Dynamics of non-homogeneous systems. Proceedings of ISA RAS, 1997, pp. 193–210.

Evtushenko Yu.G.

Computing Centre of Russian Academy of Sciences
Moscow, Russia

FAST AUTOMATIC DIFFERENTIATION

Revised version 31 May 2003

1. Introduction

Many publications have been devoted to the technique of fast automatic differentiation (FAD), used for differentiating multivariate functions. We refer to the proceedings of the first SIAM Workshop on the Automatic Differentiation of Algorithms which was held in Brekenridge, Colorado, in 1991 (see [9]). An overview of the history and the state of the art of automatic differentiation and related techniques is given by Iri in [11]. In many cases, FAD is far superior to symbolic differentiation or to divided differences approximation.

Having studied literature on FAD, we realized that this approach was very similar to one we used for solving discrete optimal control problems with delay. We looked for more general expressions that would include the FAD formulas as a special case, but that could also be used for gradient calculations in systems that arise from discrete approximation of continuous systems governed by differential equations. Our preliminary results in this field were published in [1], [4] – [7]. This paper extends the FAD approach further to complex processes described by explicit and implicit expressions. We then apply the approach to optimization of a control system governed by a partial differential equation. The mathematical model of the control system is approximated by a corresponding optimal parameter selection problem which, in turn, can be viewed as a mathematical programming problem and hence can be solved by existing optimization algorithms. Using the generalized FAD expressions, we derive the exact gradient of the objective functional. Once this is accomplished, efficient gradient type algorithms for solving mathematical programming problems can be easily applied (see [6, 7]) to solving the parameter selection problem. These gradient algorithms usually require significantly fewer iterations and function evaluations than derivative-free methods which only use function values. The described approach enables us to find formulas for exact gradients in various complex problems and to use them in numerous minimization algorithms.

In Section 2 we present general expressions, from which we obtain the so-called "forward" and "reverse" differentiation expressions as a special case. We introduce the auxiliary function and use a canonical form which turns out to be very convenient for the representation of "reverse" expressions.

In Section 3 we consider the algebraic complexity of computing a function of several variables and its partial derivatives with respect to all the variables. If FAD is used, then the ratio of the computer time for calculating all partial derivatives of an elementary function to the time for calculating the underlying function value is bounded above by 3. In Section 4 the differentiation expressions are applied to the rounding error estimation.

In Section 5 we derive the expressions for the gradient of a function defined by the solution of an initial value problem for ordinary differential equation. We discretize the problem and, applying the expressions from Section 2, we obtain the gradient by using forward and reverse computations. Going from the discrete to continuous case, we obtain the gradient expressions well-known in the theory of ordinary differential equations. Similar results for optimal control problems are given in Section 6, where we describe a general approach to approximating the infinite-dimensional optimization problem by a finite-dimensional nonlinear program. We show that the reverse computation of the gradient corresponds to integrating the adjoint system of equations that appears in Pontryagin's maximum principle.

Our approach permits us to develop a new methodology for calculating the gradient in various systems described by partial differential equations. In Section 7 we derive the exact gradient based on the general expressions given in Section 2. We show how to apply this technique in the case of parabolic equation. Gradient for continuous parabolic systems was obtained in [20, 21, 22] based on calculus of variations. This classic approach has a drawback — it is not clear how the continuous state and adjoint equations should be discretized

The difficulties in integrating the initial and adjoint equations independently are described in numerous papers. For example, in [15] the authors write, "it is difficult to pass from the continuous to the discrete formulation, especially for nonlinear advection terms. This has led many researches to use methods working exclusively from the discrete equations of the model... It can be noticed that the problems to find the "good" gradient arise from some of the nonlinear terms of the equation". In paper [15] a special "matrix approach" was proposed to overcome these difficulties.

In our approach we start with a chosen discretization scheme for original state equation and derive the exact gradient expressions. Thus we automatically generate a unique discretization scheme for the adjoint equation. This technique appears to be very efficient, universal, and it could be used for numerous complex systems.

2. General expressions

There are many ways to derive the method of FAD. Among these the shortest and most general way is based on the well-known implicit function theorem. Suppose the mappings $\Phi : \mathbb{R}^n \times \mathbb{R}^r \to \mathbb{R}^n$ and $W : \mathbb{R}^n \times \mathbb{R}^r \to \mathbb{R}^1$ are differentiable. Let $z \in \mathbb{R}^n$ and $u \in \mathbb{R}^r$ satisfy the following nonlinear system of n scalar algebraic equations:

$$\Phi(z, u) = 0_n, \tag{1}$$

where 0_s is the s-dimensional null-vector.

We will use the following notations. For $f : \mathbb{R}^n \to \mathbb{R}^m$, let $f_x^\top$ denote the $(n \times m)$-matrix, which ij-th element is equal to $\partial f^j/\partial x^i$, and mean the first derivative of the vector-row $f^\top$ over the vector-column x. When $m = 1$, the transposition sign in the vector-column f_x is omitted.

We assume that the matrix $\Phi_z^\top(z, u)$ is nonsingular. According to the implicit function theorem, this system defines a continuous function $z = z(u)$ which is differentiable and whose derivative $dz^\top/du$, denoted by $N(u) \in \mathbb{R}^{n\times r}$, satisfies the following linear algebraic system:

$$\Phi_u^\top(z(u), u) + N(u)\Phi_z^\top(z(u), u) = 0_{rn}, \tag{2}$$

where $0_{\alpha\beta}$ is the $\alpha \times \beta$ rectangular null-matrix, N is a rectangular matrix of dimension $r \times n$:

$$N(u) = -\Phi_u^\top(z(u), u)[\Phi_z^\top(z(u), u)]^{-1}. \tag{3}$$

As a rule, z and u are referred to as *dependent and independent* variables, respectively. The composite function $\Omega(u) = W(z(u), u)$ is differentiable and the gradient with respect to the independent variable u (the reduced gradient) is equal to

$$\frac{d\Omega(u)}{du} = W_u(z(u), u) + N(u)W_z(z(u), u). \tag{4}$$

We introduce the Lagrange function $L(z,u,p) = W(z,u) + \Phi^\top(z,u)p$ with the Lagrange multiplier $p \in \mathbb{R}^n$, which is required to satisfy the linear algebraic system:

$$L_z(z,u,p) = W_z(z,u) + \Phi_z^\top(z,u)p = 0_n. \tag{5}$$

Then equation (4) can be rewritten in the form

$$\frac{d\Omega(u)}{du} = W_u(z(u),u) + \Phi_u^\top(z(u),u)p = L_u(z(u),u,p). \tag{6}$$

