*-order derivatives, in these same four arrays, are triadics. Those subarrays, arising from the block $\nabla^2 L$ in M and the block $\nabla^2_{\varepsilon x} L$ in N, are $\nabla^3 L$, $\nabla^3_{x\varepsilon x} L$, $\nabla^3_{\varepsilon x x} L$, and $\nabla^3_{\varepsilon\varepsilon x} L$. The first case, $\nabla^3 L$, again is uncomplicated by derivatives with respect to ε, and thus is a triadic by Theorem 1. The proofs for the other cases are in the same vein as the proof of Theorem 2 in Jackson and McCormick (1986) and require that the formulae for the third derivatives with respect to x and ε respectively be derived for the forms in Table 1. Consider for example $\nabla^3_{\varepsilon x x} f$. The first step in showing that $\nabla^3_{\varepsilon x x} f$ is triadic is to apply the operator $D_\varepsilon\{\cdot\}$ to each factorable function form in Table 1. This is straightforward for cases 1 and 2a. Case 2b represents the first use of the new notation, and is developed below.

$$
\begin{aligned}
&D_\varepsilon\{f_j\nabla^2 f_k + \nabla f_k \nabla f_j^{\mathrm{T}} + f_k \nabla^2 f_j + \nabla f_j \nabla f_k^{\mathrm{T}}\} \\
&\quad = f_j \nabla^3_{\varepsilon x x} f_k + \nabla^2 f_k \nabla_\varepsilon\{f_j\} + \nabla f_k \nabla_\varepsilon\{\nabla f_j^{\mathrm{T}}\} + \nabla f_j^{\mathrm{T}} \nabla_\varepsilon\{\nabla f_k\} + f_k \nabla^3_{\varepsilon x x} f_j \\
&\qquad + \nabla^2 f_j \nabla_\varepsilon\{f_k\} + \nabla f_j \nabla_\varepsilon\{\nabla f_k^{\mathrm{T}}\} + \nabla f_k^{\mathrm{T}} \nabla \varepsilon\{\nabla f_j\}.
\end{aligned}
$$

The more complicated case is 2c, which is a result of the fact that, as noted earlier, each T_i (and hence $\dot{T}_i$ and $\ddot{T}_i$) is a composite function of $f(x, \varepsilon)$ and ε, requiring the chain rule to calculate the total derivative. This is shown below.

$$
\begin{aligned}
D_\varepsilon\{\dot{T}_i \nabla^2 f_j + \nabla f_j \ddot{T}_i \nabla f_j^{\mathrm{T}}\} &= \dot{T}_i \nabla^3_{\varepsilon x x} f_j + \nabla^2 f_j D_\varepsilon\{\dot{T}_i\} + \nabla_\varepsilon\{\nabla f_j\} \ddot{T}_i \nabla f_j^{\mathrm{T}} \\
&\quad + \nabla f_j D_\varepsilon\{\ddot{T}_i\} \nabla f_j^{\mathrm{T}} + \nabla f_j \ddot{T}_i \nabla_\varepsilon\{\nabla f_j^{\mathrm{T}}\}. \qquad (3.8)
\end{aligned}
$$

Table 2

Hessians with respect to x and ε of factorable function forms in sensitivity analysis

Rule	f_i	$\nabla^2_{\varepsilon x} f_i$
1	x_i	$0_{n\times r}$
2a	$f_j + f_k$	$\nabla^2_{\varepsilon x} f_j + \nabla^2_{\varepsilon x} f_k$
2b	$f_j \cdot f_k$	$f_j \nabla^2_{\varepsilon x} f_k + \nabla f_k \nabla_\varepsilon f_j^{\mathrm{T}} + f_k \nabla^2_{\varepsilon x} f_j + \nabla f_j \nabla_\varepsilon f_k^{\mathrm{T}}$
2c	$T_i[f_j, \varepsilon]$	$\dot{T}_i \nabla^2_{\varepsilon x} f_j + \nabla f_j \ddot{T}_i \nabla_\varepsilon f_j^{\mathrm{T}} + \nabla f_j \nabla \varepsilon \dot{T}_i^{\mathrm{T}}$

However, using the chain rule,

$$D_\varepsilon\{\dot{T}_i\} = \ddot{T}_i \nabla_\varepsilon\{f_j\} + \nabla_\varepsilon\{\dot{T}_i\},$$

and (3.9)

$$D_\varepsilon\{\ddot{T}_i\} = \dddot{T}_i \nabla_\varepsilon\{f_j\} + \nabla_\varepsilon\{\ddot{T}_i\}.$$

After using (3.9), the rhs of (3.8) becomes:

$$\dot{T}_i \nabla^3_{\varepsilon xx} f_j + \ddot{T}_i \nabla^2 f_j \nabla_\varepsilon\{f_j\} + \nabla^2 f_j \nabla_\varepsilon\{\dot{T}_i\} + \nabla_\varepsilon\{\nabla f_j\} \ddot{T}_i \nabla f_j^{\mathrm{T}} + \nabla f_j \ddot{T}_i \nabla_\varepsilon\{f_j\} \nabla f_j^{\mathrm{T}}$$
$$+ \nabla f_j \nabla_\varepsilon\{\ddot{T}_i\} \nabla f_j^{\mathrm{T}} + \nabla f_j \ddot{T}_i \nabla_\varepsilon\{\nabla f_j^{\mathrm{T}}\}.$$

