Proceedings of the 36th Chinese Control Conference
July 26-28, 2017, Dalian, China

Implicit Integration with Adjoint Sensitivity Propagation for Optimal Control Problems Involving Differential-Algebraic Equations

Canghua Jiang[1], Kun Xie[1], Zhiqiang Guo[1], Kok Lay Teo[2]

1. School of Electrical Engineering and Automation, Hefei University of Technology, Hefei 230009, China
E-mail: canghua.jiang@gmail.com

2. Department of Mathematics and Statistics, Curtin University, Perth 6845, Australia
E-mail: k.l.teo@curtin.edu.au

Abstract: For the solution of optimal control problem involving an index-1 differential-algebraic equation, an efficient function evaluation algorithm is proposed in this paper. In the evaluation procedure, the state equation is propagated forwards, then, adjoint sensitivity is propagated backwards. Thus, it is computationally more efficient than forward sensitivity propagation when the number of constraints is less than that of optimization variables. In order to reduce Newton iterations, the adjoint sensitivity is derived utilizing the implicit function theorem, and the integration procedure is accelerated by incorporating a predictor-corrector strategy. This algorithm is integrated with a nonlinear programming solver `Ipopt` to solve sequentially the point-to-point optimal control for a Delta robot with constrained motor torque. Numerical experiments demonstrate the efficiency of this algorithm.

Key Words: Differential-algebraic equation, Implicit Runge-Kutta method, Adjoint sensitivity, Optimal control, Delta robot

1 Introduction

Efficient and reliable integrator is an important component of a numerical solver for optimal control problems involving continuous dynamics, especially for real-time applications. For the solution of initial-value problems (IVPs) for ordinary differential equations (ODEs), explicit Runge-Kutta (ERK) methods are preferred. Forward or adjoint (backward) methods can be used to derive gradients. However, ERK methods are inappropriate to differential-algebraic equations (DAEs). In real-world applications, many dynamic systems can be described by DAEs, e.g. constrained mechanical systems, large-scale industrial processes, electrical power systems, etc. For DAEs, backward differentiation formulas (BDF) and implicit Runge-Kutta (IRK) methods are preferred [1]. Compared with ODEs, solution of DAEs is usually slower as Newton iterations are involved. Computing gradients by forward or adjoint method is expensive too, since we have to solve, respectively, IVP for variational or adjoint system in the form of DAE [2]. Thus, function evaluation procedure occupies most of time for the solution of optimal control for a differential-algebraic system.

Recently, it was proposed in [3] to embed an IRK integrator in direct multiple-shooting solver `ACADO` [4] for the solution of optimal control problems for index-1 DAEs. Instead of integrating variational DAEs, forward sensitivities are derived directly from the implicit function theorem. Furthermore, a tangential predictor was proposed in [5] to update stage variables in the IRK integration. As a result, the number of Newton iterations for integration can be reduced to one. It was proved in [5] that this method is equivalent to simultaneous collocation method in [9]. In simultaneous collocation approaches, state equation is discretized and solved simultaneously with the solution of Karush-Kuhn-Tucker conditions. In each iteration of the outer optimization loop, there is only one iteration required for the inner integration. However, a large amount of stage variables for integration need to be solved in outer optimization loop. In order to reduce the computation in optimization procedure, a lifted collocation method was proposed in [10]. It reduces the optimization variable space in simultaneous collocation approach by condensing, then restore it after the reduced optimization problem is solved.

Gradients in [3, 5] are derived by forward sensitivity propagation, where the number of sensitivity variables propagated depends on that of optimization variables. Thus, these methods are inefficient when there are a large amount of optimization variables. In [6], a sequential method embedding an IRK integrator was proposed to solve OCPs for index-1 DAE systems. Gradients were evaluated by propagating adjoint sensitivity in discrete time. Then, this method was extended to higher-index DAEs [7, 8]. This method is more efficient than forward one when the number of constraints is less than that of optimization variables [2]. In this paper, following the idea in [3, 5], we accelerate the computational procedure proposed in [6] by incorporating a predictor-corrector strategy based on sensitivity update. All these algorithms are implemented and integrated with a nonlinear programming (NLP) solver `Ipopt` [11] to construct a sequential solver for optimal control problems involving index-1 DAEs. We then verify its effectiveness and efficiency by solving a point-to-point optimal control problem for a Delta robot.

