3rd IFIP WG-7.6 Working Conference on
Optimization-Based
Computer-Aided Modelling and Design
Prague, Czech Republic, May 24–26, 1994
PROCEEDINGS, pp. 3–12
J. Doležal, J. Fidler (Eds.)
Published by the Institute of Information Theory
and Automation Academy of Sciences of the Czech
Republic, Prague 1995

Fast Automatic Differentiation and Optimal Control Theory[1]

(Revised version 28 December 2003)

Yuri G. Evtushenko
Computing Center
40 Vavilov Str.
117967 Moscow GSP-1, Russia
e-mail: evtushenko@ccas.ru

1. GENERAL FORMULAS

Many publications have been devoted to the technique of fast automatic differentiation (FAD), which is used for differentiating a function of many variables. 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 [1]). An overview of the history and the state of the art of automatic differentiation and related techniques is given by M. Iri in [6]. In many cases, FAD is far superior to symbolic differentiation or to divided difference approximation. When studying the papers about FAD, we felt that this approach is very close to one which had been used for the discrete optimal control problem with delay. We decided to find general formulas that would permit us to get as a particular case the FAD formulas and the formulas for the evaluation of a gradient in systems described by a discrete approximation of a continuous system governed by differential equations. Some preliminary results in this field were published in [2]–[5]. Here we develop this approach and apply it to a system described by a partial differential equation.

There are many ways to reach the method of FAD among which the shortest and most general way is based on the well-known implicit function theorem. Suppose that for vectors $z \in \in \mathbb{R}^n$ and $u \in \mathbb{R}^r$ the differentiable functions $W(z,u)$, $\Phi(z,u)$ define mappings $W : \mathbb{R}^n \times \mathbb{R}^r \to \to \mathbb{R}^1$, $\Phi : \mathbb{R}^n \times \mathbb{R}^r \to \mathbb{R}^n$. Vectors z and u 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 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 derivatives satisfy the following linear algebraic system:

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

[1]Research supported by the grant N° 93-012-450 from Russian Scientific fund

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

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

The composite function $\Omega(u) = W(z(u), u)$ is differentiable and

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

We introduce the Lagrange function $L(z, u) = W(z, u) + \Phi^\top(z, u)p$ with the Lagrange multiplier $p \in \mathbb{R}^n$. This vector is found by solving the following linear system:

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

The formula (4) for the derivative of the composite function $\Omega(u)$ with respect to u can be rewritten in the form

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

The formulas (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 the formula (4) corresponds to the so-called "forward" (or "contravariant", or "bottom-up") differentiation, the formula (6) — to the "reverse" (or "covariant", or "backward", or "top-down") differentiation.

Usually in multistep problems, the vectors z and u are naturally partitioned into vectors of lower dimensionally:

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

Under this assumption the relation (1) is split into k vector 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, U_i are given sets of vectors z_j, u_j, respectively, and the index i takes integer values from 1 to k. More generally we shall write $i \in D$. The number of elements of set D is denoted $|D|$ and is equal to k. For each $i \in D$ we introduce two sets of indices Q_i and K_i, containing the indices of all vectors 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

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

and the corresponding vector sets

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

It follows from the definition of these sets that if $z_q \in \bar{Z}_i$, $u_e \in \bar{U}_s$ (that is, if $q \in \bar{Q}_i$, $e \in \bar{K}_s$), then the following functional dependencies are valid:

$$z_q = F(q, \ldots, z_i, \ldots), \qquad z_e = F(e, \ldots, u_s, \ldots).$$

Therefore, the sets Q_i and K_i may be called the *input index sets*, while $\bar{Q}_i$ and $\bar{K}_i$ are the *output index sets*. The vector function $\Phi(z,u)$ in (1) can be represented as the union of vector functions $F(i, Z_i, U_i) - z_i$, where $i \in D$. We define the matrix $N_{ij} = \dfrac{\partial z_j^\top}{\partial u_i}$ of dimension $m \times s$.

