Dynamics of non-homogeneous systems. Proceedings of ISA RAS, 1999, v. 2, pp. 53–62

Birgin E.G.[1]

Applied Mathematics Department
IMECC-UNICAMP, CP 6065, CEP 13081-970
Campinas — SP — Brazil

Evtushenko Yu.G.[2]

Computing Centre of Russian Academy of Sciences
Moscow, Russia

AUTOMATIC DIFFERENTIATION FOR OPTIMAL CONTROL PROBLEMS

Revised version 27 May 2003

1. Introduction

Much attention has been paid to the development of numerical methods for solving optimal control problems. The most popular approach in this field turned to be the reduction of the original problem to a nonlinear programming problem (NLP) (see, for example, [16, 2, 18, 19]). In [3] it was shown that the computation of the gradient in this particular case is closely related with fast automatic differentiation (FAD) techniques (see [13, 14, 11, 15, 12]). In [4], using generalized FAD expressions, the exact gradient of the objective functional was derived in a very simple canonical form. The aims of this paper is to show the application of these canonical formulas to optimal control processes being integrated by Runge–Kutta family of numerical methods.

In Section 2 of this paper we present the canonical formulas and in Section 3 we apply them to the discrete version of the optimal control problem. Some final remarks are presented in Section 4.

2. Canonical Formulas

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

$$\frac{dx(t)}{dt} = f(x(t), u(t), \xi), \qquad T_0 \le t \le T_f, \tag{1}$$

where the state function x has its values in $\mathbb{R}^{n_x}$, the control u is an arbitrary piecewise continuous function of t having its values in a given compact set $U \subset \mathbb{R}^{n_u}$ and the vector of design parameters $\xi \in V \subset \mathbb{R}^{n_\xi}$. The solution of (1) is a function $x(t)$ with initial condition $x(T_0) = x_0$. In general, the scalars T_0, T_f and vector x_0 are fixed. If T_0, T_f or x_0 must be optimized then we can include them into vector of design parameters ξ.

[1] Work sponsored by FAPESP (Grant N° 95-2452-6).

[2] Work sponsored by Russian Foundation for Basic Research (Grants N° 98-01-00517 and N° 96-15-96124) and FAPESP (Grant N° 96-6631-5).

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

$$W(T_0, T_f, x(T_f), u(T_f), \xi) \tag{2}$$

subject to mixed constraints on state, control and vector of design parameters

$$h(x(t), u(t), \xi) = 0, \qquad q(x(t), u(t), \xi) \le 0, \qquad T_0 \le t \le T_f. \tag{3}$$

As a rule, this problem is reduced to a mathematical programming problem using a discretization scheme. Control function $u(t)$ is approximated by a piecewise constant function in which the accuracy of discretization depends on the problem to be solved. Sometimes it must be rather high and the software should permit us to provide it. Having some experience in solving practical problems, we came to the conclusion that very often the accuracy of integration must be higher than accuracy of founded optimal control. Therefore, for the sake of simplicity it is possible to assume that control vector u is constant at each interval of integration. Discretizing system (1) we obtain a N step process in which functions x and u are naturally represented as vectors

$$x^\top = [x_0^\top, x_1^\top, \ldots, x_N^\top], \qquad u^\top = [u_0^\top, u_1^\top, \ldots, u_N^\top],$$

where $x_i = x(t_i) \in \mathbb{R}^{n_x}$, $u_i = u(t_i) \in \mathbb{R}^{n-U}$ and $t_i = T_0 + \sum\limits_{k=0}^{i-1} h_k$ for $0 \le i \le N$, $0 < h_i \in R$ are the discretization steps satisfying

$$\sum_{i=0}^{N-1} h_i = (T_f - T_0), \tag{4}$$

$x \in \mathbb{R}^{n_x \times (N+1)}$, $u \in \mathbb{R}^{n_u \times (N+1)}$, and $v^\top$ means the transposition of vector v. The discrete version of (1) is split into the N relations