Expressions (4) and (6) are mathematically equivalent, but from the computational point of view there is a crucial difference. A slight variation in the way the function is differentiated will result in a drastic change in the efficiency of computation. In the first case we use the auxiliary matrix N; in the second case we use an additional Lagrange vector p. We shall show that expression (4) corresponds to the so-called “forward” (or “contravariant”, or “bottom-up”) differentiation, and formula (6) — to the “reverse” (or “covariant”, or “backward”, or “top-down”) differentiation.

As we will show below, (2) — (6) can be viewed as a basis for a number of expressions for calculating the gradient in various systems. In multi-step problems, the variables z and u are usually naturally partitioned into k variables of lower dimensionality:

$$z = [z_1, z_2, \ldots, z_k]^\top, \quad u = [u_1, u_2, \ldots, u_k]^\top, \qquad z_i \in \mathbb{R}^s, \quad u_i \in \mathbb{R}^m, \quad 1 \le i \le k.$$

Under this assumption, relation (1) is split into k relations as follows:

$$z_i = F(i, Z_i, U_i), \qquad 1 \le i \le k, \quad n = s \cdot k, \quad r = m \cdot k, \tag{7}$$

where Z_i and U_i are given sets of variables z_j and u_j, respectively, and the index i takes integer values from 1 to k. A more general case, when $i \in D \subset \{1, \ldots, k\}$, can be considered, but just for brevity, we suppose further that $D = \{1, \ldots, k\}$. For each $i \in D$, we introduce two sets of indices Q_i and K_i containing the indices of all variables z_i and u_i belonging to the sets Z_i and U_i, respectively. Then

$$Q_i = \{j \in D : z_j \in Z_i\}, \qquad K_i = \{j \in D : u_j \in U_i\}.$$

Let us introduce the *conjugate* index sets

$$\overline{Q}_i = \{j \in D : z_i \in Z_j\}, \qquad \overline{K}_i = \{j \in D : u_i \in U_j\}$$

and the corresponding vector sets

$$\overline{Z}_i = \{z_j : j \in \overline{Q}_i\}, \qquad \overline{U}_i = \{u_j : j \in \overline{K}_i\}.$$

The definition of these sets imply that if $z_j \in \overline{Z}_i$, $u_e \in \overline{U}_q$ (that is, if $j \in \overline{Q}_i$, $e \in \overline{K}_q$), then the following explicit functional dependencies are valid:

$$z_j = F(j, \ldots, z_i, \ldots), \qquad z_e = F(e, \ldots, u_q, \ldots).$$

Therefore, the sets Q_i and K_i may be called the *input index sets*, while $\overline{Q}_i$ and $\overline{K}_i$ are the *output index sets.*

Consider in (1) the mapping $\Phi(z,u)$ composed of the mappings $F(j,Z_j,U_j)-z_j$, where $j\in D$. Denote $N_{ij}=dz_j^\top/du_i\in\mathbb{R}^{s\times m}$. Then for process (7), we can rewrite (2) and (4) as follows:

$$N_{ij} = F_{u_i}^\top(j,Z_j,U_j)+\sum_{q\in Q_j} N_{iq}F_{z_q}^\top(j,Z_j,U_j), \tag{8}$$

$$\frac{d\Omega}{du_i} = W_{u_i}(z,u)+\sum_{j\in D} N_{ij}W_{z_j}(z,u). \tag{9}$$

With multiplier vectors $p_j\in\mathbb{R}^s$ we introduce the new auxiliary function

$$E(z,u,p)=W(z,u)+\sum_{j\in D}F^\top(j,Z_j,U_j)p_j.$$

It is very convenient to rewrite (7), (5) and (6) in the following *canonical* form:

$$z_i = E_{p_i}(z,u,p), \tag{10}$$

$$p_i = E_{z_i}(z,u,p)=W_{z_i}(z,u)+\sum_{q\in\bar{Q}_i}F_{z_i}^\top(q,Z_q,U_q)p_q, \tag{11}$$

$$\frac{d\Omega}{du_i} = E_{u_i}(z,u,p)=W_{u_i}(z,u)+\sum_{q\in\bar{K}_i}F_{u_i}^\top(q,Z_q,U_q)p_q. \tag{12}$$

We say that z_i is an output vector if the index set $\overline{Q}_i$ is empty. In this case,

$$p_i=W_{z_i}(z,u). \tag{13}$$

We say that the multistep process (7) is *explicit* if for every $i\in D$ the input set Q_i is such that for any element $j\in Q_i$ the inequality $j<i$ holds. According to (8) and (10), each matrix $N_{i\ell}$ and each vector z_i can be expressed for such processes by means of the previous matrices $N_{\ell j}$ and vectors z_j, respectively, where $1\le j<i$. In the last computational step we obtain z_k and calculate $p_k=W_{z_k}(z,u)$. Then we find from (11) all components p_i. Expression (12), as well as (9), yields all derivatives. We say that z_i and $N_{\ell i}$ are computed in *forward mode* because during their computation the index i increases from 1 to k. On the other hand, all vectors p_i are found in the reverse, or top-down, mode, which means that i decreases from $i=k$ to $i=1$. Explicit formulas are often used in discrete optimal control problems, where continuous differential equations are integrated using explicit numerical schemes.