These third-derivative formulae for $\nabla^3_{\varepsilon xx} f$ are collected in Table 3. The proof that $\nabla^3_{\varepsilon xx} f$ is triadic is a straightforward application of induction to the factored sequence and mimics the argument used in the proof of Theorem 2 of Jackson and McCormick (1986). The result is that $\nabla^3_{\varepsilon xx} f$ can be written as the sum of triads of the form

$$(\alpha: a_1 a_2 a_3),$$

where

a_1 is $(n \times 1)$,

a_2 is $(n \times 1)$,

a_3 is $(r \times 1)$,

and α is a product of factored sequence functions and first, second, and third derivatives of the single-variable transformations used in forming the factored sequence. Also, the vector factors a_1 and a_2 are gradients with respect to x of a factored sequence function, and a_3 is a gradient with respect to ε of a factored-sequence function or of a first or second derivative with respect to f of one of the single-variable transformations in the factored sequence.

Similar arguments, tables, and results are obtained for the remaining cases: $\nabla^3_{x\varepsilon x} f$ and $\nabla^3_{xx\varepsilon} f$. See Jackson (1983) for the details.

Table 3

Third-order tensors with respect to x, x, and ε of factorable function forms in sensitivity analysis

Rule	f_i	$\nabla^3_{\varepsilon xx} f_i$
1	x_i	$0_{n \times n \times r}$
2a	$f_j + f_k$	$\nabla^3_{\varepsilon xx} f_j + \nabla^3_{\varepsilon xx} f_k$
2b	$f_i \cdot f_j$	$f_j \nabla^3_{\varepsilon xx} f_k + \nabla^2 f_k \nabla_\varepsilon\{f_j\} + \nabla f_k \nabla_\varepsilon\{\nabla f_j^{\mathrm{T}}\} + \nabla f_j^{\mathrm{T}} \nabla_\varepsilon\{\nabla f_k\} + f_k \nabla^3_{\varepsilon xx} f_j$ $+ \nabla^2 f_j \nabla_\varepsilon\{f_k\} + \nabla f_k \nabla_\varepsilon\{\nabla f_k^{\mathrm{T}}\} + \nabla f_k^{\mathrm{T}} \nabla_\varepsilon\{\nabla f_j\}$
2c	$T_i[f_j, \varepsilon]$	$\dot{T}_i \nabla^3_{\varepsilon xx} f_j + \ddot{T}_i \nabla^2 f_j \nabla \varepsilon\{f_j\} + \nabla^2 f_j \nabla_\varepsilon\{\dot{T}_i\} + \nabla_\varepsilon\{\nabla f_j\} \ddot{T}_i \nabla f_j^{\mathrm{T}}$ $+ \nabla f_j \ddot{T}_i \nabla_\varepsilon\{f_j\} \nabla f_j^{\mathrm{T}} + \nabla f_j \nabla_\varepsilon\{\ddot{T}_i\} \nabla f_j^{\mathrm{T}} + \nabla f_j \ddot{T}_i \nabla_\varepsilon\{\nabla f_j^{\mathrm{T}}\}$

3.5. Array multiplication with generalized outer product matrices

In the previous section, the polyadic nature of the arrays in (3.6) was exhibited. This section takes up the notion of multiplying these generalized outer product matrices by $Y = \nabla_\varepsilon y$, which appears in (3.6) with two different three-space orientations; via., $\nabla_\varepsilon y$ which is $(p \times r \times 1)$ and $\nabla_\varepsilon \{y\}$ which is $(p \times 1 \times r)$. Multiplication is one area wherein Factoriable Programming, through the natural polyadic nature of the function derivatives involved, offers a potentially great computational saving over alternative approaches.

Consider for instance the case of multiplication of a dyad and a matrix,

$$(\alpha : ab) * F = (a\alpha b^{\mathrm{T}})F,$$

wher a is $(n \times 1)$, b is $(m \times 1)$, F is $(m \times n)$ and α is a scalar. Of course one method of computing this is to form $(\alpha a)b^{\mathrm{T}}$ which requires $n(m+1)$ multiplications, then to multiply the result by F, requiring mn^2 multiplications and additions. A far more efficient way, however, is to form $c = b^{\mathrm{T}}F$, requiring the same nm multipliccations and additions, but now store the result in the dyadic form as

$$(\alpha : ac) = a\alpha c^{\mathrm{T}},$$

thus saving the $mn^2 + n$ operations required to compute everything explicitly. In fact, one of the basic tenets of Factorable Programming is that certain matrices need never be formed explicitly, since *all* required calculations can be performed with the dyadic structures. This of course offers a potentially great computational saving.

