Automatic differentiation and iterative processes

Jean Charles Gilbert

AUTOMATIC DIFFERENTIATION AND ITERATIVE PROCESSES*

Jean Charles GILBERT†

April 1991 (Revised in January 1992)

We identify a class of iterative processes that can be used in the definition of a function while preserving the good behavior of automatic differentiation codes on this function. By iterative process, we mean a process where the number of iterations is determined by a stopping criterion, which can depend on the independent variables. By good behavior, we mean that the derivatives will be calculated correctly, asymptotically. The class contains the Newton method.

Key words: automatic differentiation, iterative process, Newton's method.

1 Introduction

Automatic differentiation is a technique used for generating computer programs that compute the value of the derivatives of a function

$$f : \mathbb{R}^n \to \mathbb{R}^m : x \mapsto f(x),$$

given by an original computer program. See [8, 4, 3] for an introduction to the subject and [6] for discussions of more recent developments and applications. The variables forming the vector x are called *independent*. All the other variables used in the program evaluating $f(x)$ are called *intermediate*.

The programs generated by automatic differentiation calculate the value of the derivatives correctly in a wide variety of situations. However, the value may be false when the original program uses an iterative process to define the function f. Therefore, researchers have focused attention on finding a means to determine whether a given iterative process can be treated correctly [7, 2, 3].

In the automatic differentiation framework, we call an *iterative process* a part of a computer program whose instructions are executed several times until a *stopping criterion* is reached. This stopping criterion can depend on some independent or intermediate variables. As a result, the number k of iterations realized in a particular run of the program is not known beforehand, but depends on the value x given to the independent variables: $k = k(x)$. A typical example of such a situation occurs when an iterative method, such as Newton's method, is used to calculate an implicit function, say, the solution of a nonlinear equation.

Let us assume that the program contains a single iterative process. We denote by $f_k : \mathbb{R}^n \to \mathbb{R}^m$ the function realized by the program when k iterations are executed. We

*This work was supported in part by the Centre National d'Etudes Spaciales (CNES), 3 avenue Édouard Belin, 31055 Toulouse (France).

†INRIA, Domaine de Voluceau, Rocquencourt, B.P. 105, 78153 Le Chesnay Cedex (France).

shall restrict our attention to the case when the iterative process aims at making f_k close to a certain function $f : \mathbb{R}^n \to \mathbb{R}^m$. If the algorithm is well conceived, we can expect that the sequence of functions $\{f_k\}_{k\geq 0}$ will locally *converge pointwise* to f, meaning that

$$\forall\, x \in \Omega,\ f_k(x) \to f(x), \quad \text{when} \quad k \to \infty, \tag{1}$$

where Ω is some open set in $\mathbb{R}^n$. It is important to note here that if automatic differentiation is used with such a program and if, for a particular x, k iterations are executed by the original program, the derivative computed at x by the generated program will be $f_k'(x)$ and not $\xi'(x)$, the derivative of the function $\xi(x) := f_{k(x)}(x)$. It is indeed the function representing all the operations executed in a computation that is differentiated. Note also that this function ξ will probably be discontinuous.

Now, we can wonder to what extent the fact that $f_k(x)$ is close to $f(x)$ (a situation detected by the stopping criterion) implies that $f_k'(x)$ will be close to $f'(x)$. Since this question is difficult to answer, we instead ask whether we can expect that some *convergence* of f_k to f implies, for instance, the local pointwise convergence of f_k' to f':

$$\forall\, x \in \Omega,\ f_k'(x) \to f'(x), \quad \text{when} \quad k \to \infty. \tag{2}$$

This is a theoretical question since it concerns asymptotic behavior, which has little to do with practical situations; but it has the major advantage of being mathematically tractable. Moreover, an asymptotic analysis may give some insight into what can happen in practical finite situations. Our aim, therefore, is to identify some of the processes for which property (1) implies property (2).

One can argue that this is not the right approach because, in optimization for instance, what is really needed is the gradient ($m = 1$) of the computed function ξ. In this regard, we can say that the function ξ is usually not continuous, and therefore its analysis is difficult. On the other hand, this analysis would require more information on the iterative process, in particular on the stopping criterion (obviously, if this criterion is always satisfied, we would have $\xi = f_0$), which we do not require in our simple study. Finally, the user may really wish to compute $f'(x)$, not $\xi'(x)$ (when this derivative exists).