$$x_i = F(X_i, U_i, \xi), \qquad 1 \le i \le N, \tag{5}$$

where X_i and U_i are given sets of variables x_j and u_j, respectively, and the index j takes values from 0 to N. At initial step we have $X_0 = U_0 = \emptyset$ and for simplicity we write $x_0 = F(X_0, U_0, \xi)$. T he mixed constraints (3) are considered at each grid point

$$h(x_i, u_i, \xi) = 0, \qquad q(x_i, u_i, \xi) \le 0, \qquad 0 \le i \le N \tag{6}$$

and the discretized optimal control problem for an approximated solution of the original problem is to minimize

$$W(T_0, T_f, x_N, u_N, \xi) \tag{7}$$

with respect to control vector u and vector of design parameters ξ and subject to $u_i \in U$ for $0 \le i \le N$, $\xi \in V$ and constraints (6).

From (5), fixing control vector u and vector of design parameters ξ, we obtain the state vector $x_N(u, \xi)$ and substitute it in the expression $W(T_0, T_f, x_N, u_N, \xi)$. Then we define the composite function $\Omega(u, \xi) = W(T_0, T_f, x_N(u, \xi), u_N, \xi)$. For numerical minimization of this function it is important to know the total derivatives of Ω with respect to u and ξ as it allows us to use efficient gradient type minimization algorithms. In [4] it was shown that, for multistep process (5), formulas to compute total derivatives $\dfrac{d\Omega}{du}$ and $\dfrac{d\Omega}{d\xi}$ can be obtain as follow.

For each set X_i and U_i we introduce the sets of indices Q_i and K_i containing the indices of all variables x_j and u_j belonging to the sets X_i and U_i, respectively. Then

$$Q_i = \{j : x_j \in X_i\}, \qquad K_i = \{j : u_j \in U_i\}$$

and we define its *conjugate* indices sets

$$\overline{Q}_i = \{j : x_i \in X_j\}, \qquad \overline{K}_i = \{j : u_i \in U_j\}.$$

The sets Q_i and K_i are the *input* indices sets at i-th step, while $\overline{Q}_i$ and $\overline{K}_i$ are the *output* indices sets at i-th step. Let us define adjoints vectors $p_i \in \mathbb{R}^{n_x}$ for $0 \le i \le N$, total adjoint vector $p^\top = [p_0^\top, p_1^\top, \ldots, p_N^\top] \in \mathbb{R}^{n_x \times (N+1)}$ and introduce the new auxiliary function

$$E(x, u, \xi, p) = W(T_0, T_f, x_N, u_N, \xi) + \sum_{i=o}^{N} F(X_i, U_i, \xi)^\top p_i. \tag{8}$$

Finally, formulas for the computation of adjoint vectors p_i and total derivatives $\dfrac{d\Omega}{du}$ and $\dfrac{d\Omega}{d\xi}$ can be written in the following *canonical* form:

$$\begin{aligned} x_i &= E_{p_i}(x, u, \xi, p), & (9)\\ p_i &= E_{x_i}(x, u, \xi, p) = W_{x_i}(T_0, T_f, x_N, u_N, \xi) + \sum_{j \in \overline{Q}_i} F_{x_i}(X_j, U_j, \xi)^\top p_j, & (10) \end{aligned}$$

$$\frac{d\Omega(u,\xi)}{du_i} = E_{u_i}(x, u, \xi, p) = W_{u_i}(T_0, T_f, x_N, u_N, \xi) + \sum_{j \in \overline{K}_i} F_{u_i}(X_j, U_j, \xi)^\top p_j \tag{11}$$

and

$$\frac{d\Omega(u,\xi)}{d\xi} = E_{\xi}(x, u, \xi, p) = W_{\xi}(T_0, T_f, x_N, u_N, \xi) + \sum_{j=0} F_{\xi}(X_j, U_j, \xi)^\top p_j, \tag{12}$$

where i varying from 0 to N and H_y denotes, from now on, the partial derivative of function H with respect to y, i.e., $H_y = \dfrac{\partial H}{\partial y}$ whereas $\dfrac{dH}{dy}$ denotes the total derivative of H with respect to y. We assume that relation (9) defines an explicit process, i.e., at each i-th step the input set Q_i is such that for any $k \in Q_i$ the inequality $k < i$ holds. In this case, from (10), we have

$$p_N = W_{x_N}(T_0, T_f, x_N, u_N, \xi). \tag{13}$$

Note that, considering expression (10), we can conclude that adjoints values p_i are partial derivatives of Ω with respect to state variables x_i.