It only needs demonstrating, therefore, how to perform the multiplications in (3.6). Consider then multiplication between a generalized outer product matrix of order 3, a triad, and a two-dimensional matrix in three-space. Since there are three possible orientations for a two-dimensional matrix in three-space, one could guess that there are six ways to perform the multiplication depending on whether the matrix is pre- or post-multiplying the three-dimensional array. This is of course the case, and these multiplications are illustrated in Fig. 10, where in each case the matrix post-multiplies each similarly oriented matrix-slice of the three-dimensional array. A similar situation obtains for pre-multiplication.

Just as with the multiplication of a dyad and a matrix, these three multiplications are simplified when the three-dimensional array is stored as a triadic and it is desired to store the result as a triadic also. Let one of the triads be $(\alpha : abc)$ and the matrix be F, and consider the post-multiplication of each matrix-slice by F. Then the result of the multiplications in Fig. 10 is just

(i) $(\alpha : a[b^{\mathrm{T}}F]^{\mathrm{T}}c)$,

(ii) $(\alpha : ab[c^{\mathrm{T}}F]^{\mathrm{T}})$, and

(iii) $(\alpha : ab[c^{\mathrm{T}}F]^{\mathrm{T}})$.

(That which appears at first to be an error in (iii) is in fact a result of the simple fact that the vector in the third position, associated with the third dimension, is

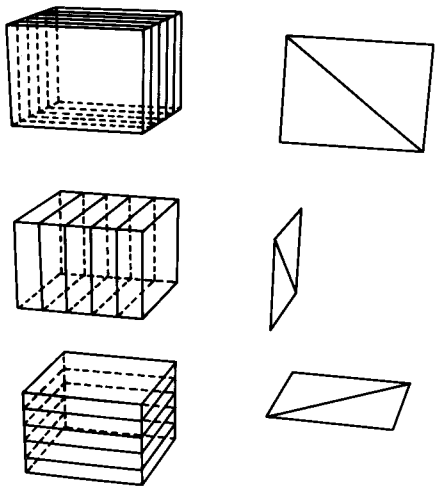

Fig. 10. Three ways to multiply a three-dimensional array by a matrix.

involved in defining two sets of slabs: one set with the vector a in the first dimension, and one set with the vector b in the second dimension).

One use of this ability to multiply polyads and matrices is in further exploiting the polyadic structure of the matrices in (3.6). This was used, for example, in a recent application of this technique to an unconstrained problem for the U.S. Department of Energy. For an unconstrained problem, (3.6) becomes

$$\nabla^2_{\varepsilon\varepsilon}x = -(\nabla^2 f)^{-1}[\nabla^3 f \nabla_\varepsilon\{x\}\nabla_\varepsilon x + \nabla^3_{\varepsilon xx} f \nabla_\varepsilon x + \nabla^3_{x\varepsilon x} f \nabla_\varepsilon\{x\} + \nabla^3_{\varepsilon\varepsilon x} f].$$

In this particular problem, all but the leading term inside the brackets vanished, leaving

$$\nabla^2_{\varepsilon\varepsilon}x = -(\nabla^2 f)^{-1}[\nabla^3 f \nabla_\varepsilon\{x\}\nabla_\varepsilon x].$$

Also in this problem it turned out that the triadic structure of $\nabla^3 f$ was naturally symmetric, and thus can be written

$$\nabla^3 L = \sum_{j=1}^{n} (\beta_j : b_j b_j b_j).$$

This by the way is not unusual. Many large problems have a natural symmetric polyadic structure. The objective function for the chemical equilibrium problem is another example.

For the DOE problem above, the Hessian inverse was also factorable and was written

$$-(\nabla^2 f)^{-1} = \sum_{i=1}^{m} a_i \alpha_i c_i^{\mathrm{T}} = \sum_{i=1}^{m} (\alpha_i : a_i c_i).$$

Then, for this unconstrained case,

$$\nabla^2_{\varepsilon\varepsilon}x = \sum_{i=1}^{m} (\alpha_i : a_i c_i) \sum_{j=1}^{m} (\beta_j : b_j b_j b_j)\nabla_\varepsilon\{x\}\nabla_\varepsilon x.$$

Application of (ii) yields

$$\nabla^2_{\varepsilon\varepsilon} x = \sum_{i=1}^{m} (\alpha_i : a_i c_i) \sum_{i=1}^{m} (\beta_j : b_j [b_j^{\mathrm{T}} \nabla_\varepsilon x] b_j) \nabla_\varepsilon \{x\},$$

and application of (iii) yields

$$\nabla^2_{\varepsilon\varepsilon} x = \sum_{i=1}^{m} (\alpha_i : a_i c_i) \sum_{j=1}^{m} (\beta_j : b_j [b_j^{\mathrm{T}} \nabla_\varepsilon x][b_j^{\mathrm{T}} \nabla_\varepsilon x]).$$

Similar rules obtain for premultiplication and yield, using the associative law also,

$$\nabla^2_{\varepsilon\varepsilon} x = \sum_{i=1}^{m} \sum_{j=1}^{m} (\alpha_i \beta_j [c_i^{\mathrm{T}} b_j] : a_i [b_j^{\mathrm{T}} \nabla_\varepsilon x][b_j^{\mathrm{T}} \nabla_\varepsilon x]),$$

the efficiency of which should be apparent. If it were desired, as it was in the Department of Energy problem, to produce each frontal slab of this in turn, the kth of these is given by

$$\sum_{i=1}^{m} \sum_{j=1}^{n} (\alpha_i \beta_j [a_i^{\mathrm{T}} b_j] a_{i,k} : [b_j^{\mathrm{T}} \nabla_\varepsilon x][b_j^{\mathrm{T}} \nabla_\varepsilon x]),$$

where, as before, $a_{i,k}$ denotes the kth element of the vector a_i.