Generally speaking, there is no reason for property (1) to imply property (2). Worse, even if f_k' converges pointwise, it can be to a function different from f'. In fact, it is well known that this situation would be avoided if f_k' would converge locally uniformly; see, for example, Schwartz [10, Chapter IV, Theorem 111]. As an illustration, let us consider the following example [2] where $n = m = 1$:

$$f_k(x) = xe^{-kx^2}. \tag{3}$$

The sequence of functions $\{f_k\}$ converges *uniformly* to the function $f \equiv 0$, $f_k'(x) \to 0$ if $x \neq 0$, but $f_k'(0) = 1$ for any k. As a result, the pointwise limit of f_k' differs from the function $f' \equiv 0$. Note that when the components of the functions f_k are smooth and *convex*, (1) implies (2); see Rockafellar [9, Theorem 24.5].

In Section 2, however, Proposition 1 shows that, for a wide class of convergent iterative processes, the derivative computed by automatic differentiation will be *asymptotically* correct, that is, close to the derivative of f.

2 A Class of Safe Iterative Processes

To make Proposition 1 useful for practical situations, we need to particularize the problem and to focus our attention on that part of the code where the iterative process occurs.

Let us separate the variables existing just before beginning the iterative process in a triplet (y_0, a_0, u), where the vector u gathers the variables that will not be modified during the iterations, while the vectors y_0 and a_0 gather the variables that will be modified at each iteration to become y_k and a_k *after* iteration k. We assume that the independent variables x are not updated, so that they are part of the variables u. We distinguish between auxiliary variables (a_k) that can be dismissed in the definition of the function f_k and those (y_k) that are necessary or that we wish to use in this definition. In other words, it must be possible to define $f_k(x)$ as a function of y_k and u, avoiding the dependence in a_k:

$$f_k(x) = \psi(y_k, u). \tag{4}$$

The dependence of f_k on x derives from that of y_k and u. We shall see later the advantage of expressing f_k independently of some auxiliary variables a_k with regard to the analysis of the *convergence* of $f_k'(x)$ to $f'(x)$. But this structure in the variables need not be known by the automatic differentiator, whose role is always to "differentiate blindly."

Next, we also suppose that the update of the variables $y_k \in \mathbb{R}^q$ during iteration $k+1$ can be described by a smooth operator Φ defined on an open set of $\mathbb{R}^q \times \mathbb{R}^p$ with value in $\mathbb{R}^q$:

$$y_{k+1} = \Phi(y_k, u), \quad k \geq 0. \tag{5}$$

Note that, in the program, the initial iterate y_0 may be considered as a function of u. By induction, rule (5) makes y_k a function of u alone; thus, (5) can be rewritten as follows:

$$y_{k+1}(u) = \Phi(y_k(u), u), \quad k \geq 0. \tag{6}$$

We assume that the iterative process converges, that is, that the sequence $\{y_k(u)\}_{k\geq 0}$ in $\mathbb{R}^q$ converges to a vector $y_* = y_*(u)$ (u is fixed). Then, $y_*(u)$ is necessarily a *fixed point* of $\Phi(\cdot, u)$, as we have from (6)

$$y_*(u) = \Phi(y_*(u), u). \tag{7}$$

Therefore, the map $u \mapsto y_*(u)$ appears as an implicit function of the equation $y = \Phi(y, u)$.

According to (4), we have for fixed x and for K tending to infinity

$$f_k(x) \to f(x) := \psi(y_*(u), u). \tag{8}$$

The question is whether $f_k'(x)$ converges to $f'(x)$. If we denote by ψ_y' and ψ_u' the partial derivatives of ψ, we have from (4) and (8)

$$f_k'(x) = \psi_y'(y_k(u), u) \cdot y_k'(u) \cdot u'(x) + \psi_u'(y_k(u), u) \cdot u'(x), \tag{9}$$

$$f'(x) = \psi_y'(y_*(u), u) \cdot y_*'(u) \cdot u'(x) + \psi_u'(y_*(u), u) \cdot u'(x). \tag{10}$$

With our assumptions, we see that the convergence of the derivatives $f_k'(x) \to f'(x)$ will follow from the convergence of $y_k'(u) \to y_*'(u)$. Proposition 1 gives conditions on Φ that

guarantee this convergence. A similar result has been given by H. Fischer [1], where the iterative process is used for solving a linear system. Below, $\mathcal{L}(E,F)$ denotes the space of linear operators between the linear spaces E and F.