Instead of E we can use the Lagrange function

$$L(z,u,p)=W(z,u)+\sum_{j\in D}[F^\top(j,Z_j,U_j)-z_j^\top]p_j.$$

The main expressions (10) – (12) can be rewritten as follows:

$$L_{p_i}(z,u,p)=0_s,\quad L_{z_i}(z,u,p)=0_s,\quad \frac{d\Omega}{du_i}=L_{u_i}(z,u,p).$$

The latter representation is traditionally employed for first-order necessary optimality conditions, where the statement $L_{u_i}(z,u,p)=0$ is added and all conditions acquire a symmetric form. Since the representation (10) – (12) is based essentially on a particular structure of the mapping Φ given by (7), this representation suits more for our purposes. Therefore, it will be used in sequel.

There is an interesting graph representation for the computational process of evaluating functions and their derivatives (see, for example, [11]). The process is visualized figuratively and graphically. However, applying this technique to implicit scheme is very difficult. The simplest example of such a problem is the process (28) given in Section 6. There the computational graph is cyclic and we have to use our unified approach based on expressions (10) – (12), where we did not postulate the explicitness of the computational process (7). If implicit integration expressions are used, then at each step i we have to solve the system of nonlinear equations (7) and to define the vector z_i. Next, from the linear algebraic systems (8) and (11), we define N_{ij} and p_i, respectively.

We should emphasize that there is an important difference between our expressions (8) – (12) and the well-known "forward" and "reverse" differentiation formulas. Our expressions reduce to the latter when process (7) is explicit. In the general case, our expressions could not be called "forward" or "reverse". The difference occurs when the equation (7) is implicit, i.e., $z_i \in Z_i$. In this case, the definition of p_i requires solving the system of linear equations, where the vector p_i appears on the left- and right-hand sides of (11). Similarly, the matrix N_{ij} appears on both sides in (8). Therefore, we cannot define N_{ij} and p_i sequentially. We have to solve all linear algebraic equations (8) or (11) simultaneously.

We consider simple examples that illustrate the characteristic properties of the two approaches presented for evaluating gradients. In many cases the reverse mode of computation has an advantage over the forward mode.

3. Differentiation of elementary functions

The expressions presented above cover a wide range of problems. In this section, we apply them to differentiating elementary functions.

The functions a^x $(a > 0)$, x^a, $\log_a x$ $(a > 0,\ a \neq 1)$, $\sin x$, $\cos x$, $\tan x$, $\cot x$, $\arcsin x$, $\arccos x$, $\arctan x$, $\mathrm{arccot} x$ are called *main elementary functions*. We assume that the codes for calculating the main elementary functions and their derivatives are stored in a computer and these calculations are carried out exactly (or with machine precision). We can add new functions to the list of the main elementary functions, as needed.

Assume that a real-valued function $f(u)$ is given explicitly. We say that a function $f(u)$ is *an elementary function* if it can be represented as a finite composition of main elementary functions and arithmetic operations.

Suppose that we have to calculate partial derivatives of a scalar-valued function $f(u)$, $u \in$ $\in \mathbb{R}^r$, with respect to all components u^i, $1 \leq i \leq r$. The function f is assumed to be a differentiable elementary function. Therefore, $f(u)$ can be defined by a sequential program. We introduce a new vector $z \in \mathbb{R}^k$ of intermediate variables. The evaluation of $f(u)$ is now carried out as a k-step computational process:

$$z^1 = F(1, Z_1, U_1), \quad z^2 = F(2, Z_2, U_2), \quad \ldots \quad z^k = F(k, Z_k, U_k), \tag{14}$$

where all $z^j \in \mathbb{R}^1$, $z^k = f(u)$; Z_i and U_i are sets of some components of the vectors z and u, respectively; $z^1 = Z_2$, Z_1 is empty. The sets Z_i consist of the already computed quantities z^j with $j < i$. In other words, f is the composition of basic operations whose derivatives are assumed to be computable for all arguments of interest.

We introduce the scalars $p^i \in \mathbb{R}^1$ and define the function

$$E = z^k + \sum_{i=1}^{k} F(i, Z_i, U_i) p^i.$$

In (10) – (12) we set $W = z^k$, therefore, (13) yields $p^k = 1$. Using (11) and (12), we find the gradient of $f(u)$ in reverse mode. In this way we obtain the following expressions for FAD:

$$p^i = \sum_{q \in \bar{Q}_i} F_{z^i}^{\top}(q, Z_q, U_q) p^q, \qquad \partial f(u)/\partial u^i = \sum_{q \in \bar{K}_i} F_{u^i}^{\top}(q, Z_q, U_q) p^q. \tag{15}$$

Many publications analyze various algorithms for automatic differentiation from the point of view of the algebraic complexity of the computation, i.e., the total number of arithmetic operations required to compute a function and its partial derivatives. For results in this field, we refer the interested readers to [2], [8]–[14] and the relevant references cited therein.

Let T_0 denote the total time required to calculate the value of the underlying function $f(u)$. Let T_g denote the additional time required for computing all partial derivatives $\partial f(u)/\partial u^i$, $1 \le i \le r$.

Theorem 3.1. *Suppose that*

1. *$f(u)$ is an elementary scalar-valued differentiable function of the vector $u \in \mathbb{R}^r$;*
2. *the time for computing the derivative of each main elementary function is less than twice the time required to evaluation the main elementary function itself;*
3. *the time required for memory processing and for execution of the assignment operator is negligible.*

If the formulas (15) *for fast automatic differentiation of $f(u)$ are used, then the gradient f_u is exact and the ratio $R = T_g/T_0$ is bounded above by 3.*

For comparison, recall that if we approximate the derivatives by divided differences, this ratio is $R = r$, and the gradient is not exact for any nonlinear scalar function f.

Proof. To prove of the theorem we use the approach developed in [9, 12, 13]. We assume that f is a composition of K main elementary functions whose values and gradients are assumed to be exactly computable.