3.6. Parameter tensors of the optimal value function

In this section, formulae are developed for the tensors (through order 3) of the optimal value function $f^*(\varepsilon) = f[x(\varepsilon), \varepsilon]$ of problem $P(\varepsilon)$ given in (3.1). Armacost and Fiacco (1974) were the first to extend basic first-order results for the right-hand-side perturbation problem to the general parametric problem in (3.1), and also developed the second-order results given below in theorem 5. Fiacco (1983) gives a complete treatment of all cases for all the variations of (3.1). Our interest is in providing results for the third-order tensor of $f^*(\varepsilon)$ and we begin with the first and second-order cases. Theorem 5 is due to Armacost and Fiacco (1974).

Theorem 5 (First- and second-order changes in $f^*(\varepsilon)$ for $P(\varepsilon)$). *Let x be a feasible point for $P(\hat{\varepsilon})$ and assume conditions* (i) *through* (iv) *in Theorem* 3. *Then, for ε in a sufficiently small neighborhood of $\hat{\varepsilon}$*

(a) $f^*(\varepsilon) = L^*,$

(b) $$\nabla_\varepsilon f^*(\varepsilon) = \nabla_\varepsilon L = \nabla_\varepsilon f - \sum_{i=1}^{m} u_i \nabla_\varepsilon g_i$$
$$= \nabla_\varepsilon f - u^T \nabla_\varepsilon g, \text{ and}$$

(c) $$\nabla^2_{\varepsilon\varepsilon} f^*(\varepsilon) = \nabla^2_{y\varepsilon} L \nabla_\varepsilon y + \nabla^2_{\varepsilon\varepsilon} L$$
$$= \nabla^2_{x\varepsilon} L \nabla_\varepsilon x - \nabla_\varepsilon g^T \nabla_\varepsilon u + \nabla^2_{\varepsilon\varepsilon} L.$$

Proof. See Fiacco (1983).

It is easy to see from their forms and from the results given in the previous section that $\nabla_\varepsilon f^*(\varepsilon)$ is monadic and $\nabla^2_{\varepsilon\varepsilon} f^*(\varepsilon)$ is dyadic. The result for the third-order tensor of $f^*(\varepsilon)$ is given in the following.

Theorem 6 (Third-order changes in $f^*(\varepsilon)$ for $P(\varepsilon)$). *If the conditions of Theorem* 5 *hold and all third-order partial derivatives in x and third-order partial derivatives in x and ε exist and are continuous in x and ε in a neighborhood of* $(\hat{x}, \hat{\varepsilon})$, *then*

$$\begin{aligned}\nabla^3_{\varepsilon\varepsilon\varepsilon} f^*(\varepsilon) &= \nabla^2_{y\varepsilon} L \nabla^2_{\varepsilon\varepsilon} y + \nabla^3_{yy\varepsilon} L \nabla_\varepsilon\{y\}\nabla_\varepsilon y + \nabla^3_{\varepsilon y\varepsilon} L \nabla_\varepsilon y + \nabla^3_{\varepsilon\varepsilon\varepsilon} L + \nabla_{y\varepsilon\varepsilon} L \nabla_\varepsilon y \\ &= \nabla^2_{x\varepsilon} L \nabla^2_{\varepsilon\varepsilon} x + \nabla^3_{xx\varepsilon} L \nabla_\varepsilon\{x\}\nabla_\varepsilon x + \nabla^3_{ux\varepsilon} L \nabla_\varepsilon\{u\}\nabla_\varepsilon x + \nabla^3_{\varepsilon x\varepsilon} L \nabla_\varepsilon x \\ &\quad - \nabla_\varepsilon g^{\mathrm{T}} \nabla^2_{\varepsilon\varepsilon} u - \nabla\{\nabla_\varepsilon g^{\mathrm{T}}\}\nabla_\varepsilon\{x\}\nabla_\varepsilon u - \nabla_\varepsilon\{\nabla_\varepsilon g^{\mathrm{T}}\}\nabla_\varepsilon u + \nabla^3_{\varepsilon\varepsilon\varepsilon} L,\end{aligned}$$

and $\nabla^3_{\varepsilon\varepsilon\varepsilon} f^*(\varepsilon)$ *is a triadic.*

Proof. Straightforward differentiation of (c) in Theorem 5 gives the formula for $\nabla^3_{\varepsilon\varepsilon\varepsilon} f^*(\varepsilon)$. The proof of its triadic structure is also straightforward using the results of the previous sections.