3. Application to Runge–Kutta Methods

As a practical application we consider multistep process (5) given by Runge–Kutta family methods

$$x_{i+1}^0 = F(X_{i+1}, U_{i+1}, \xi) = x_i^0 + h_i \sum_{j=0}^{\rho-1} [\alpha_j f(z_i^j)], \qquad i = 0, \ldots, N-1 \tag{14}$$

and

$$x_i^{j+1} = x_i^0 + \beta_j h_i f(z_i^j), \qquad j = 0, \ldots, \rho-2, \quad i = 0, \ldots, N-1, \tag{15}$$

where α_j are defined for $j = 0, \ldots, \rho - 1$ and β_j for $j = 0, \ldots, \rho - 2$, $x_i^0 \equiv x_i$, and $t_i^0 \equiv t_i$ for $i = 0, \ldots, N$, auxiliary vectors $x_i^j \in \mathbb{R}^{n_z}$, $t_i^j = t_i + \beta_{j-1} h_i$, $u_i^j = u(t_i^j)$ and $z_i^j = (x_i^j, u_i^j, \xi)$ for $i = 0, \ldots, N$ and $j = 0, \ldots, \rho - 1$. As we are assuming that control variables u_i are constant into the integration step we have $u_i = u_i^j$ and $z_i^j = (x_i^j, u_i^j, \xi)$ for $i = 0, \ldots, N$ and $j = 0, \ldots, \rho - 1$. Table shows some examples of Runge–Kutta family methods (see, for example, [17]).

Examples of Runge–Kutta family methods

ρ	Integration Method	Values
1	Runge–Kutta order 1 or Euler	$\alpha_0 = 1$
2	Runge–Kutta order 2 or Modified Euler	$\alpha_0 = 0$, $\alpha_1 = 1$ $\beta_0 = 0.5$
2	Runge–Kutta order 2	$\alpha_0 = 0.5$, $\alpha_1 = 0.5$ $\beta_0 = 1$
4	Runge–Kutta order 4	$\alpha_0 = \frac{1}{6}$, $\alpha_1 = \frac{1}{3}$, $\alpha_2 = \frac{1}{3}$, $\alpha_3 = \frac{1}{6}$ $\beta_0 = 0.5$, $\beta_1 = 0.5$, $\beta_2 = 1$

Applying canonical formulas (9) – (12) to Runge–Kutta definition (14) – (15) we obtain

$$E(x, u, \xi, p) = W(T_0, T_f, z_N^0) + \sum_{i=0}^{N-1} \sum_{j=0}^{\rho-2} [x_i^{j+1}]^\top p_i^{j+1} + \sum_{i=0}^{N-1} [x_{i+1}^0]^\top p_{i+1}^0 \tag{16}$$

and replacing, in (16), x_{i+1}^0 and x_i^{j+1} for its values in (14) and (15) we arrive to

$$E(x, u, \xi, p) = W(T_0, T_f, z_N^0) + \sum_{i=0}^{N-1} \sum_{j=0}^{\rho-2} [x_i^0 + \beta_j h_i f(z_i^j)]^\top p_i^{j+1} + \sum_{i=0}^{N-1} [x_i^0 + h_i \sum_{j=0}^{\rho-1} [\alpha_j f(z_i^j)]^\top p_{i+1}^0. \tag{17}$$