For the process (7) we can rewrite formulas (2) and (4) as follows:

$$
\begin{aligned}
N_{ij} &= F_{u_i}^\top(j, Z_j, U_j) + \sum_{q \in Q_j} N_{iq} F_{z_q}^\top(j, Z_j, U_j), && (8)\\
\frac{d\Omega}{du_i} &= W_{u_i}(z,u) + \sum_{j \in D} N_{ij} W_{z_j}(z,u). && (9)
\end{aligned}
$$

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

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

and rewrite (7), (5) and (6) in the *canonical* form:

$$
\begin{aligned}
z_i &= H_{p_i}(z,u,p), && (10)\\
p_i &= H_{z_i}(z,u,p) = W_{z_i}(z,u) + \sum_{q \in \bar{Q}_i} F_{z_i}^\top(q, Z_q, U_q) p_q, && (11)\\
\frac{d\Omega}{du_i} &= H_{u_i}(z,u,p) = W_{u_i}(z,u) + \sum_{q \in \bar{K}_i} F_{u_i}^\top(q, Z_q, U_q) p_q. && (12)
\end{aligned}
$$

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

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

If Q_j and K_j are empty, then z_j is an input state vector j and $N_{ij} = 0_{sm}$ for all $i \in D$.

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 (11) each matrix N_{si} and each vector z_i can be expressed by means of the previous matrices N_{sj} and vectors z_j, respectively, where $1 \le j \le i$. In the last computational step we obtain z_k and calculate $p_k = W_{z_k}(z,u)$. After that, we find from (11) all components p_i. Formulas (9) and (12) give all derivatives. We say that z_i and N_{si} are found 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 $i = 1$. All matrices N_{si} vanish if $i \ge s$. Explicit formulas are often used in discrete optimal control problems, where continuous differential equations are integrated by using explicit numerical schemes.

If implicit integration formulas are used, then at each step i we have to solve the system of nonlinear equations (7) and define the vector z_i. Next, from the linear algebraic systems (8) and (11), we define N_{ij} and p_i, respectively. We consider simple examples that illustrate the characteristic properties of the two approaches presented for evaluating gradients. In some cases the reverse mode of computation has an advantage over the forward mode, in other cases not.

Assume that the functions Φ and W are twice differentiable. In this case the composite function $\Omega(u)$ is also twice differentiable and

$$
\frac{d^2\Omega}{du^2} = L_{uu} + N L_{zu} + L_{uz} N^\top + N L_{zz} N^\top. \tag{14}
$$

Introduce matrices M and R of dimensions $n \times r$ and $n \times n$, respectively. These matrices are found from the following linear systems:

$$\Phi_z^\top M + L_{zu} = 0_{nr}, \qquad \Phi_z^\top R \Phi_z = L_{zz}.$$

Instead of using (14) we can compute the second derivative as follows:

$$\frac{d^2\Omega}{du^2} = L_{uu} + \Phi_u^\top M + M^\top \Phi_u + \Phi_u^\top R \Phi_u.$$

This formula was used in [5] for solving the optimal control problem with state-vector constraints by Newton's method.

2. DIFFERENTIATION OF ELEMENTARY FUNCTIONS

The following functions: a^x $(a > 0)$, x^α, $\log_a x$ $(a > 0,\ a \neq 1)$, $\sin x$, $\cos x$, $\operatorname{tg} x$, $\operatorname{ctg} x$, $\arcsin x$, $\arccos x$, $\operatorname{arctg} x$, $\operatorname{arcctg} x$ are called *main elementary functions*. We suppose that the codes for calculation of main elementary functions and their derivatives are stored in a computer and these calculations are carried out exactly (or with machine precision).

We say that a function $f(x)$ 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 the partial derivatives of a scalar-valued function $f(u)$, $u \in \mathbb{R}^r$, with respect to all variables u^i. 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:

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

where $z^j \in \mathbb{R}^1$, $z^k = f(u)$, $z^1 = Z_2$, Z_1 is empty and each $F(i, Z_i, U_i)$ is a unary or binary basic operation. In the first case F is a main elementary function of a single argument. In the second case F is an arithmetic operation. In both cases the partial derivatives of F with respect to all arguments are known.