Proposition 1. *Suppose that the application Φ defined on the product of open sets $\omega_y \times \omega_u \subset \mathbb{R}^q \times \mathbb{R}^p$ with value in $\mathbb{R}^q$ is continuously differentiable and that its derivative map $\Phi' : \omega_y \times \omega_u \to \mathcal{L}(\mathbb{R}^q \times \mathbb{R}^p, \mathbb{R}^q)$ is Lipschitz continuous. Suppose also that the initial iterate y_0 is a differentiable function of u on ω_u and that, for fixed $u \in \omega_u$, the sequence $\{y_k(u)\}_{k\geq 0}$ defined by (6) is in ω_y and converges to a point $y_* = y_*(u) \in \omega_y$. If the spectral radius ρ of the restriction $\Phi'_y(y_*,u)$ of $\Phi'(y_*,u)$ to $\mathbb{R}^q$ (partial derivative with respect to y) satisfies*

$$\rho\left(\Phi'_y(y_*,u)\right) < \tau < 1, \tag{11}$$

then

(*i*) *the convergence of the sequence $\{y_k\}_{k\geq 0}$ is asymptotically linear; that is, there exist a vector norm $\|\cdot\|$ and an index k_0 such that $\|y_{k+1} - y_*\| \leq \tau \|y_k - y_*\|$, for all indices $k \geq k_0$; and*

(*ii*) *the sequence of derivatives $\{y'_k(u)\}$ converges to $y'_*(u)$.*

Proof. We first show that the convergence of y_k is asymptotically linear. According to the Taylor formula, we have

$$y_{k+1} - y_* = \Phi(y_k,u) - \Phi(y_*,u) = \int_0^1 \Phi'_y(y_* + t(y_k - y_*), u) \cdot (y_k - y_*)\, dt.$$

From (11), there exists a matrix norm subordinated to a vector norm, both denoted by $\|\cdot\|$, such that $\|\Phi'_y(y_*,u)\| < \tau < 1$. Then, as soon as y_k is sufficiently close to y_*, say for $k \geq k_0$, we have

$$\|\Phi'_y(y_* + t(y_k - y_*), u)\| \leq \tau, \quad \text{for} \quad t \in [0,1], \tag{12}$$

and therefore

$$\|y_{k+1} - y_*\| \leq \tau \, \|y_k - y_*\|, \quad \text{for} \quad k \geq k_0.$$

This shows (i).

Before proving (ii), we remark that the application $u \mapsto y_k(u)$ is differentiable because, by (6), it is the composition of k differentiable applications with the differentiable application $u \mapsto y_0(u)$. The application $u \mapsto y_*(u)$ is also differentiable because it is an implicit function of the equation $F(y,u) \equiv y - \Phi(y,u) = 0$ and because the conditions to apply the implicit function theorem are satisfied. To check this, first observe that the conditions of smoothness are fulfilled. Next, note that $F'_y(y_*,u) = I - \Phi'_y(y_*,u)$ is nonsingular, because if $\eta \in \mathbb{R}^q$ is such that $F'_y(y_*,u) \cdot \eta = 0$, then $\eta = \Phi'_y(y_*,u) \cdot \eta$ and, using (12), we have $\|\eta\| \leq \tau \|\eta\|$, which implies $\eta = 0$ (because $\tau < 1$).

Now, we can differentiate (6) and (7) with respect to u in a direction $v \in \mathbb{R}^p$. Writing $\delta_k = y'_k(u) \cdot v$ and $\delta_* = y'_*(u) \cdot v$, we have

$$\begin{aligned} \delta_{k+1} &= \Phi'_y(y_k,u) \cdot \delta_k + \Phi'_u(y_k,u) \cdot v, \\ \delta_* &= \Phi'_y(y_*,u) \cdot \delta_* + \Phi'_u(y_*,u) \cdot v. \end{aligned}$$

Subtracting each side of these identities, we get:

$$\delta_{k+1} - \delta_* = \Phi'_y(y_k, u) \cdot [\delta_k - \delta_*] + [\Phi'_y(y_k, u) - \Phi'_y(y_*, u)] \cdot \delta_* + [\Phi'_u(y_k, u) - \Phi'_u(y_*, u)] \cdot v.$$

From the Lipschitz continuity of Φ' and to (12), we obtain for sufficiently large k (say, for $k \geq k_1 \geq k_0$) and a positive constant C

$$\|\delta_{k+1} - \delta_*\| \leq \tau \|\delta_k - \delta_*\| + C\|y_k - y_*\|. \tag{13}$$