Let T_i denote the time required to compute the i-th main elementary function and N_i be the number of computations of the i-th main elementary function value in process (14). The time required for one execution of addition, subtraction, multiplication and division is denoted by T_+, T_-, $T_\times$, $T_/$, respectively. Let N_+, N_-, $N_\times$, $N_/$ denote the number of additions, subtractions, multiplications, and divisions, respectively, among binary operations. We assume that $T_+ = T_- \le T_\times \le T_/ \le T_i$ for any i. The total time T_0 for computing the value of a function $f(u)$ is

$$T_0 = \sum_{i=1}^{K} T_i N_i + T_+ N_+ + T_- N_- + T_\times N_\times + T_/ N_/.$$

The number of computational steps in process (14) is equal to

$$k = \sum_{i=1}^{K} N_i + N_+ + N_- + N_\times + N_/.$$

Hence we obtain the following relation:

$$R \le \frac{T_g}{T_0} \le \frac{\left[\sum\limits_{i=1}^{K} N_i(2T_i + T_\times + T_+) + 2T_+N_+ + 2T_-N_- + 2N_\times(T_\times + T_+) + 2N_/(T_/ + T_+) - kT_+\right]}{T_0}.$$

The right-hand side is a nonlinear function of N_i, N_+, N_-, $N_\times$, $N_/$. To find the upper bound, we maximize this ratio with respect to these variables, i.e., we solve a linear fractional problem, whose solution is obvious. Taking into account $T_\times \le T_i$, we have the estimate

$$R \le \max\left[3, \max_{1\le i\le K}(2T_i + T_\times)/T_i\right] = 3,$$

which was to be proved. $\square$

This theorem gives us an upper estimate of the ratio R. Sometimes this ratio is smaller. Following [8], we consider the simplest product example. Let $f(u) = \prod_{i=1}^{r} u_i$. For this function we generate the computational process and define E-function as

$$z^1 = u^1, \quad z^i = z^{i-1}u^i, \quad 1 \le i \le r,$$
$$E = W + u^1p^1 + \sum_{i=2}^{r} z^{i-1}u^ip^i, \quad W = z^r.$$

Applying the above technique to the problem, we obtain

$$p^r = 1, \quad p^i = u^{i+1}p^{i+1}, \quad 1 \le i \le r-1,$$
$$\frac{\partial f}{\partial u^1} = p^1, \quad \frac{\partial f}{\partial u^j} = z^{j-1}p^j, \quad 2 \le j \le r.$$

It is obvious that in this case $R = 2$. We ignore the storage requirements. To compute the gradient far the example considered, we have to store an r-vector of intermediate variables.

4. Rounding error estimation

Suppose that at each step of process (7) we determine every state vector z_i with error ε_i. Thus, instead of (7) we use the following formula:

$$z_i = F(i, Z_i, U_i) + \varepsilon_i, \quad 1 \le i \le k. \tag{16}$$

The auxiliary function can be written as

$$E(z, u, \varepsilon) = W(z, u) + \sum_{j\in D}[F^\top(j, Z_j, U_j) + \varepsilon_j^\top]p_j.$$

Both vectors z_i and p_i have the same dimensionality. If process (16) is explicit, then the vector ε_i is the machine precision of an arithmetic computation of the vector $F(i, Z_i, U_i)$. In the implicit case, the error norms $\|\varepsilon_i\|$ tend to be much larger and are mainly determined by the accuracy of the solution to the nonlinear equations (16). Here Ω and p are composite functions of the control vector u and the error vector $\varepsilon^\top = [\varepsilon_1^\top, \varepsilon_2^\top, \ldots, \varepsilon_k^\top]$. Therefore, we can write $\Omega(u, \varepsilon)$, $p(u, \varepsilon)$. Let us use the canonical equations (10) – (12). We consider ε_i in these expressions as a component of the control vector u. We obtain exactly the same expression as (11). Using (12), we find that the gradient of Ω with respect to ε_i is given by

$$d\Omega(u, \varepsilon)/d\varepsilon_i = p_i(u, \varepsilon).$$

Suppose that the control vector u is given with an error and that, instead of u, we use $\bar u = u + \delta$. Then

$$\Omega(\bar u, \varepsilon) - \Omega(u, 0) = \sum_{i=1}^{k}\left[\langle p_i(u,0), \varepsilon_i\rangle + \left\langle \frac{d\Omega(u,0)}{du_i}, \delta_i\right\rangle\right] + O(\|\varepsilon\|^2 + \|\delta\|^2),$$

where the derivatives of Ω with respect to u_i are obtained from (12) and $\langle a, b\rangle$ denotes the inner product of vectors a and b. This estimate can be used instead of a laborious interval analysis. Theoretical and practical aspects of error estimation were investigated by Iri [12].

5. Derivatives with respect to initial conditions

The expressions for “forward” and “reverse” differentiation should not be considered as a stand-alone topic in mathematical analysis. In this section we present similar results which were obtained in the theory of ordinary differential equations. These results can be derived from expressions given in Section 2. We briefly illustrate how this can be done.

To compare the forward and reverse modes of differentiation, we consider the simplest problem in which we have to differentiate a function $W(z)$, $z \in \mathbb{R}^n$. Let the process be described by the following system of ordinary differential equations:

$$\frac{dz}{dt} = f(z), \qquad T_1 \le t \le T_2. \tag{17}$$

The solution of (17) is a function $z(t, z_1)$, with initial condition $z(T_1, z_1) = z_1$. Assume that for each z the right-hand side of (17) is a continuously differentiable function of z. We will find the derivative of the composite function $\Omega(z_1) = W(z(T_2, z_1))$. We introduce a matrix $N(t)$ with size $n \times n$ and a n-dimensional vector p as follows:

$$N(t) = \frac{\partial z^\top(t, z_1)}{\partial z_1}, \qquad p(t) = \frac{\partial W(z(T_2, z_1))}{\partial z(t, z_1)},$$

with the initial and terminal conditions

$$N(T_1) = I, \qquad p(T_2) = \frac{\partial W(z(T_2, z_1))}{\partial z(T_2, z_1)}, \tag{18}$$

where I is the identity matrix.