Now, differentiating (17) we obtain

$$p_i^0 = \frac{dE(x, u, \xi, p)}{dx_i^0} = W_{x_i^0}(T_0, T_f, z_N^0) + p_i^1 + [\beta_0 h_i f_{x_i^0}(z_i^0)]^\top p_i^1 + p_{i+1}^0 + \left[h_i [\alpha_0 f_{x_i^0}(z_i^0)]\right]^\top p_{i+1}^0, \tag{18}$$

$$p_i^j = \frac{dE(x, u, \xi, p)}{dx_i^j} = W_{x_i^j}(T_0, T_f, z_N^0) + [\beta_j h_i f_{x_i^0}(z_i^j)]^\top p_i^{j+1} + h_i [\alpha_j f_{x_i^j}(z_i^j)]^\top p_{i+1}^0, \tag{19}$$

$$\frac{dE(x, u, \xi, p)}{du_i} = W_{u_i}(T_0, T_f, z_N^0) + \sum_{j=0}^{\rho-2} [\beta_j h_i f_{u_i}(z_i^j)]^\top p_i^{j+1} + h_i \sum_{j=0}^{\rho-1} [\alpha_j f_{u_i}(z_i^j)]^\top p_{i+1}^0, \tag{20}$$

$$\frac{dE(x, u, \xi, p)}{du_N} = W_{u_N}(T_0, T_f, z_N^0) \tag{21}$$

and

$$\frac{dE(x, u, \xi, p)}{d\xi} = W_\xi(T_0, T_f, z_N^0) + \sum_{i=0}^{N-1} \sum_{j=0}^{\rho-2} [\beta_j h_i f_\xi(z_i^j)]^\top p_i^{j+1} + \sum_{i=0}^{N-1} \left[h_i \sum_{j=0}^{\rho-1} [\alpha_j f_\xi(z_i^j)]\right]^\top p_{i+1}^0, \tag{22}$$

where $i = 0, \ldots, N-1$ and $j = 1, \ldots, \rho - 1$. Finally, rearranging formulas (18) – (22) and discarding null derivatives, we arrive to the subroutine for the computation of $\Omega(u, \xi)$, $\frac{d\Omega(u, \xi)}{du}$

and $\dfrac{d\Omega(u,\xi)}{d\xi}$. It is necessary to remark that the approach presented above (formulas (18) – (22)) is simpler and closer to computer implementation than analogous results given in [2] (pages 379–382). Meanwhile our formulas can be used for wide class of Runge–Kutta methods. This improvement comes from the application of auxiliary function (8) and canonical formulas (9) – (12) introduced in [4].

Subroutine 3.1

Set $x_0^0 \leftarrow x_0$.
For $i = 0, \ldots, N-1$ (increasing loop)
{
For $j = 0, \ldots, \rho - 2$ (increasing loop defined only for $\rho \geq 2$)
{
Set $y \leftarrow f(z_i^j)$,
compute $x_i^{j+1} = x_i^0 + h_i \beta_j y$ and
compute $x_{i+1}^0 = x_{i+1}^0 + h_i \alpha_j y$.
}
Set $y \leftarrow f(z_i^{\rho-1})$ and
compute $x_{i+1}^0 = x_{i+1}^0 + h_i \alpha_{\rho-1} y$.
}
Compute $W(T_0, T_f, z_N^0)$.
Set $\dfrac{dE(x,u,\xi,p)}{d\xi} \leftarrow W_\xi(T_0, T_f, z_N^0)$,
set $\dfrac{dE(x,u,\xi,p)}{du_N} \leftarrow W_{u_N}(T_0, T_f, z_N^0)$ and
set $p_N \leftarrow W_{x_N}(T_0, T_f, z_N^0)$.
For $i = N-1, \ldots, 0$ (decreasing loop)
{
Compute $\dfrac{dE(x,u,\xi,p)}{d\xi} = \dfrac{dE(x,u,\xi,p)}{d\xi} + h_i \alpha_{\rho-1} [f_\xi(z_i^{\rho-1})]^\top p_{i+1}$,
compute $\dfrac{dE(x,u,\xi,p)}{du_i} = h_i \alpha_{\rho-1} [f_{u_i}(z_i^{\rho-1})]^\top p_{i+1}$,
compute $v = h_i \alpha_{\rho-1} [f_{x_i^{\rho-1}}(z_i^{\rho-1})]^\top p_{i+1}$ and
compute $p_i = p_{i+1} + v$.
For $j = \rho - 2, \ldots, 0$ (decreasing loop defined only for $\rho \geq 2$)
{
Compute $y = \alpha_j p_{i+1} + \beta_j v$,
compute $\dfrac{dE(x,u,\xi,p)}{d\xi} = \dfrac{dE(x,u,\xi,p)}{d\xi} + h_i \alpha_j [f_\xi(z_i^j)]^\top y$,
compute $\dfrac{dE(x,u,\xi,p)}{du_i} = \dfrac{dE(x,u,\xi,p)}{du_i} + h_i \alpha_j [f_{u_i}(z_i^j)]^\top y$,
compute $v = h_i [f_{x_i^j}(z_i^j)]^\top y$ and
compute $p_i = p_i + v$.
}
}