We define the vector $p \in \mathbb{R}^k$ and introduce the Hamiltonian function

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

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

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

There are many papers analyzing various algorithms for automatic differentiation from the point of view of computational complexity. We refer to [1, 7, 8, 9]. Let T_0 denote the total time for calculating the value of the underlying function $f(u)$. Let T_g denote the additional time for computing all partial derivatives $\dfrac{\partial f(u)}{\partial u^i}$, $1 \leq i \leq r$, required after evaluation of $f(u)$.

Theorem 1. *Suppose that*

1) *$f(u)$ is an elementary scalar-valued differentiable function of vector $u \in \mathbb{R}^r$,*

2) *the time for computing the derivative of each main elementary function is less than twice the time needed for the evaluation of the main elementary function itself,*

3) *the time required for memory processing and for execution of the assignment operator is negligible.*

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

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

3. ROUNDING ERROR ESTIMATION

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

$$z_i = F(i, Z_i, U_i) + \varepsilon_i, \qquad 1 \le i \le k. \tag{15}$$

The Hamiltonian can be written as

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

In the case of explicit formulas, the components ε_i are the machine precision of an arithmetic computation of the value $F(i, Z_i, U_i)$. 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 (15). Now Ω and p become the 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 canonical equation (10) – (12). We consider ε_i in these formulas as a component of the control vector u. We obtain exactly the same formula as (11). Therefore, from (12) one obtains,

$$\frac{d\Omega(u, \varepsilon)}{d\varepsilon_i} = p_i(u, \varepsilon).$$

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

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

where the derivatives of Ω with respect to u_i are obtained from (12). This estimation can be used instead of a laborious interval analysis. Theoretical and practical aspects of such an approach were developed by M. Iri [6].

4. DERIVATIVES WITH RESPECT TO INITIAL CONDITIONS

To compare the forward and reverse modes of differentiation, we start from the simplest problem in which we have to differentiate a function $W(z)$, $z \in \mathbb{R}^n$, with respect to the initial condition z_1.

Let the process be described by the following system of ordinary differential equations:

$$\frac{dz}{dt} = f(z), \qquad t_1 \leq t \leq t_2. \tag{16}$$

The solution of (16) is $z(t, z_1)$, with initial state $z(t_1, z_1) = z_1$. We will find the derivative of $W(z(t_2, z_1))$ with respect to z_1. We introduce a matrix $N(t)$ with size $n \times n$ and an 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 $p(\cdot) \in \mathbb{R}^n$, and the boundary conditions

$$N(t_1) = I_n, \qquad p(t_2) = \frac{\partial W(z(t_2, z_1))}{\partial z(t_2, z_1)},$$

where I_s is the $s \times s$ identity matrix.

We differentiate both sides of equation (16) with respect to z_1. Using the chain rule, we get the matrix differential equation

$$\frac{dN(t)}{dt} = N(t) \frac{\partial f^\top(z(t, z_1))}{\partial z(t, z_1)}. \tag{17}$$

It easy to show that $p(t)$ satisfies the following vector differential equation:

$$\frac{dp(t)}{dt} = -f_z^\top(z(t, z_1)) p(t). \tag{18}$$

The gradient of the function W can be found in two ways:

$$\frac{dW}{dz_1} = N(t_2) \frac{\partial W(z(t_2, z_1))}{\partial z(t_2, z_1)}, \qquad \frac{dW}{dz_1} = p(t_1).$$

Both formulas give exactly the same result, but in the first case we have to integrate the matrix system (17), which consists of n^2 scalar differential equations, while in the second case we integrate only the vector equation (18) which consists of n scalar differential equations. Thus the second approach will be n times less costly than the first approach. The last formula was given in [9].