By applying the Euler numerical integration method, we obtain the following discrete approximation of continuous system (17):

$$z_1 = u, \quad z_i = z_{i-1} + hf(z_{i-1}), \quad 1 \le i \le k,$$

where $h = (T_2 - T_1)/(k - 1)$.

In discrete case $\Omega(z_1) = W(z_k)$. Denote $N_i = \partial z_i^\top / \partial u$, $1 \le i \le k$. Then, using (8), we obtain

$$N_1 = I, \quad N_i = N_{i-1} + hN_{i-1} f_z^\top(z_i), \quad 2 \le i \le k, \tag{19}$$

where $f_z^\top$ denotes the transposed Jacobian of the right-hand side of (17) with respect to z.

Introducing the auxiliary function E, using (11) and (13), we obtain the difference equations for vectors $p_i \in \mathbb{R}^n$:

$$\begin{gathered} E(z, u, p) = W(z_k) + p_1^\top u + \sum_{i=2}^{k} p_i^\top (z_{i-1} + hf(z_{i-1})), \\ p_i = p_{i+1} + hf_{z_i}^\top(z_i) p_{i+1}, \quad p_k = W_{z_k}(z_k), \qquad 1 \le i \le k - 1. \end{gathered} \tag{20}$$

The desired gradient is given by the two formulas:

$$\frac{d\Omega(z_1)}{dz_1} = N_k W_{z_k}(z_k), \qquad \frac{d\Omega(z_1)}{dz_1} = p_1.$$

Upon taking the limit as $h \to 0$ and $k \to \infty$, we find from (19) that the resulting continuous trajectory is described by the following matrix differential equation:

$$\frac{dN(t)}{dt} = N(t) \frac{\partial f^\top(z(t, z_1))}{\partial z(t, z_1)}. \tag{21}$$

Using (20), it easy to show that hi the limit as $h \to 0$ and $k \to \infty$, the adjoint (or costate) vector $p(t)$ satisfies the following vector differential equation:

$$\frac{dp(t)}{dt} = -f_z^\top(z(t, z_1))p(t). \tag{22}$$

The equations (21) and (22) appear in publications devoted to the theory of ordinary differential equations [3]. Expression (21) can be found by differentiating both sides of equation (17) with respect to z_1, and using the chain rule. For each given z_1 we compute the solutions $z(t, z_1)$ and $N(t)$ by integrating the differential equations (17) and (21) forward in time from $t = T_1$ to $t = T_2$. We solve the differential equation (22) with the terminal condition (18) backward in time from $t = T_2$ to $t = T_1$. The gradient of the composite function Ω can be found in two ways:

$$\frac{d\Omega(z_1)}{dz_1} = N(T_2)W_z(z(T_2, z_1)), \qquad \frac{d\Omega(z_1)}{dz_1} = p(T_1).$$

The last expression was given in [12]. Both expressions give exactly the same result, but in the first case, we have to integrate the matrix system (21), which consists of n^2 scalar differential equations. In the second case, we integrate vector system (22) which consists of only n scalar differential equations. Thus the second approach will be n times less costly than the first one. The formula (22) can be used for numerical solution of two-point boundary-value problems. The technique is based on the multiple back-and-forth shooting method which transforms a given boundary-value problem into a sequence of initial-value problems. The method involves forward integration of (17) and backward integration of (22).

We also mention the less common case where forward differentiation is less time consuming than backward differentiation. This case arises when we need to find the gradients of m functions $\Omega^i = W^i(z(T_2, z_1))$, where $1 \leq i \leq m$. We define $N(T_2)$ from (21), and all gradients are found as follows:

$$\frac{d\Omega^i(z_1)}{dz_1} = N(T_2)W_z^i(z(T_2, z_1)).$$

Therefore, if $m > n$, then the forward mode is more efficient than the reverse mode.

6. The optimal control problem

Optimal control theory has been formalized as a generalized extension of the calculus of variations. Optimal control theory has many successful applications in various disciplines ranging from mathematics and engineering to economics, social and management sciences. Many numerical methods for solving optimal control problems have been proposed, and the research continues, especially in the field of nonlinear problems.

The basic problem of optimal control can be described as follows. Let a process be governed by a system of ordinary differential equations:

$$\frac{dz}{dt} = f(t, z, u, \xi), \quad T_1 \leq t \leq T_2, \quad z(T_1, z_1) = z_1, \tag{23}$$

where the state vector $z \in \mathbb{R}^n$, the control u is an arbitrary piecewise continuous function of t having its values in U. The feasible set U is a given compact subset in the space $\mathbb{R}^r$. The vector of design parameters is $\xi \in V \subset \mathbb{R}^s$. As a rule, the scalars T_1, T_2 are fixed. If T_1, T_2, z_1 must be optimized, then we include them into vector ξ.

The problem is to find a control function $u(t) \in U$ and a vector of design parameters $\xi \in V$ that minimize the cost functional $W(z(T_2, z_1), \xi)$, subject to "mixed" constraints on state, control and vector of design parameters:

$$g(t, z(t), u(t), \xi) = 0, \qquad q(t, z(t), u(t), \xi) \leq 0, \qquad T_1 \leq t \leq T_2.$$

As a rule, this problem is reduced to a mathematical programming problem by using the control parametrization technique. Here we use only the simplest discretized version of (23), which is given by the Euler formula

$$z_i = z_{i-1} + h_{i-1} f(t_{i-1}, z_{i-1}, u_{i-1}, \xi) = F(t_{i-1}, z_{i-1}, u_{i-1}, \xi), \tag{24}$$

where

$$\sum_{i=2}^{k} h_{i-1} = T_2 - T_1, \qquad 0 < h_i, \quad t_1 = T_1, \quad t_k = T_2, \quad t_i = t_{i-1} + h_{i-1}, \quad 2 \leq i \leq k.$$

Thus we approximate the control by a piecewise constant function. We take into account the mixed constraints at each grid point:

$$g(t_i, z_i, u_i, \xi) = 0, \qquad q(t_i, z_i, u_i, \xi) \leq 0, \qquad 1 \leq i \leq k. \tag{25}$$

Now the objective function is $W(z_k, \xi)$ and the auxiliary function is expressed as

$$E(z, u, p, \xi) = W(z_k, \xi) + \sum_{i=2}^{k} F^{\top}(t_{i-1}, z_{i-1}, u_{i-1}, \xi) p_i.$$

Discretizing control and constraints, we arrive at the following parametrization-discretization scheme for approximate solution of optimal control problem.