3.7. Second-order sensitivity results in use

The direct use of first- and second-order sensitivity results is in estimating the solution and multiplier vectors for $P(\varepsilon)$ as the problem is perturbed away from $P(\hat{\varepsilon})$. This estimation is done using the Taylor series approximations to two and three terms:

$$y(\varepsilon) \approx y(\hat{\varepsilon}) + \nabla_\varepsilon y(\hat{\varepsilon})(\varepsilon - \hat{\varepsilon}), \tag{3.12}$$

$$y(\varepsilon) \approx y(\hat{\varepsilon}) + \nabla_\varepsilon y(\hat{\varepsilon})(\varepsilon - \hat{\varepsilon}) + \tfrac{1}{2}\nabla^2_{\varepsilon\varepsilon} y(\hat{\varepsilon})(\varepsilon - \hat{\varepsilon})(\varepsilon - \hat{\varepsilon}), \tag{3.13}$$

where the multiplication in the third term on the rhs of (3.13) is understood to be inner product multiplication that reduces the dimension of $\nabla^2_{\varepsilon\varepsilon} y$.

To illustrate this idea as well as some of the other ideas in this section, consider the following parameterized nonlinear programming problem:

$$\underset{x \in R^n}{\text{minimize}} \quad f(x, \varepsilon) = x^{\mathrm{T}} x + \varepsilon_1 \exp(\varepsilon_2 a^{\mathrm{T}} x).$$

At $\hat{\varepsilon} = 0$, the solution by inspection is at $\hat{x} = 0$. Since $L(x, u, \varepsilon) = f(x, \varepsilon)$ for this problem, the first-order sensitivity equation, $Y = -M^{-1}N$, reduces to

$$\nabla_\varepsilon x(\varepsilon) = -(\nabla^2 f)^{-1} \nabla^2_{\varepsilon x} f.$$

Also

$$\nabla f = 2x + [\varepsilon_1 \varepsilon_2 \exp(\varepsilon_2 a^{\mathrm{T}} x)]a,$$

$$\nabla^2 f = 2I + a[\varepsilon_1 \varepsilon_2^2 \exp(\varepsilon_2 a^{\mathrm{T}} x)]a^{\mathrm{T}},$$

and

$$\nabla^2_{\varepsilon x} f = a[\exp(\varepsilon_2 a^{\mathrm{T}} x)][\varepsilon_2,\ \varepsilon_1 \varepsilon_2 a^{\mathrm{T}} x + \varepsilon_1].$$

Since $(2I)^{-1} = \frac{1}{2} I$, $(\nabla^2 f)^{-1}$ can readily be obtained using the Sherman-Woodbury-Morrison formula (see McCormick, 1983, p. 70) that gives the inverse of a matrix perturbed by a dyad. This formula is

$$(A + ucv^{\mathrm{T}})^{-1} = A^{-1} - A^{-1} u[c(1 + v^{\mathrm{T}} A^{-1} uc)^{-1}] v^{\mathrm{T}} A^{-1}.$$

Using this, and letting $c = \varepsilon_1 \varepsilon_2^2 \exp(\varepsilon_2 a^{\mathrm{T}} x)$,

$$(\nabla^2 f)^{-1} = (2I + aca^{\mathrm{T}})^{-1} = \tfrac{1}{2} I - a\left[\left(\frac{4}{c} + 2a^{\mathrm{T}} a\right)^{-1}\right] a^{\mathrm{T}}.$$

Evaluating these at $\hat{x} = 0$ and $\hat{\varepsilon} = 0$ yields

$$(\nabla^2 f)^{-1} = \tfrac{1}{2} I,$$

and

$$\nabla^2_{\varepsilon x} f = 0.$$

Therefore

$$\nabla_{\varepsilon} x(\varepsilon) = 0,$$

and the first-order sensitivity analysis approximation (3.12) gives no additional information as the problem is perturbed away from $\hat{\varepsilon} = 0$. This problem is ideally suited then for a second-order sensitivity analysis using equations (3.13) and (3.6). But since $\nabla_{\varepsilon} x(\varepsilon) = 0$, (3.6) reduces to

$$\nabla^2_{\varepsilon\varepsilon} x(\varepsilon) = -(\nabla^2 f)^{-1} \nabla^3_{\varepsilon\varepsilon x} f.$$

Using the formulae developed in the proof of Theorem 3,

$$\begin{aligned}
\nabla^3_{\varepsilon\varepsilon x} f &= \nabla_{\varepsilon}(\exp[\varepsilon_2 a^{\mathrm{T}} x] : a[0,\ \varepsilon_1 \varepsilon_2 a^{\mathrm{T}} x]^{\mathrm{T}}) + \nabla_{\varepsilon}(\exp[\varepsilon_2 a^{\mathrm{T}} x] : a[\varepsilon_2,\ \varepsilon_1]^{\mathrm{T}}) \\
&= (\exp[\varepsilon_2 a^{\mathrm{T}} x] : a[\varepsilon_2,\ \varepsilon_1 \varepsilon_2 a^{\mathrm{T}} x]^{\mathrm{T}}[0,\ a^{\mathrm{T}} x]^{\mathrm{T}}) \\
&\quad + (\exp[\varepsilon_2 a^{\mathrm{T}} x] : a[0,\ a^{\mathrm{T}} x]^{\mathrm{T}}[\varepsilon_2,\ \varepsilon_1]^{\mathrm{T}}) \\
&\quad + (\exp[\varepsilon_2 a^{\mathrm{T}} x] : a(\varepsilon_2,\ \varepsilon_1 \varepsilon_2 a^{\mathrm{T}} x]^{\mathrm{T}}[0,\ a^{\mathrm{T}} x]^{\mathrm{T}}) \\
&\quad + (\exp[\varepsilon_2 a^{\mathrm{T}} x] : a[0,\ 1]^{\mathrm{T}}[1,\ 0]^{\mathrm{T}}) \\
&\quad + (\exp[\varepsilon_2 a^{\mathrm{T}} x] : a[1,\ 0]^{\mathrm{T}}[0,\ 1]^{\mathrm{T}}).
\end{aligned}$$

