
Using AD-generated Derivatives in Optimal Control of an Industrial Robot

Roland Griesse¹ and Andrea Walther²

¹ Chair of Mathematics in Engineering, University of Bayreuth, Germany
`roland.griesse@uni-bayreuth.de`

² Institute of Scientific Computing, Technical University Dresden, Germany
`awalther@math.tu-dresden.de`

Summary. This article reviews the use of Automatic (or Algorithmic) Differentiation (AD) in nonlinear programming problems arising from the discretization of constrained optimal control problems with ordinary differential equations. Depending on the number and type of constraints, the forward or reverse mode of AD are favored. As an example, we consider a fast turn-around manoeuvre of an industrial robot.

Key words: optimal control, multibody system, automatic differentiation

1 Introduction

Automatic or Algorithmic Differentiation (AD) is a valuable tool in generating first and higher order derivatives of mathematical formulae given by computer code segments. It is known to be more efficient than symbolic differentiation, more accurate than numerical differentiation, and more convenient and reliable than hand-coded derivatives. We particularly advocate the use of AD when the functions to be differentiated are of high complexity, as verified by the right hand side of the multibody ODE optimal control problem under consideration. We only briefly recall the problem, which has been investigated with a different emphasis in [GW02]:

An industrial robot—as depicted in Figure 1—is to perform a fast turn-around manoeuvre. We denote by $q = (q_1, q_2, q_3)$ the angular coordinates of the robot's joints, q_1 referring to the angle between the base and the two-arm system. The robot is controlled via three control functions u_1 through u_3 , denoting the respective angular momentum applied to the joints (from bottom to top) by electrical motors. The goal is to minimize the energy-related objective

$$J(q, u) = \int_0^{t_f} [u_1(t)^2 + u_2(t)^2 + u_3(t)^2] dt \quad (1)$$

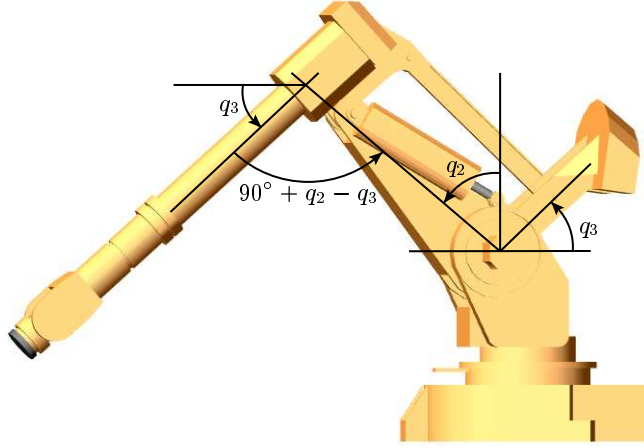


Fig. 1. Industrial robot ABB IRB 6400 and its coordinates

where the final time t_f is given. The robot's dynamics obey a system of three differential equations of second order:

$$M(q)\ddot{q} = v(q, \dot{q}) + w(q) + \tau_{\text{friction}}(\dot{q}) + \tau_{\text{reset}}(q) + u \quad (2)$$

where $M(q)$ is a 3×3 symmetric positive definite matrix containing moments of inertia, called a generalized mass matrix, see also [GW02]. The complete equations of motion can be found in Knauer [Kna01] and Heim [Hei98].

The robot's task to perform a turn-around manoeuver is expressed in terms of initial and terminal conditions:

$$\begin{aligned} q_1(0) &= \pi/2 & \dot{q}_1(0) &= 0 & q_1(t_f) &= -\pi/2 & \dot{q}_1(t_f) &= 0 \\ q_2(0) &= 0 & \dot{q}_2(0) &= 0 & q_2(t_f) &= 0 & \dot{q}_2(t_f) &= 0 \\ q_3(0) &= 0 & \dot{q}_3(0) &= 0 & q_3(t_f) &= 0 & \dot{q}_3(t_f) &= 0. \end{aligned} \quad (3)$$

The choice of control functions is restricted by the presence of control constraints (after scaling to the interval $[-1, 1]$)

$$|u_i(t)| \leq 1 \quad \text{for all } t \in [0, t_f] \text{ and } i = 1, 2, 3 \quad (4)$$

as well as state constraints

$$\begin{aligned} -100^\circ/s &\leq \dot{q}_1(t) \leq 100^\circ/s & -70^\circ &\leq q_2(t) \leq 70^\circ \\ -100^\circ/s &\leq \dot{q}_2(t) \leq 100^\circ/s & -28^\circ &\leq q_3(t) \leq 105^\circ \\ -65^\circ &\leq q_2(t) - q_3(t) \leq 65^\circ \end{aligned} \quad (5)$$

again for all $t \in [0, t_f]$.