By induction and by using (i), one finds for $k \geq k_1$

$$\begin{aligned} \|\delta_{k+1} - \delta_*\| &\leq \tau^{k-k_1+1}\|\delta_{k_1} - \delta_*\| + C\left(\sum_{i=k_1}^{k} \tau^{k-i}\|y_i - y_*\|\right) \\ &\leq \tau^{k-k_1+1}\|\delta_{k_1} - \delta_*\| + C\,(k-k_1+1)\tau^{k-k_1}\|y_{k_1} - y_*\|. \end{aligned}$$

Since $\tau < 1$, we have that $\tau^{k-k_1+1} \to 0$ and that $(k-k_1+1)\tau^{k-k_1} \to 0$, when k goes to infinity. These results prove that $\delta_k = y'_k(u) \cdot v$ converges to $\delta_* = y'_*(u) \cdot v$. Since this conclusion is true for all $v \in \mathbb{R}^p$, we deduce (in finite dimension) that $y'_k(u) \to y'_*(u)$ in $\mathcal{L}(\mathbb{R}^p, \mathbb{R}^q)$. □

Note that we cannot get rid of a hypothesis like (11). Indeed, the iterations (3) give a counterexample for this hypothesis, because these can be written $f_{k+1} = \Phi(f_k, x)$, with

$$\Phi(f, x) = fe^{-x^2}, \qquad f_0(x) = x,$$

and $\Phi'_f(f, 0) = e^{-x^2}\Big|_{x=0} = 1$ does not satisfy (11).

This result is essentially qualitative and describes only an *asymptotic* behavior. In particular, even in the conditions of Proposition 1, if for a given $x_1 \in \mathbb{R}^n$ the process stops at iteration k_1 because the stopping criterion is able to assert that $f_{k_1}(x_1)$ is close to $f(x_1)$, there is no reason for $f'_{k_1}(x_1)$ (with index k_1) to be close to $f'(x_1)$. The closeness of the derivatives may be obtained only for an index k_2 much larger than k_1.

3 Application to Newton's Method

As an application of the preceding result, let us consider the case when Newton's iterations are used to calculate an implicit function, say, the solution $y : u \in \mathbb{R}^p \mapsto y(u) \in \mathbb{R}^q$ of

$$F(y, u) = 0.$$

Suppose that the application $F : \omega_y \times \omega_u \to \mathbb{R}^q$ is sufficiently smooth and that the initial iterate y_0 is obtained as a smooth function of u. With Newton's method, the iterates are calculated by the following formula:

$$y_{k+1} = y_k - F'_y(y_k, u)^{-1} F(y_k, u), \quad \text{for} \quad k \geq 0. \tag{14}$$

It is well known that when the point $y_0(u)$, $u \in \omega_u$, is sufficiently close to a point $y_* = y_*(u)$ verifying $F(y_*, u) = 0$ with $F'_y(y_*, u)$ nonsingular, the sequence $\{y_k(u)\}_{k\geq 0}$ is well defined and converges to y_*. Here, the function Φ used in the proposition can be written

$$\Phi(y, u) = y - F'_y(y, u)^{-1} F(y, u).$$

The condition (11) is satisfied because, using $F(y_*, u) = 0$, we have

$$\Phi'_y(y_*, u) = I + F'_y(y_*, u)^{-1} F''_{yy}(y_*, u) F'_y(y_*, u)^{-1} F(y_*, u) - F'_y(y_*, u)^{-1} F'_y(y_*, u) = 0.$$

Consequently, the sequence $\{y'_k(u)\}_{k\geq 0}$ will converge to $y'_*(u)$.

This example enables us to distinguish between updated variables y_k on which f_k depends and those a_k that can be dismissed in the description of f_k. An implementation of Newton's method (14) uses, indeed, many other variables a_k in addition to the variables y_k updated by (14). If we had made no distinction between these variables, we would have had difficulty applying the proposition to the present case. On the one hand, it is often difficult, and in any case tedious, to give an updated rule $\tilde{\Phi}$ for some auxiliary variables a_k. On the other hand, there is generally no reason for the global law $(\Phi, \tilde{\Phi})$ to satisfy condition (11). To see this last point, just add to the process useless iterations like (3) acting on useless additional variables: the global law will not satisfy (11) at some point, although convergence of the derivatives f'_k to f' can be assured if f_k can be defined without using the additional variables.