We also mention the case where forward differentiation is less time consuming than backward differentiation. This case arises when we must find gradients of m functions $W_i(z(t_2, z_1))$, where $m > n$. We define $N(t_2)$ from (17), and all gradients are found as follows:

$$\frac{\partial W_i(z(t_2, z_1))}{\partial z_1} = N(t_2) \frac{\partial W_i(z(t_2, z_1))}{\partial z(t_2, z_1)}.$$

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

5. OPTIMAL CONTROL PROBLEM

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

$$\frac{dz}{dt} = f(t, z, u), \qquad t_1 \le t \le t_2, \quad z(t_1, z_1) = z_1, \tag{19}$$

where $u(\cdot) \in \mathbb{R}^s$, $z(\cdot) \in \mathbb{R}^r$. The problem is to find a control u that minimizes the cost functional $W(z(t_2, z_1))$. We use the simplest discrete approximation of (19) which is given by the Euler formula

$$z_i = z_{i-1} + hf(t_{i-1}, z_{i-1}, u_{i-1}) = F(i, z_{i-1}, u_{i-1})$$

for $i = 2, \dots, k$, and z_1 is given.

The objective function is $W(z_k)$ and the Hamiltonian is

$$H = W(z_k) + \sum_{i=2}^{k} \langle F(i, z_{i-1}, u_{i-1}), p_i \rangle.$$

All vectors $p_i \in \mathbb{R}^s$ and the derivatives of W are found from

$$p_k = W_z(z_k), \quad p_i = p_{i+1} + hf_z^\top(t_i, z_i, u_i)p_{i+1}, \quad \frac{dW}{du_i} = hf_u^\top(t_i, z_i, u_i)p_{i+1}, \quad 1 \le i \le k-1. \tag{20}$$

If $h \to 0$ and $k \to \infty$, then we obtain from (20)

$$\dot{p} = -f_z^\top(t, z, u)p, \qquad p(t_2) = W_z(z(t_2, z_1)).$$

This is the so-called adjoint differential equation, which is used in the Pontryagin maximum principle.

6. OPTIMAL CONTROL PROBLEM OF A PARABOLIC SYSTEM

In this section we turn our attention to a control system governed by a partial differential equation. For the sake of simplicity we consider the second order parabolic heat equation

$$\frac{\partial z}{\partial t} = a^2 \frac{\partial^2 z}{\partial x^2} + u(x, t), \tag{21}$$

where $z(x, t)$ is the temperature at moment t in a point x; $u(x, t)$ is a distributed control.

The initial and boundary conditions are given by

$$\begin{gathered} z(x, 0) = \varphi(x), \qquad 0 \le x \le \ell, \\ \frac{\partial z(0, t)}{\partial x} = 0, \quad \frac{\partial z(\ell, t)}{\partial x} = \nu[g(t) - z(\ell, t)], \qquad 0 \le t \le T, \end{gathered} \tag{22}$$

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

$$W(u, g) = \int_0^t \Psi(z(s, T))ds. \tag{23}$$

The problem is discretized by a finite difference approximation scheme. Denote

$$\begin{gathered} w = \{(x_i, t_j) : x_i = i\Delta x, \ t_j = j\Delta t, \ i = 0, \dots, k, \ j = 0, \dots, m\}, \\ \Delta x = \ell/k, \ \Delta t = t/m, \ z_{i,j} = z(i\Delta x, j\Delta t), \ u_{i,j} = u(i\Delta x, j \ t), \\ \varphi_i = \varphi(i\Delta x), \ g_j = g(j\Delta t), \ i = 0, \dots, k, \ j = 0, \dots, m. \end{gathered}$$

The cost function (23), the differential equation (21) and the conditions (22) are replaced by

$$W = \Delta x \sum_{i=0}^{k} \alpha_i \Psi(z_{i,m}),$$

$$z_{i,j} = \begin{cases} (1-2\lambda)z_{i,j-1} + \lambda(z_{i-1,j-1} + z_{i+1,j-1}) + \Delta t u_{i,j-1}, & 1 \le i \le k-1, \quad 1 \le j \le m, \\ z_{1,j}, & i = 0, \quad 1 \le j \le m, \\ \mu z_{k-1,j} + \mu\nu\Delta x g_j, & i = k, \quad 1 \le j \le m, \\ \varphi_i, & 0 \le i \le k, \quad j = 0, \end{cases}$$