After a transformation of (2) to a first order system of ODEs, and rewriting the objective (1) in Mayer form by means of an additional ODE, our example problem fits into the framework of the following general optimal control problem:

Find a state/control pair (y, u) in $W^{1,\infty}(0, t_f; \mathbb{R}^{n_y}) \times L^\infty(0, t_f; \mathbb{R}^{n_u})$

$$\text{which minimizes } \varphi(y(t_f)) \quad (6)$$

$$\text{subject to } \dot{y}(t) = f(y(t), u(t), t) \quad \text{a.e. on } [0, t_f] \quad (7)$$

$$\text{initial conditions } y(0) = y_0 \quad (8)$$

$$\text{terminal conditions } B(y(t_f)) = 0 \quad (9)$$

$$\text{control constraints } a \leq u(t) \leq b \quad \text{a.e. on } [0, t_f] \quad (10)$$

$$\text{and state constraints } c \leq S(y(t)) \leq d \quad \text{on } [0, t_f] \quad (11)$$

where $\varphi : \mathbb{R}^{n_y} \rightarrow \mathbb{R}$, $f : \mathbb{R}^{n_y} \times \mathbb{R}^{n_u} \times \mathbb{R} \rightarrow \mathbb{R}^{n_y}$, $B : \mathbb{R}^{n_y} \rightarrow \mathbb{R}^{n_b}$, and $S : \mathbb{R}^{n_y} \rightarrow \mathbb{R}^{n_s}$, and \leq is understood componentwise.

2 The Discretized Problem

We assume that the time interval has been subdivided into $N_T - 1$ sub-intervals, separated by grid points τ^i , $i = 1, \dots, N_T$. In order to transcribe the infinite-dimensional into a finite-dimensional optimization problem, we choose the discretized control components \mathbf{u} as optimization variables. The state equation (7)–(8) is discretized using an explicit Runge-Kutta scheme

$$\mathbf{y} = (\mathbf{y}^1, \dots, \mathbf{y}^{N_T})^\top = (\psi^1(\mathbf{u}), \dots, \psi^{N_T}(\mathbf{u}))^\top \in \mathbb{R}^{N_Y}.$$

We obtain the following non-linear programming problem:

$$\text{Find } \mathbf{u} \in \mathbb{R}^{N_U} \text{ which minimizes } \varphi(\psi^{N_T}(\mathbf{u})) \quad (12)$$

$$\text{subject to } B(\psi^{N_T}(\mathbf{u})) = 0 \quad (13)$$

$$a \leq \mathbf{u} \leq b \quad (14)$$

$$c \leq S(\psi(\mathbf{u})) \leq d. \quad (15)$$

Its dimensions are $N_U = n_u \cdot N_T$, $N_Y = n_y \cdot N_T$, $S : \mathbb{R}^{N_Y} \rightarrow \mathbb{R}^{N_S}$ with $N_S = n_s \cdot N_T$ and B and φ as above.

Optimization software like the SQP solver NPSOL by Gill [GMSW98] is a suitable tool for the numerical solution of (12)–(15). Such software benefits from user-provided exact derivatives of the objective and the constraint functions, i.e. the gradient of $\varphi \circ \psi^{N_T}$ and the Jacobians of $B \circ \psi^{N_T}$ and $S \circ \psi$. The following section addresses the question how this is accomplished using Automatic Differentiation.

3 Using Automatic Differentiation

For a thorough introduction to AD, we refer to the book of Griewank [Gri02]. We use the source transformation variety of AD, which, for a given computer code segment with specified input (independent) and output (dependent) variables, generates an augmented program which additionally computes the Jacobian of the output w.r.t. the input variables.

In contrast to AD, symbolic differentiation, as carried out by computer algebra programs, yields a symbolic formula for the derivative of a given symbolic expression, whose evaluation is often inefficient compared to AD. Numerical differentiation by finite differences, as a third possibility, is easy to implement but it introduces truncation error and is also inefficient if the number of input variables is large.

We only briefly recall that the two modes of AD, forward and reverse, have different characteristic features: Mainly, the forward mode computes one column of the Jacobian at a time, while with the reverse mode, one computes rows. That is, the forward is favorable over the reverse mode whenever the Jacobian is "slim", and vice versa when it is "flat".

In the optimization problem (12)–(15), the dimensions of the respective Jacobians are

$$J_{\varphi \circ \psi^{N_T}}(\mathbf{u}) = J_{\varphi}(\psi^{N_T}(\mathbf{u})) \cdot J_{\psi^{N_T}}(\mathbf{u}) \in \mathbb{R}^{1 \times N_U} \quad (16)$$

$$J_{B \circ \psi^{N_T}}(\mathbf{u}) = J_B(\psi^{N_T}(\mathbf{u})) \cdot J_{\psi^{N_T}}(\mathbf{u}) \in \mathbb{R}^{n_b \times N_U} \quad (17)$$

$$J_{S \circ \psi}(\mathbf{u}) = J_S(\psi^{N_T}(\mathbf{u})) \cdot J_{\psi}(\mathbf{u}) \in \mathbb{R}^{N_s \times N_U}. \quad (18)$$