2 Problem Formulation and Parameterization

Consider the following differential-algebraic system,

$$0 = f\left(t, \dot{x}(t), x(t), z(t), u(t), \zeta\right), \quad t \in (0, T], \tag{1a}$$

where $0 < T \in \mathbb{R}$ is given; $x(t) \in \mathbb{R}^{n_x}$ and $\dot{x}(t) \in \mathbb{R}^{n_x}$ are, respectively, the differential state and its time derivative; $z(t) \in \mathbb{R}^{n_z}$ is the algebraic state; $u(t) \in \mathbb{R}^{n_u}$ and $\zeta \in \mathbb{R}^{n_\zeta}$ are, respectively, the control and system parameter; $f : (0, T] \times \mathbb{R}^{2n_x+n_z+n_u+n_\zeta} \to \mathbb{R}^{n_x+n_z}$ is continuously differentiable. The initial value of $x(t)$ is given as

$$x(0) = x_0(\zeta), \tag{1b}$$

where $x_0 : \mathbb{R}^{n_\zeta} \to \mathbb{R}^{n_x}$ is continuously differentiable. For convenience, let $y(t) \triangleq \dot{x}(t)$. It is assumed that Jacobian $[\partial f/\partial y, \partial f/\partial z]$ is nonsingular. Thus, description (1) represents an index-1 DAE.

Let $\Upsilon \triangleq \{\gamma \in \mathbb{R}^{n_u} | \bar{\gamma}_i \leq \gamma_i \leq \underline{\gamma_i},\ i = 1, \ldots, n_u\}$ and $\mathcal{Z} \triangleq \{z \in \mathbb{R}^{n_\zeta} | \bar{z}_i \leq z_i \leq \underline{z_i},\ i = 1, \ldots, n_\zeta\}$, where γ_i and z_i are the ith elements of γ and z respectively; $\bar{\gamma}_i \in \mathbb{R}$, $\underline{\gamma_i} \in \mathbb{R}$, $\bar{z}_i \in \mathbb{R}$ and $\underline{z_i} \in \mathbb{R}$ are given constants. It is required that $u(t) \in \Upsilon$ for almost all $t \in (0, T]$. Let $\mathcal{U}$ denote the class of all such control functions that are piecewise continuous. Similarly, $\zeta \in \mathcal{Z}$ with $\mathcal{Z}$ being its admissible set. For each $u(t) \in \mathcal{U}$ and $\zeta \in \mathcal{Z}$, we assume that there exists a unique function pair $(x(\cdot|u, \zeta), z(\cdot|u, \zeta))$ satisfying equation (1a) almost everywhere and the initial condition (1b) as well.

For clarity, we only consider terminal constraints:

$$g_m (u(t), \zeta) \triangleq \Phi_m (x(T|u(t), \zeta), \zeta) \geq 0, \qquad m = 1, \ldots, n_c, \tag{2}$$

where $\Phi_m : \mathbb{R}^{n_x} \times \mathbb{R}^{n_\zeta} \to \mathbb{R}$ is continuously differentiable. Let $\mathcal{F}_u$ and $\mathcal{F}_\zeta$ denote, respectively, the sets of $u(t) \in \mathcal{U}$ and $\zeta \in \mathcal{Z}$ satisfying constraints in (2). Then, the optimal control problem investigated in this paper is formulated as follows:

Problem 1 *For the index-1 differential-algebraic system (1), choose a feasible control input $u(t) \in \mathcal{F}_u$ and a feasible parameter $\zeta \in \mathcal{F}_\zeta$ such that the cost functional*

$$g_0 (u(t), \zeta) \triangleq \Phi_0 (x(T|u(t), \zeta), \zeta) \tag{3}$$

is minimized over $\mathcal{F}_u \times \mathcal{F}_\zeta$, where $\Phi_0 : \mathbb{R}^{n_x} \times \mathbb{R}^{n_\zeta} \to \mathbb{R}$ is continuously differentiable.