In subroutine above, $y, v \in \mathbb{R}^{n_z}$ are auxiliary vectors for intermediate computations. It is important to remark that there is not need to save adjoints values p_i^j of intermediate variables x_i^j with $j \neq 0$. For this reason we use notation p_i for adjoints values p_i^0 of x_i^0. This kind of implementation is a mixed strategy which try to find an equilibrium between computational cost and memory storage. It is easy to see that, as W and its partial derivatives W_ξ, W_{u_N} and W_{x_N} are being computed together, we should call to a unique function which compute all of them using reverse mode. This is not the case of function f and its derivatives which are being computed at different times. For this reason it is not possible to take advantage of common expressions between f and f_{x_i}, f_{u_i} and f_ξ. We call this strategy of *hybrid* FAD. In this implementation we are sacrificing computational time to save memory storage.

A particular and important class of control problems are such in which the goal is to minimize the duration of the process. A possible strategy to handle this situation is to introduce a new design parameter ξ^{n_ξ}, define goal function (7) as

$$W(T_0, T_f, z_N^0) = (T_f - T_0)\xi^{n_\xi}, \tag{23}$$

i.e, goal function is the duration of controlled process, and rewrite (1) as

$$\frac{dx(t)}{dt} = \bar{f}(x, u, \xi) = \xi^{n_\xi} f(x, u, \xi), \qquad T_0 \leq t \leq T_f, \tag{24}$$

$$0 \leq \xi^{n_\xi} \leq +\infty. \tag{25}$$

If we apply subroutine 2.1 to compute the gradient of this particular case it will compute f two times at the same point. First one to compute $\bar{f}(x, u, \xi) = \xi^{n_\xi} f(x, u, \xi)$ and the second one to compute $\bar{f}_{\xi^{n_\xi}}(x, u, \xi) = f(x, u, \xi)$. This problem comes from the way in which we apply canonical formulas (9) – (12). To solve it we should use, instead of auxiliary vector $y \in \mathbb{R}^{n_x}$, a three dimensional array $y \in \mathbb{R}^{n_x \times N \times \rho}$ (or $N \times \rho$ vectors $y \in \mathbb{R}^{n_x}$). In this way $f(z_i^j)$ values will be saved in y_i^j as $(N+1) \times \rho$ vectors $x_i^j \in \mathbb{R}^{n_x}$ are being saved) to be used in the computation of $\bar{f}_{\xi^{n_\xi}}(z_i^j)$. This modification comes from the observation of subroutine 2.1 or from the application of canonical formulas to the following reformulation of Runge–Kutta family methods:

$$x_{i+1}^0 = F(X_{i+1}, U_{i+1}, \xi) = x_i^0 + h_i \sum_{j=0}^{\rho-1} [\alpha_j \xi^{n_\xi} f_i j], \qquad i = 0, \ldots, N-1,$$

$$x_i^{j+1} = x_i^0 + \beta_j h_i \xi^{n_\xi} f_i^j, \qquad j = 0, \ldots, \rho-2, \quad i = 0, \ldots, N-1$$

and

$$f_i^j = f(z_i^j), \qquad j = 0, \ldots, \rho-1, \quad i = 0, \ldots, N-1.$$

This is an special modification for a particular case. In this new approach we are doubling the storage space in order to avoid the computational cost of evaluate f twice.