4 Implementation Issues

At first glance, it may appear expensive to use automatic differentiation techniques for functions defined by iterative processes, because apparently the differentiation has to go through all the iterations of the original program to get the correct value of the derivatives. The automatic differentiation approach could look unattractive, especially in comparison to linearization techniques, which linearize equation (7) and obtain $y'_*(u)$ by solving the associated linear system. Because the calculation of $y'_*(u)$ is independent of the way the solution point y_* is obtained, this approach looks better.

Nevertheless, it is possible to avoid differentiating the complete program to obtain the desired approximation of y_*. The clue lies in the observation [5] that, as far as the calculation of the derivatives is concerned, the form of the function $u \mapsto y_0(u)$ has no effect on the convergence of $y'_k(u)$ to $y'_*(u)$ (although this form generally influences the *speed* of convergence). Proposition 1 makes indeed no other hypotheses on this function than its differentiability. As a result, the automatic differentiator can consider that y_0 is the function of u realized by the original program or any other convenient function—for instance, the constant function $u \mapsto y_0$. The first case is desirable if one has some reason to think that the function $u \mapsto y_0(u)$ realized by the original program is such that $y_0(u)$ is close to $y_*(u)$ and $y'_0(u)$ is close to $y'_*(u)$. The second case is interesting in many other situations, which we discuss below.

One way to avoid differentiating the complete program formed of all the iterations calculating approximations of y_* is to take as starting point, y_0, of the iterative process a

good approximation of the solution, y_*. This approximation can be obtained by a first run of the original program. The idea is therefore to generate a derivation code from a *modified* original program, in which the starting function is now the constant function $u \mapsto y_0 \simeq y_*$ and the number of iterations aims only at assuring the convergence of the derivatives in the generated program. The improvement in the approximations y'_k of the derivative y'_* can be observed in inequality (13), which shows that δ_k converges to δ_* linearly (the second term of the right hand side is approximately zero). This approach corresponds to solving the system coming from the linearization of (7) at $y_k \simeq y_*$ by the iterations

$$\delta_{k+1} := \Phi'_y(y_*, u) \cdot \delta_k + \Phi'_u(y_*, u) \cdot v, \tag{15}$$

where δ_k stands for $y'_k(u) \cdot v$.

This idea is extremely efficient when applied to methods for which $\Phi'_y(y_*, u) = 0$. In such cases, a good approximation of the derivatives is obtained in just one iteration (see, e.g., (15)). For example, for Newton's method (14), we would do just one iteration, starting from $y_0 \simeq y_*$ considered as a constant:

$$z = y_0 - F'_y(y_0, u)^{-1} F(y_0, u).$$

Since $F(y_0, u) \simeq F(y_*, u) = 0$, we see that the derivative of z can be written

$$z'(u) \cdot v \simeq -F'_y(y_0, u)^{-1} F'_u(y_0, u) \cdot v \simeq -F'_y(y_*, u)^{-1} F'_u(y_*, u) \cdot v,$$

which is precisely the expected derivative $y'_*(u) \cdot v$.

In the general case where $\Phi'_y(y_*, u) \neq 0$, the number of iterations needed to get convergence of the derivatives should be controlled during the differentiation phase, in the generated code. In the *direct mode* of differentiation, a direction $d \in \mathbb{R}^n$ is usually given, and the directional derivative $f'_k(x) \cdot d$ is evaluated by computing the directional derivatives $\tilde{z} = z'(x) \cdot d$ of all the variables z in the original program, in parallel with the calculation of z. Since $\tilde{y}_k = (y_k \circ u)'(x) \cdot d = y'_k(u) \cdot \tilde{u}$, a simple stopping criterion would be to test whether

$$\tilde{y}_k \simeq \tilde{y}_{k-1}.$$

This is less restrictive than requiring $y'_k(u) \simeq y'_{k-1}(u)$ but it is enough to get $\tilde{f}_k \simeq \tilde{f}$, because the nonsingularity of $I - \Phi'_y(y_k, u)$ implies then that

$$\tilde{y}_k \simeq \tilde{y}_* \equiv (y_* \circ u)'(x) \cdot d.$$

In the *reverse mode* of differentiation, a direction $r \in \mathbb{R}^m$ is given and the gradient $\nabla(r^\top f)$ with respect to x is calculated by updating the adjoint variable $\bar{z}$ associated with any variable z in the original code, in reverse order of their evaluation. The adjoint variable $\bar{z}$ is the current estimation of the partial derivative $\partial(r^\top f)/\partial z$. In this mode, a reasonable stopping criterion would consist of testing whether

$$\bar{y}_0 \simeq 0. \tag{16}$$

To see this, observe from (9), (10), and $y_k \simeq y_*$ that

$$\left(r^\top f_k\right)'(x) \simeq \left(r^\top f\right)'(x)$$

if and only if

$$\left(r^\top \psi\right)'_y (y_*, u) \cdot \left[y'_k(u) - y'_*(u)\right] \cdot u'(x) \simeq 0.$$