Notice the triadic nature of this tensor and the frequency of occurrence of certain terms. Evaluating these at $\hat{x} = 0$ and $\hat{\varepsilon} = 0$ yields

$$\nabla^3_{\varepsilon\varepsilon x} f = (1 : ae_2 e_1) + (1 : ae_1 e_2),$$

where e_i is the ith unit vector in R^n. The second partial derivatives with respect to ε of the solution vector are given by

$$\nabla^2_{\varepsilon\varepsilon} x(\varepsilon) = -(\nabla^2 f(\hat{x}))^{-1} \nabla^3_{\varepsilon\varepsilon x} f(\hat{x}) = -\tfrac{1}{2} I[(1 : ae_2e_1) + (1 : ae_1e_2)]$$

$$= (-\tfrac{1}{2} : ae_2e_1) + (-\tfrac{1}{2} : ae_1e_2).$$

For this unconstrained example (3.13) reduces to the second-order estimation

$$x(\varepsilon) \approx x(\hat{\varepsilon}) + \nabla_\varepsilon x(\hat{\varepsilon})(\varepsilon - \hat{\varepsilon}) + \tfrac{1}{2}\nabla^2_{\varepsilon\varepsilon} x(\hat{\varepsilon})(\varepsilon - \hat{\varepsilon})(\varepsilon - \hat{\varepsilon}),$$

which at $\hat{\varepsilon} = 0$ becomes

$$x(\varepsilon) \approx x(0) + \nabla_\varepsilon x(0)\varepsilon + \tfrac{1}{2}\nabla^2_{\varepsilon\varepsilon} x(0)\varepsilon\varepsilon.$$

Since $x(0)$ is the solution to the original problem, $x(0) = \hat{x} = 0$. Furthermore, $\nabla_\varepsilon x(0)$ was shown above to vanish also. Thus

$$\begin{aligned} x(\varepsilon) &\approx \tfrac{1}{2}\nabla^2_{\varepsilon\varepsilon} x(0)\varepsilon\varepsilon \\ &= \tfrac{1}{2}[(-\tfrac{1}{2} : ae_2e_1) + (-\tfrac{1}{2} : ae_1e_2)]\varepsilon\varepsilon \\ &= (-\tfrac{1}{4}\varepsilon_2\varepsilon_1 : a) + (-\tfrac{1}{4}\varepsilon_1\varepsilon_2 : a) \\ &= (-\tfrac{1}{2}\varepsilon_1\varepsilon_2 : a). \end{aligned} \tag{3.14}$$

For a more concrete example of this technique, let $a = (1, 2)^T$ and perturb the problem by $\varepsilon = (-1.902, 0.1)^T$. The solution to this new problem is calculated in Jackson and McCormick (1986) using Halley's third-order method of tangent hyperbolas, and is $(0.1, 0.2)^T$. However, an estimate is given by (3.14), without having to solve the new nonlinear programming problem. The approximation is

$$x(\varepsilon) \approx ((-0.5)(-1.902)(0.1) : [1, 2]^T)$$

$$= (0.0951, 0.1902)^T,$$

which of course is much better than the first-order approximation, $x(\varepsilon) = 0$.

Another use of the second-order sensitivity formulae is in solving implicitly defined optimization problems. Consider for example the optimization problem

$$\max_{(p,q)} F(p, q) = x^*(p, q) + y^*(p, q) + pq$$

where (x^*, y^*) is defined implicitly as the solution of

$$\min_{(x,y)} G(x, y) = (x - y + 3pq)^2 + (x - (p - q)^2)^2.$$

The analytic solution of this problem is easily obtained as

$$x^*(p, q) = (p - q)^2,$$

$$y^*(p, q) = (p - q)^2 + 3pq.$$

Substituting, the other problem becomes

$$\max_{(p,q)} 2p^2+2q^2,$$

with solution $(p^*, q^*) = (0, 0)$.