4. Conclusions

In this work we show how to apply the methodology introduced in [4] to Runge–Kutta family of integration methods. An equivalent approach can be applied to other integration methods like, for example, Newton–Cotes and Adams–Moulton (see [17] and its numerous references [1, 5, 6, 7, 20]). Initial optimal control problem (1) – (3) is approximated by discrete process (14) – (15) with constraints (6) and goal function (7). This discrete problem can be solved by many nonlinear programming methods like augmented Lagrangian, linearization, Newton's methods, interior point techniques, etc. There are many ways to take into account

constraints (6). If we use sequential minimization techniques (as penalty function methods) then part of these constraints, as for example box constraints, can be considered explicitly in the optimization process while other constraints can be penalized. In all cases, rewriting auxiliary function (8), canonical formulas (9) – (12) and subroutine 3.1 are applicable for the computation of total derivatives. Moreover, in the same way, derivatives of higher order can be computed.

References

1. *Butcher J.C.* On Runge–Kutta processes of High Order // Journal of Australian Mathematics Society. 1964. Vol. 4. P. 179–194.

2. *Evtushenko Y.G.* Numerical Optimization Techniques // Optimization Software Inc. Publications Division. New York, 1985.

3. *Evtushenko Y.G.* Automatic Differentiation Viewed from Optimal Control Theory // Automatic Differentiation of Algorithms. Theory, Implementation and Application. Griewank A. and Corliss G.F. (eds.). Philadelphia, SIAM, 1991. P. 25–30.

4. *Evtushenko Y.G.* Fast Automatic Differentiation // Dynamics of Non-Homogeneous Systems — Proceedings of ISA. Popkov Y. (ed.). Moscow: Institute for System Analysis, 1997. P. 193–210 (in Russian).

5. *Fehlberg E.* New High-Order Runge–Kutta Formulas with Stepsize Control for Systems of First and Second-Order Differential Equations // Angew Z. Math. Mech. 1964. Vol. 44. P. T17–T29.

6. *Fehlberg E.* New High-Order Runge–Kutta Formulas with an Arbitrary Small Truncation Error // Angew Z. Math. Mech. 1964. Vol. 46. P. 1–16.

7. *Fehlberg E.* Klassische Runge–Kutta Formeln fünfter und siebenter Ordnung mit Schrittweitenkontrolle // Computing. 1964. Vol. 4. P. 93–106.

8. *Goldstein A.A.* Convex Programming in Hilbert Space // Bulletin of the American Mathematical Society. 1964. Vol. 70. P. 709–710.

9. *Grachev N.I. and Evtushenko Y.G.* A Library of Programs for Solving Optimal Control Problems // USSR Computational Mathematics and Mathematical Physics. 1980. Vol. 19. P. 99–119.

10. *Grachev N.I. and Filkov A.N.* Solution of Optimal Control Problems in System DIOS. Moscow: Computing Centre of Russian Academy of Sciences, 1986 (in Russian).

11. *Griewank A.* On Automatic Differentiation // Mathematical Programming: Recent Developments and Applications. Iri M. and Tanabe K. (eds.). Kluwer Academic Publishers, 1989. P. 83–108.

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

13. *Iri M.* Simultaneous Computation of Functions, Partial Derivatives and Estimates of Rounding Errors — Complexity and Practicality // Japan Journal of Applied Mathematics. 1984. Vol. 1. P. 223–252.

14. *Iri M. and Kubota K.* Methods of Fast Automatic Differentiation and Applications // Research memorandum RMI. University of Tokyo, Faculty of Engineering, Department of Mathematical Engineering and Instrumental Physics, 1987. P. 87–102.

15. *Iri M.* History of Automatic Differentiation and Rounding Error Estimation // Automatic Differentiation of Algorithms. Theory, Implementation and Application. Griewank A. and Corliss G.F (eds.). Philadelphia: SIAM, 1991. P. 3–16.

16. *Polak E.* Computation Methods in Optimization. New York and London: Academic Press, 1971.

17. *Stoer J. and Bulirsch R.* Introduction to Numerical Analysis // Berlin, Heidelberg and New York: Springer-Verlag Inc., 1972.

18. *Teo K.L. and Wu Z.S.* Computation Methods for Optimizing Distributed Systems. Orlando: Academic Press, 1984.

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

20. *Shanks E.B.* Solution of Differential Equations by Evaluation of Functions // Mathematical Computation. 1966. Vol. 20. P. 21–38.