Minimize $W(z_k, \xi)$ with respect to $u_i \in U$, $1 \leq i \leq k$, and $\xi \in V$, subject to mixed constraints (25).

In order to apply NLP-solvers to this problem we must obtain an efficient algorithm for computing the first order derivatives of the objective and constraints. Applying the results of Section 2, we find that all vectors $p_i \in \mathbb{R}^n$ and the derivatives of $\Omega(u, \xi) = W(z_k, \xi)$ can be calculated from

$$\begin{gathered} p_i = p_{i+1} + h f_{z_i}^{\top}(t_i, z_i, u_i, \xi) p_{i+1}, \qquad p_k = W_{z_k}(z_k, \xi), \qquad 1 \leq i \leq k-1, \\ \frac{d\Omega}{du_i} = h_i f_{u_i}^{\top}(t_i, z_i, u_i, \xi) p_{i+1}, \qquad 1 \leq i \leq k-1, \qquad \frac{d\Omega}{du_k} = 0, \\ \frac{d\Omega}{d\xi} = W_{\xi}(z_k, \xi + \sum_{i=2}^{k} F_{\xi}^{\top}(t_{i-1}, z_{i-1}, u_{i-1}\xi) p_i. \end{gathered} \tag{26}$$

The finite-dimensional approximate problems are solved by standard or adapted nonlinear programming methods (penalty function method, modified Lagrangian, gradient projection method, linearization, interior point techniques, etc.). The gradient methods have been successfully implemented and have been found to be more effective and robust than derivative-free methods. Second derivative methods are most attractive, but suffer from high dimensionality.

To solve the discretized problem by standard nonlinear programming methods we introduce additional functions (for example, penalty function, Lagrangian, modified Lagrangian and so on). According to (11) – (12), the expressions for computing their derivatives will be similar to (26) if instead of W we substitute these functions and take into account all nonzero derivatives W_{u_i}, W_{ξ} and W_{z_i}.

If we use a differentiable exterior penalty function with penalty coefficient τ, then the function E is defined as follows:

$$E = W(z_k, \xi) + \sum_{i=2}^{k} F^\top(t_{i-1}, z_{i-1}, u_{i-1}, \xi) p_i + \tau \sum_{i=1}^{k} \left[\|g(t_i, z_i, u_i, \xi)\|^2 \right] + \left[\|q_+(t_i, z_i, u_i, \xi)\|^2 \right],$$

where a_+ denotes the vector in $\mathbb{R}^\ell$ with components

$$(a_+)^i = \max[a^i, 0], \qquad 1 \le i \le \ell.$$

The expressions for computing gradients of the objective and constraints are derived via the adjoint system for the Euler and Runge–Kutta discretization schemes in [6]. These expressions and similar results for second derivatives were used for solving the optimal control problem with mixed constraints by gradient and Newton's methods. The implementation of expressions (26), a short description of an optimal control package, and some numerical illustrative examples are given in [6, 7].

Taking the limit as $h_i \to 0$ and $k \to \infty$, we find from (26) that the function $p(t)$ satisfies the following differential equation:

$$\dot{p} = -f_z^\top(t, z, u, \xi) p, \qquad p(T_2) = W_z(z(T_2, z_1), \xi). \tag{27}$$

This is the so-called costate (or adjoint, or conjugate) equation, which is used in the Pontryagin maximum principle [17]. The adjoint system is integrated backward in time, from the $t = T_2$ to $t = T_1$.

If system (23) is stiff, we have to use an implicit integration scheme [19]. For example, the application of the implicit Euler formula leads to

$$z_i = z_{i-1} + h_i f(t_i, z_i, u_i, \xi), \qquad 2 \le i \le k. \tag{28}$$

Although we have to solve a system of nonlinear equations at each step in this case, we can take much bigger step-size h_i in (28) than in (24). We obtain the following discrete costate equations from (11):

$$\begin{gathered} p_i = p_{i+1} + h_i f_{z_i}^\top(t_i, z_i, u_i, \xi) p_i, \qquad 2 \le i \le k-1, \\ p_k = W_{z_k}(z_k, \xi) + h_k f_{z_k}^\top(t_k, z_k, u_k, \xi) p_k. \end{gathered}$$

Hence all vectors p_i are found from implicit linear algebraic systems. The gradient expressions become

$$\frac{d\Omega}{du_i} = h_i f_{u_i}^\top(t_i, z_i, u_i, \xi) p_i, \qquad 2 \le i \le k, \qquad \frac{d\Omega}{du_1} = 0.$$

In the limit ($h_i \to 0$, $k \to \infty$) we obtain the same expressions as given in (27).

In some publications, the gradients are found from necessary optimality conditions by discretizing the initial and costate systems. In this case, some errors may arise. Indeed, if we simultaneously discretize the system of ordinary differential equations (23) and (27) using the Euler scheme, then we obtain (24) and

$$\begin{gathered} p_{i+1} = p_i - h_i f_{z_i}^\top(t_i, z_i, u_i, \xi) p_i, \qquad 1 \le i \le k-1, \\ \frac{d\Omega}{du_i} = h_i f_{u_i}^\top(t_i, z_i, u_i, \xi) p_i. \end{gathered}$$

These expressions do not coincide with (26) and, therefore, the gradient based on this approach is not correct. If h_i is rather small, then the difference between this expression and exact expression (26) is $O(h_i^2)$. But the error in the gradient calculation is not desirable when we use sensitive minimization algorithms (for example, conjugate gradients) or when the step size h_i is not small enough.