Let $\varepsilon = (p, q)^T$. The solution of the maximization problem will be accomplished in one iteration of Newton's method using the second-order sensitivity formulas. In order to achieve this, it is required to compute

$$\varepsilon_0 - [\nabla^2_{\varepsilon\varepsilon} F(\varepsilon_0)]^{-1} \nabla_\varepsilon F(\varepsilon_0).$$

Now

$$\nabla_\varepsilon F = \nabla_\varepsilon x^* + \nabla y^* + [q, p]^T,$$

and

$$\nabla^2_{\varepsilon\varepsilon} F = \nabla^2_{\varepsilon\varepsilon} x^* + \nabla^2_{\varepsilon\varepsilon} y^* + \begin{vmatrix} 0 & 1 \\ 1 & 0 \end{vmatrix}.$$

Let z denote $[x, y]^T$. Then from first-order sensitivity analysis

$$\nabla_\varepsilon z^* = -(\nabla^2_{\varepsilon\varepsilon} G)^{-1} \nabla_{\varepsilon z} G.$$

Now

$$\nabla^2_{zz} G = \begin{vmatrix} 1 \\ -1 \end{vmatrix} (2)[1, -1] + \begin{vmatrix} 1 \\ 0 \end{vmatrix} (2)[1, 0]$$

and

$$\nabla^2_{\varepsilon z} G = \begin{vmatrix} 1 \\ -1 \end{vmatrix} (2)[3q, 3p] + \begin{vmatrix} 1 \\ 0 \end{vmatrix} (4)(pq)[1, -1].$$

The most natural representation of $(\nabla^2_{zz} G)^{-1}$ is in dyadic form and is

$$\begin{vmatrix} 0 \\ -1 \end{vmatrix} (\tfrac{1}{2})[0, -1] + \begin{vmatrix} 1 \\ 1 \end{vmatrix} (\tfrac{1}{2})[1, 1].$$

The second-order sensitivity formula (3.6) can be rewritten (conceptually) in this case as

$$\nabla^2_{\varepsilon\varepsilon} z^* = -(\nabla^2_{\varepsilon\varepsilon} G)^{-1} [(\nabla^3_{\varepsilon zz} G)(\nabla_\varepsilon z^*) + (\nabla^3_{zzz} G)(\nabla_\varepsilon z^*)^2 + \nabla^3_{\varepsilon\varepsilon z} G + (\nabla^3_{z\varepsilon z} G)(\nabla_\varepsilon z^*)].$$

For this problem the only term multiplying the inverse which does not vanish is $\nabla^3_{\varepsilon\varepsilon z} G$, a $(2 \times 2 \times 2)$ matrix. Its triadic form is

$$\nabla^3_{\varepsilon\varepsilon z} G = (6 : [1, -1]^T [1, 0]^T [0, 1]^T) + (6 : [1, -1]^T [0, 1]^T [1, 0]^T) + (-4 : [1, 0]^T [1, -1]^T [1, -1]^T).$$

Suppose in this case that $[p_0, q_0] = [2, 1]$. Then

$$\nabla_\varepsilon Z_0^* = -\begin{vmatrix} 0 \\ -1 \end{vmatrix} (\tfrac{1}{2})[0, -1] + \begin{vmatrix} 1 \\ 1 \end{vmatrix} (\tfrac{1}{2})[1, 1] + \begin{vmatrix} 1 \\ -1 \end{vmatrix} (2)[3, 6] + \begin{vmatrix} 1 \\ 1 \end{vmatrix} [1, -1]. \tag{4}$$

Therefore

$$\nabla_\varepsilon x_0^* = [2, -2]^T, \nabla_\varepsilon y_0^* = [5, 4]^T,$$

and thus

$$\nabla_\varepsilon F_0 = [2, -2]^T + [5, 4]^T + [1, 2]^T = [8, 4]^T.$$

The second-order sensitivity formulas yield

$$\begin{aligned} -[\nabla_{zz}^2 G]^{-1}[\nabla_{\varepsilon\varepsilon z}^3 G] = &-[(\tfrac{1}{2}:[0, -1]^T[0, -1]^T) + (\tfrac{1}{2}:[1, 1]^T[1, 1]^T)] \\ &* [(6:[1, -1]^T[1, 0]^T[0, 1]^T) \\ &+ (6:[1, -1]^T[0, 1]^T[1, 0]^T) \\ &+ (-4:[1, 0]^T[1, -1]^T[1, -1]^T)]. \end{aligned}$$

To illustrate the computation, one of the six product terms will be computed:

$$\begin{aligned} &(-\tfrac{1}{2}:[0, 1]^T[0, -1]^T) * (6:[1, -1]^T[1, 0]^T[0, 1]^T) \\ &\quad = (-(1)(\tfrac{1}{2})(6)[0, -1)[1, -1]^T:[0, -1]^T[1, 0]^T[0, 1]^T) \\ &\quad = (-3:[0, 1]^T[1, 0]^T[0, 1]^T). \end{aligned}$$

In all, the resulting triadic form has the following terms:

$$\begin{aligned} &(-3:[0, -1]^T[1, 0]^T[0, 1]^T) + (-3:[0, -1]^T[0, 1]^T[1, 0]^T) \\ &\quad + (2:[1, 1]^T[1, -1]^T[1, -1]^T[1, -1]^T). \end{aligned}$$

From this,

$$\nabla_{\varepsilon\varepsilon}^2 x_0^* = (2:[1, -1]^T[1, -1]^T)$$

and

$$\nabla_{\varepsilon\varepsilon}^2 y_0^* = (2:[1, -1]^T[1, -1]^T) + (3:[0, 1]^T[1, 0]^T) + (3:[1, 0]^T[0, 1]^T)$$