From the considerations above, it is advisable to use the reverse mode of AD for the first two Jacobians (16)–(17) since $n_b \ll N_U$. As for the state constraints along the trajectory (18), one should use the forward mode if there are many constraints ($n_s > n_u$), and the reverse mode if there are few ($n_s < n_u$), compared to the number of control components. An advanced user may consider to decompose the Jacobians according to the chain rule as indicated above, and to compute $J_{\psi}(\mathbf{u})$ only once, using the forward mode since usually $n_y > n_u$. This corresponds to solving the so-called sensitivity ODE, obtained from the linearization of (7). Then $J_{\psi}(\mathbf{u})$ can be re-used to obtain the Jacobians (16)–(18) by evaluating matrix products, forming $J_{\varphi}(\psi^{N_T}(\mathbf{u})) \in \mathbb{R}^{1 \times n_y}$, $J_B(\psi^{N_T}(\mathbf{u})) \in \mathbb{R}^{n_b \times n_y}$ and $J_S(\psi^{N_T}(\mathbf{u})) \in \mathbb{R}^{n_s \times n_y}$ using, e.g., the reverse mode. Finally, we observe that in the absence of state constraints S along the trajectory, the reverse mode for the computation of (16) and (17) is highly advisable, and in the case of (16), it can be interpreted as solving the continuous adjoint equation, discretized with an integration scheme known to be the so-called transposed scheme of ψ , see Hager [Hag00].

4 Numerical Results

For our robot example, the dimensions are $n_u = 3$, $n_y = 7$, $n_s = 5$ and $n_b = 6$. In the discretization, we have used the classical fourth order explicit Runge-Kutta scheme as the integrator ψ . The final time of $t_f = 2.05$ is close to the minimum turn-around time possible. The time interval $[0, 2.05]$ has been subdivided using $N_T = 82$ equidistant time grid points. For the derivative computation, we have used ODYSSEE's [FP98] reverse mode for (16) and ADIFOR's [BCKMH94] (vector) forward mode for (17)–(18), see also [GW02]. The problem has been solved using NPSOL [GMSW98] and the objective value for this solution is $J = 0.8576925$.

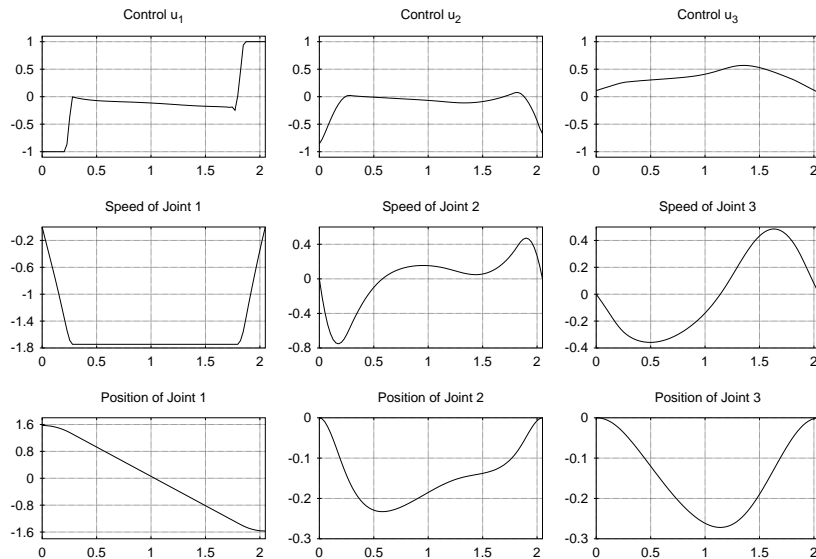


Fig. 2. Optimal solution

References

- [BCKMH94] Bischof, C., Carle, A., Khademi, P., Maurer, A., Hovland, P.: ADIFOR 2.0 User's Guide. Technical Report, Argonne National Laboratory. ANL/MCS-TM-192 (1994)
- [FP98] Faure, C., Papegay, Y.: Odyssee User's Guide Version 1.7. Rapport technique 0224, INRIA (1998)
- [GMSW98] Gill, P., Murray, W., Saunders, M., Wright, M.: User's Guide for NPSOL 5.0: A Fortran Package for Nonlinear Programming. Technical Report NA 98-2, Department of Mathematics, University of California, San Diego (1998)
- [GW02] Griesse, R., Walther, A.: Parametric Sensitivities for Optimal Control Problems using Automatic Differentiation, submitted (2002)

- [Gri02] Griewank, A.: Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation. SIAM Frontiers in Applied Mathematics 19, Philadelphia (2000)
- [Hag00] Hager, W.: Runge-Kutta Methods in Optimal Control and the Transformed Adjoint System, Numer. Math., **87**, 247–282 (2000)
- [Hei98] Heim, A.: Modellierung, Simulation und optimale Bahnplanung von Industrierobotern, Dissertation, Technische Universität München (1998)
- [Kna01] Knauer, M.: Sensitivitätsanalyse verschiedener Gütekriterien bei der optimalen Bahnplanung von Industrierobotern, Diplomarbeit, Universität Bayreuth (2001)