We emphasize this important conclusion: when we deal with optimal control problems, the discretization of the costate equation must correspond to the integration scheme of the initial system. It should not be performed independently. The same property is valid for more complicated processes described by partial differential equations. However, there the property is not as obvious as in the case of ordinary differential equations.

7. The optimal control problem of a parabolic system

The optimal control theory for distributed parameter systems has been extensively studied in [16, 20, 21, 22] and in many others research publications. In this and in the next sections we will show that our approach gives us a new methodology for finding the exact gradients in complex control systems governed by partial differential equations.

We discretize the infinite dimensional optimization problem to obtain an approximate finite dimensional nonlinear programming problem. We start with the second order parabolic heat equation

$$\frac{\partial z(x,t)}{\partial t} = a^2 \frac{\partial^2 z(x,t)}{\partial x^2} + u(x,t), \qquad 0 < x < \ell, \quad 0 < t < T, \tag{29}$$

where $z(x,t)$ is the temperature at time t at a point x and $u(x,t)$ is a distributed control.

The initial and boundary conditions are given by

$$z(x,0) = \varphi(x), \qquad 0 \le x \le \ell, \tag{30}$$

$$\frac{\partial z(0,t)}{\partial x} = 0, \qquad \frac{\partial z(\ell,t)}{\partial x} = \nu[g(t) - z(\ell,t)], \qquad 0 < t \le T, \tag{31}$$

where $g(t)$ is a boundary control. The problem is to find control functions $u(x,t)$ and $g(t)$ that minimize the cost functional

$$W = \int_0^\ell \Psi(z(s,T))ds, \tag{32}$$

where Ψ is continuously differentiable with respect to its argument.

If we solve the problem (29) with conditions (30) and (31), then we substitute the solution into (32) and evaluate W. This value is a composite function of the control functions $u(x,t)$ and $g(t)$. Therefore, we denote this dependence as $\Omega(u,g)$.

Since the optimal control cannot be obtained as an analytic solution of the necessary and sufficient optimality conditions, we attempt to find it numerically by minimizing W via a descent algorithm. We are thus faced with computing the gradient of the cost functional for which we apply the expressions given in Section 2. The problem is discretized by a finite difference approximation scheme. For the sake of simplicity we use uniform partition lengths and denote

$$\begin{gathered} x_i = i\Delta x, \quad t_j = j\Delta t, \quad i = 0,\dots,k, \quad j = 0,\dots,m, \\ \Delta x = \ell/k, \quad \Delta t = t/m, \quad z_i^j = z(i\Delta x, j\Delta t), \quad u_i^j = u(i\Delta x, j\Delta t), \\ \varphi = \varphi(i\Delta x), \quad g^j = g(j\Delta t), \quad i = 0,\dots,k, \quad j = 0,\dots,m. \end{gathered}$$

Let us discretize the parabolic equation (29) with an explicit forward Euler scheme in time, a centered scheme in space and use the simplest discretization in the vicinity of boundaries.

Then the cost functional (32), the differential equation (29) and the conditions (30), (31) are replaced by

$$\overline{W} = \Delta x \sum_{i=0}^{k} \alpha_i \Psi(z_i^m), \tag{33}$$

$$z_i^j = \begin{cases} (1-2\lambda)z_i^{j-1} + \lambda(z_{i-1}^{j-1} + z_{i+1}^{j-1}) + \Delta t u_i^{j-1}, & 1 \le i \le k-1, \quad 1 \le j \le m, \\ z_1^j, & i = 0, \quad 1 \le j \le m, \\ \mu z_{k-1}^j + \mu\nu\Delta x g^j, & i = k, \quad 1 \le j \le m, \\ \varphi_i, & 0 \le i \le k, \quad j = 0, \end{cases} \tag{34}$$

where $\lambda = a^2\Delta t/(\Delta x)^2$, α_i — quadrature coefficients, $\mu = 1/(1+\nu\Delta x)$.

We introduce the adjoint variables p_i^j and the auxiliary function

$$E = \overline{W} + \sum_{i=1}^{k-1}\sum_{j=1}^{m}[(1-2\lambda)z_i^{j-1} + \lambda(z_{i-1}^{j-1} + z_{i+1}^{j-1}) + \Delta t u_i^{j-1}]p_i^j + \\ + \sum_{j=1}^{m}[z_1^j p_0^j + \mu(z_{k-1}^j + \nu\Delta x g^j)p_k^j] + \sum_{i=0}^{k}\varphi_i p_i^0.$$

Applying formula (11) we obtain

$$p_i^j = \begin{cases} (1-2\lambda)p_i^{j+1} + \lambda(p_{i-1}^{j+1} + p_{i+1}^{j+1}), & 2 \le i \le k-2, \quad 0 \le j \le m-1, \\ (1-2\lambda)p_1^{j+1} + p_0^j + \lambda p_2^{j+1} & i = 1, \quad 1 \le j \le m-1, \\ (1-2\lambda)p_{k-1}^{j+1} + \mu p_k^j + \lambda p_{k-2}^{j+1} & i = k-1, \quad 1 \le j \le m-1, \\ \alpha_i \Delta x \Psi_{z_i^m} + p_0^m \delta_i^1 + \mu p_k^m \delta_i^{k-1}, & 0 \le i \le k, \quad j = m, \\ (1-2\lambda)p_i^1 + \lambda p_{i+1}^1 & i = 1, \quad i = k-1, \quad j = 0, \\ \lambda p_1^{j+1} & i = 0, \quad 0 \le j \le m-1, \\ \lambda p_{k-1}^{j+1} & i = k, \quad 0 \le j \le m-1, \end{cases}$$

where $\delta_\alpha^\beta = 1$ if $\alpha = \beta$, else $\delta_\alpha^\beta = 0$.