Differentiating (6) and (7) and using k times the resulting equations, we see that this is also equivalent to

$$\left(r^\top \psi\right)'_y (y_*, u) \cdot \Phi'_y(y_*, u)^k \cdot \left[y'_0(u) - y'_*(u)\right] \cdot u'(x) \simeq 0.$$

Since functions $u \mapsto y_0(u)$ and $u \mapsto y_*(u)$ are usually not linked and since $x \mapsto u(x)$ is independent of the iterative process, the only way to ensure this condition is to take k sufficiently large so as to have

$$\left(r^\top \psi\right)'_y (y_*, u) \cdot \Phi'_y(y_*, u)^k \simeq 0.$$

This is approximately equivalent to condition (16). More precisely, we have

$$\bar{y}_0 = \Phi'_y(y_0, u)^\top \Phi'_y(y_1, u)^\top \cdots \Phi'_y(y_{k-1}, u)^\top \nabla_y \left(r^\top \psi\right)(y_k, u).$$

In the reverse mode, this quantity is evaluated from right to left. This shows that, in this mode of differentiation, it is not easy to adapt the number of iterations to satisfy condition (16). However, since $y_0 \simeq y_1 \simeq \ldots \simeq y_k \simeq y_*$, the value of $\bar{y}_0$ can be approximated by

$$\bar{y}_0 \simeq \Phi'_y(y_{k-1}, u)^\top \Phi'_y(y_{k-2}, u)^\top \cdots \Phi'_y(y_0, u)^\top \nabla_y \left(r^\top \psi\right)(y_0, u),$$

which allows an easy adaptation of k during the reverse part of the calculation. With this approximation of $\bar{y}_0$, condition (16) is a reasonable criterion.

To conclude, let us mention another situation where it is better not to differentiate the function $u \mapsto y_0(u)$ realized by the original program. This situation arises when the iterative process obeys a rule like (5) only after a certain number of iterations, say s. Before this, some nondifferentiable actions such as step-sizing are taken. In this case, it is preferable that the automatic differentiator not generate codes for the calculation of $y_s(u)$; rather, it should consider that the iterative process starts with a constant initial approximation y_s.

Acknowledgments

The author has benefited from fruitful discussions with C. Lemaréchal and from enlighting comments by A. Griewank, which are at the root of Section 4.

References

[1] H. Fischer (1991). Automatic differentiation of the vector that solves a parametric linear system. *Journal of Computational and Applied Mathematics*, 35, 169–184. 4

[2] H. Fischer (1991). Special problems in automatic differentiation. Presented at the "1991 SIAM Workshop on Automatic Differentiation of Algorithms: Theory, Implementation and Application", January 6-8, 1991, Breckenridge, Colorado. 1, 2

[3] J. Ch. Gilbert, G. Le Vey, and J. Masse (1991). La différentiation automatique de fonctions représentées par des programmes. Rapport de Recherche nº 1557, INRIA, BP 105, F-78153 Le Chesnay, France. 1

[4] A. Griewank (1989). On automatic differentiation. In M. Iri and K. Tanabe (editors), *Mathematical Programming: Recent Developments and Applications*, pp. 83–108. Kluwer Academic Publishers, Dordrecht. 1

[5] A. Griewank (1991). Personal communication. 6

[6] A. Griewank and G. Corliss, eds. (1991). *Automatic Differentiation of Algorithms: Theory, Implementation, and Application*, Proceedings in Applied Mathematics 53. SIAM. Philadelphia. 1

[7] G. Kedem (1980). Automatic differentiation of computer programs. *ACM Transactions on Mathematical Software*, 6, 150–165. 1

[8] L. B. Rall (1981). *Automatic Differentiation, Techniques and Applications*. Lecture Notes in Computer Science 120. Springer-Verlag, Berlin. 1

[9] R. T. Rockafellar (1970). *Convex Analysis*. Princeton University Press, Princeton, New Jersey. 2

[10] L. Schwartz (1981). *Cours d'analyse*, Tome I. Hermann, Paris. 2