Thus

$$\nabla_{\varepsilon\varepsilon}^2 F_0 = \nabla_{\varepsilon\varepsilon}^2 x_0^* + \nabla_{\varepsilon\varepsilon}^2 y_0^* + \begin{vmatrix} 0 & 1 \\ 1 & 0 \end{vmatrix} = \begin{vmatrix} 4 & 0 \\ 0 & 4 \end{vmatrix}.$$

Combining,

$$\varepsilon_0 - (\nabla_{\varepsilon\varepsilon}^2 F_0)^{-1}(\nabla_\varepsilon F_0) = \begin{vmatrix} 2 \\ 1 \end{vmatrix} - \begin{vmatrix} 4 & 0 \\ 0 & 4 \end{vmatrix}^{-1} \begin{vmatrix} 8 \\ 4 \end{vmatrix} = \begin{vmatrix} 0 \\ 0 \end{vmatrix}$$

as desired.

References

R.L. Armacost and A.V. Fiacco, "Computational experience in sensitivity analysis for nonlinear programming," *Mathematical Programming* 6 (1974) 301-326.

R.L. Armacost and A.V. Fiacco, "Sensitivity analysis for parametric nonlinear programming using penalty methods," in: *Computers and Mathematical Programming*, National Bureau of Standards Special Publication 502 (1978) pp. 261-269.

G.A. Bliss, *Lectures on the calculus of variations* (University of Chicago Press, Chicago, 1946).

R.S. Dembo, "Sensitivity analysis in geometric programming," *Journal of Optimization Theory and Applications* 37 (1982) 1-21.

J.E. Falk, "Global solutions of signomial problems," Technical report T-274, George Washington University, Department of Operations Research, (Washington, DC, 1973).

J.E. Falk and R.M. Soland, "An algorithm for separable nonconvex programming problems," *Management Science* 15 (1969) 550-569.

A.V. Fiacco and G.P. McCormick, *Nonlinear Programing: Sequential Unconstrained Minimization Techniques* (Wiley, New York, 1968).

A.V. Fiacco, Sensitivity analysis for nonlinear programming using penalty methods," *Mathematical Programming* 10 (1976) 287-311.

A.V. Fiacco, "Nonlinear programming sensitivity analysis results using strong second order assumptions," in: L.C.W. Dixon and G.P. Szego, eds., *Numerical Optimization of Dynamic Systems* (North-Holland, Amsterdam, 1980), pp. 327-348.

A.V. Fiacco, *Introduction to Sensitivity and Stability Analysis in Nonlinear Programming* (Academic Press, New York, 1983).

J.J. Filliben, "DATAPLOT—an interactive high level language for graphics, nonlinear fitting, data analysis and mathematics," *Computer Graphics* 15 (1981) 199-213.

A. Graham, *Kronecker Products and Matrix Calculus with Applications* (Wiley, New York, 1981).

K.L. Hoffman, "NUGLOBAL—User Guide," Technical report TM-64866, George Washington University, Department of Operations Research (Washington, DC, 1975).

R.H.F. Jackson, "Tensors, Polyads, and High-Order Methods in Factorable Programming," Dissertation, The George Washington University, Department of Operations Research (Washington, DC, 1983).

R.H.F. Jackson and G.P. McCormick, "The polyadic structure of factorable function tensors with application to high-order minimization techniques," *Journal of Optimization Theory and Applications* 51 (1986) 63-94.

R.H.F. Jackson, G.P. McCormick and A. Sofer, "FACTNLS, a factorable programming system for nonlinear least squares problems," National Bureau Standards Working Paper, to appear.

J. Kyparisi, "Sensitivity and stability for nonlinear and geometric programming: theory and applications. Dissertation, The George Washington University, Department of Operations Research (Washington, DC, 1983).

S.G. Leaver, "Computing global maximum likelihood parameter estimates for product models for frequency tables involving indirect observation," Dissertation, The George Washington University, Department of Operations Research (Washington, DC, 1984).

A. Linneman, "Higher-order necessary conditions for infinite and semi-infinite optimization," *Journal of Optimization Theory and Applications* 38 (1982) 483-511.

W.H. Marlow, *Mathematics for Operations Research* (Wiley, New York, 1978).

G.P. McCormick, "Second order conditions for constrained minima," *SIAM Journal of Applied Mathematics* 15 (1967) 37-47.

G.P. McCormick, "Computability of global solutions to factorable nonconvex programs: Part I — Convex underestimating problems," *Mathematical Programming* 10 (1976) 147-175.

G.P. McCormick, *Nonlinear Programming: Theory, Algorithms and Applications* (Wiley, New York, 1983).

G.P. McCormick, "Global solutions to factorable nonlinear optimization problems using separable programming techniques," Technical Report NBSIR 85-3206, National Bureau of Standards (Gaithersburg, MD, 1985).

L. Pennisi, "An indirect proof of the problem of Lagrange with differential inequalities as added side conditions," *Transactions of the American Mathematical Society* 74 (1953) 177-198.