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

Introduce adjoint variables $p_{i,j}$ and the Hamiltonian function

$$\begin{aligned} H &= \sum_{i=1}^{k-1}\sum_{j=1}^{m}[(1-2\lambda)z_{i,j-1} + \lambda(z_{i-1,j-1} + z_{i+1,j-1}) + \Delta t u_{i,j-1}]p_{i,j} + \\ &+ \sum_{j=1}^{m} z_{1,j}p_{0,j} + \mu\sum_{j=1}^{m}(z_{k-1,j} + \nu\Delta x g_j)p_{k,j} + \sum_{i=0}^{k}\varphi_i p_{i,0} + \Delta x \sum_{i=0}^{k}\alpha_i\Psi(z_{i,m}). \end{aligned}$$

Applying formula (11) we obtain

$$p_{i,j} = \begin{cases} (1-2\lambda)p_{i,j+1} + \lambda(p_{i-1,j+1} + p_{i+1,j+1}), & 2 \le i \le k-2, \quad 0 \le j \le m-1, \\ (1-2\lambda)p_{1,j+1} + p_{0,j} + \lambda p_{2,j+1}, & i = 1, \quad 0 \le j \le m-1, \\ \lambda p_{1,j+1}, & i = 0, \quad 0 \le j \le m-1, \\ (1-2\lambda)p_{k-1,j+1} + \mu p_{k,j} + \lambda p_{k-2,j+1}, & i = k-1, \quad 0 \le j \le m-1, \\ \lambda p_{k-1,j+1}, & i = k, \quad 0 \le j \le m-1, \\ \alpha_i\Delta x\Psi_{z_{i,m}}, & i \in [0:k]/\{1,k-1\}, \quad j = m, \\ \alpha_i\Delta x\Psi_{z_{i,m}} + p_{0,m}, & i = 1, \quad j = m, \\ \alpha_i\Delta x\Psi_{z_{k-1,m}} + \mu p_{k,m}, & i = k-1, \quad j = m. \end{cases}$$

Using (12) we find the formulas for gradients

$$\begin{aligned} \frac{dW}{du_{i,j}} &= \Delta t p_{i,j+1}, \qquad 1 \le i \le k-1, \quad 0 \le j \le m-1, \\ \frac{dW}{dg_j} &= \mu\nu\Delta x p_{k,j}, \qquad 1 \le j \le m. \end{aligned}$$

If we let $k \to \infty$ $\Delta t \to 0$, $\Delta x \to 0$, then we obtain that the function $p(x,t)$ satisfies the following conditions

$$\begin{aligned} \frac{\partial p}{\partial t} &= -a^2\frac{\partial^2 p}{\partial x^2}, \\ \frac{\partial p(0,t)}{\partial x} &= 0, \quad \frac{\partial p(\ell,t)}{\partial x} = \nu p(\ell,t), \;\; p(x,T) = \Psi_z(z(x,T)), \;\; 0 \le x \le 1. \end{aligned} \tag{24}$$

Equation (24) is conjugate to (21).

The approach which we described here enables us to find quick formulas for exact gradients in various complicated problems and to use them in numerous minimization algorithms. These gradient algorithms usually require significantly fewer iterations and function evaluations than the methods which use only the evaluation of the function values.

References

[1] *Automatic differentiation of algorithms. Theory, implementation and application.* Edited by A. Griewank, G.F. Corliss, SIAM, Philadelphia (1991).

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

[3] Y.G. Evtushenko, V.P. Mazouric. *Optimization software*. Znanie, Moscow (1989) (in Russian).

[4] Y. Evtushenko. *Automatic differentiation viewed from optimal control theory.* In [1], pp. 25–30.

[5] Y.G. Evtushenko. *Numerical Optimization techniques. Optimization software.* Inc. Publications Division. New York (1985).

[6] M. Iri. *History of Automatic differentiation and rounding error estimation.* In [1], pp. 3–16.

[7] W. Baur, V. Strassen. The complexity of partial direvatives. *Theoretical Computer Sciences*, 22 (1983), pp. 317–320,

[8] K. Kim, Y. Nesterov, V. Skokov, B. Cherkasskij. Efficient algorithm for differentiation and extremal problem. *Economy and mathematical methods*, 20 (1984), pp. 309–318 (in Russian).

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