Using (12) we find the derivatives

$$\begin{aligned} &\frac{d\Omega}{du_i^j} = \Delta t p_i^{j+1}, && 1 \le i \le k-1, && 0 \le j \le m-1, \\ &\frac{d\Omega}{du_0^j} = \frac{d\Omega}{du_k^j} = 0, && && 0 \le j \le m-1, \\ &\frac{d\Omega}{du_i^m} = 0, && 0 \le i \le k, && \\ &\frac{d\Omega}{dg^j} = \mu\nu\Delta x p_k^j, && && 1 \le j \le m. \end{aligned} \tag{35}$$

According to [18], the discrete approximation (34) is stable only if $\lambda \le 1/2$. We can replace it with the more complicated implicit scheme which is stable for all λ:

$$z_i^{j+1} = z_i^j + \lambda[z_{i-1}^{j+1} - 2z_i^{j+1} + z_{i+1}^{j+1}] + \Delta t u_i^{j+1}.$$

In the interior region, the vector p is now defined by the difference equation

$$p_i^j = p_i^{j+1} + \lambda[p_{i+1}^j - 2p_i^j + p_{i-1}^j].$$

If we let $k \to \infty$, $\Delta t \to 0$, $\Delta x \to 0$, then in both cases we find that the function $p(x,t)$ satisfies the following conditions:

$$\begin{aligned} &\frac{\partial p(x,t)}{\partial t} + a^2 \frac{\partial^2 p(x,t)}{\partial x^2} = 0, && 0 < x < \ell, \quad 0 < t < T, \\ &p(x,t) = \Psi_z(z(x,T)), && 0 \le x \le \ell, \\ &\frac{\partial p(0,t)}{\partial x} = 0, \quad \frac{\partial p(\ell,t)}{\partial x} + \nu p(\ell t) = 0, && 0 < t < T. \end{aligned} \tag{36}$$

Discrete approximations of all these conditions were obtained on the basis of (11) and, therefore, (35) gives us the exact gradients for the discretize process (34). Equation (36) is conjugate (or adjoint) to (29).

The gradients of the cost functional for the continuous problem are given by

$$\frac{d\Omega}{du(x,t)} = p(x,t), \qquad \frac{d\Omega}{dg(t)} = \nu a^2 p(\ell,t). \tag{37}$$

Using the expressions widely used in classical calculus of variations, we can rewrite (37) in the following standard form:

$$\delta\Omega = \int_0^T \int_0^\ell p(x,t)\delta u(x,t)dxdt + \nu a^2 \int_0^T p(\ell t)\delta g(t)dt.$$

The meaning of notation (37) becomes clear. Similar results were obtained in [22] by calculus of variations.

References

1. *Aida-Zade K.R. and Evtushenko Y.G.*, (1989) Fast automatic differentiation, *Mathematical Modelling*, 1, 121–139 (in Russian).

2. *Baur W. and Strassen V.*, (1983) The complexity of partial derivatives, *Theoretical Computer Sciences*, 22, 317–320.

3. *Caratheodory C.*, (1956) Variationsrechnung und partielle Differentialgleichungen erster Ordnung, Band 1, Leipzig.

4. *Evtushenko Y.G. and Mazouric V.P.*, (1989) Optimization Software, Znanie, Moscow (in Russian).

5. *Evtushenko Y.G.*, (1991) Automatic differentiation viewed from optimal control theory, in [9], 25–30.

6. *Evtushenko Y.G.*, (1985) Numerical Optimization Techniques, Optimization Software, Inc., Publications Division, New York.

7. *Grachev N.I. and Evtushenko Y.G.*, (1980) A library of programs for solving optimal control problems, U.S.S.R. Comput. Maths. Math. Phys. (Pergamon Press), 19, 99–119.

8. *Griewank A.*, (1989) On automatic differentiation, in: M. Iri and K. Tanabe (Eds.) *Mathematical Programming: Recent Developments and Applications*, Kluwer Academic Publishers, 83–108.

9. *Griewank A. and Corliss G.F., Eds.* (1991) *Automatic Differentiation of Algorithms Theory, Implementation and Application*, SIAM, Philadelphia.

10. *Griewank A.*, (1992) Achieving logarithmic growth of temporal and spatial complexity in reverse automatic differentiation, *Optimization Methods and Software*, 1, 35–54.

11. *Iri M.*, (1991) History of Automatic differentiation and rounding error estimation. In: [9], 3–16.

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

13. *Iri M. and Kubota K.*, (1987) Methods of fast automatic differentiation and applications, Research memorandum RMI 87-02. Department of Mathematical Engineering and Instrumental Physics, Faculty of Engineering, University of Tokyo.

14. *Kim K., Nesterov Y., Skokov V., Cherkasskij B.*, (1984) Efficient algorithm for differentiation and extremal problem, *Economy and Mathematical Methods*, 20, 309–318 (in Russian).

15. *Lellouche J.M., Devenon J.L., Dekeyser I.*, (1994) Boundary control of Burger's equation. — A numerical approach, *Computers and Mathematics with Applications*, 28 (5), 33–44.

16. *Marchuk G.I.*, (1992) *Conjugate equations and analysis of complex systems*, Science, Moscow (in Russian).

17. *Pontryagin L.S., Boltyansky V.G„ Gamkrelidze R.V., Mitshenko E.F.*, (1961) *Mathematical Theory of Optimal Process*, Nauka, Moscow.

18. *Samarskii A.A.*, (1971) *Introduction to the Theory of Finite-difference Methods*, Nauka, Moscow (in Russian).

19. *Stetter H.J.*, (1973) *Analysis of Discrimination Methods for Ordinary Differential Equations*, Springer-Verlag, Berlin, Heidelberg, New York.

20. *Teo K.L. and Wu Z.S.*, (1984) Computation Methods for Optimizing Distributed Systems, Academic Press, Inc.

21. *Teo K.L., Goh C.J., Wong K.H.*, (1991) A Unified Computation Approach to Optimal Control Problems, Longman Scientific & Technical, England.

22. *Vasiliev F.P.*, (1981) Numerical Methods for Solving Optimization Problems, Nauka, Moscow (in Russian).