In this paper, only Mayer problem is investigated as the integration term in Bolza problem can be reduced by introducing an auxiliary differential equation (cf. Section 4).

In general, Problem 1 can only be solved approximately. Let $1 \leq p \in \mathbb{Z}$ and $\Xi^p \triangleq \{\sigma \in \mathbb{R}^{pn_u} | \sigma_i \in \Upsilon, i = 1, \ldots, p\}$ with $\sigma \triangleq [\sigma_1^T, \ldots, \sigma_p^T]^T$. Let us consider piecewise constant control functions with equidistant time grids:

$$u^p(t|\sigma) \triangleq \sum_{\ell=1}^{p} \sigma_\ell \chi_{\mu_\ell^p}(t), \quad t \in (0, T], \tag{4}$$

where $\sigma \in \Xi^p$, $\mu_\ell^p \triangleq \left(\frac{T(\ell-1)}{p}, \frac{T\ell}{p}\right]$ and

$$\chi_\mu(t) \triangleq \begin{cases} 1, & \text{if } t \in \mu, \\ 0, & \text{otherwise.} \end{cases}$$

It is clear that $u^p(\cdot|\sigma) \in \mathcal{U}$. Let $x^p(\cdot|\sigma, \zeta) \triangleq x(\cdot|u^p(\cdot|\sigma), \zeta)$, $y^p(\cdot|\sigma, \zeta) \triangleq y(\cdot|u^p(\cdot|\sigma), \zeta)$ and $z^p(\cdot|\sigma, \zeta) \triangleq z(\cdot|u^p(\cdot|\sigma), \zeta)$. We have from (1) that, for $t \in \mu_\ell^p$ and $\ell = 1, \ldots, p$,

$$0 = f(t, y^p(t|\sigma, \zeta), x^p(t|\sigma, \zeta), z^p(t|\sigma, \zeta), \sigma_\ell, \zeta) \tag{5a}$$

and

$$x^p(0|\zeta) = x_0(\zeta). \tag{5b}$$

The constraints in (2) are transformed correspondingly into:

$$\begin{aligned} g_m^p(\sigma, \zeta) &\triangleq g_m(u^p(\cdot|\sigma), \zeta) \\ &= \Phi_m (x(T|u^p(\cdot|\sigma), \zeta), \zeta) \geq 0, \end{aligned} \tag{6}$$

for $m = 1, \ldots, n_c$. Let Ω^p denote the set of all pairs $(\sigma, \zeta) \in \Xi^p \times \mathcal{Z}$ satisfying constraints in (6), i.e.

$$\Omega^p \triangleq \{(\sigma, \zeta) \in \Xi^p \times \mathcal{Z} | u^p(\cdot|\sigma) \in \mathcal{F}_u, \zeta \in \mathcal{F}_\zeta\}.$$

We define an approximate problem of Problem 1 as follows.

Problem 2 *For differential-algebraic system (5), choose a pair $(\sigma, \zeta) \in \Omega^p$ that minimizes the cost function*

$$g_0^p(\sigma, \zeta) \triangleq g_0(u^p(\cdot|\sigma), \zeta) = \Phi_0 (x(T|u^p(\cdot|\sigma), \zeta), \zeta) \tag{7}$$

over Ω^p.

3 Implicit Integration with Adjoint Sensitivity Propagation

3.1 Implicit Runge-Kutta Integration

For sequential solution of Problem 2, it is required to solve the IVP for DAE (5) corresponding to the current control pair (σ, ζ). For completeness of presentation, we summarize here the integration procedure based on IRK formula [3, 5, 6]. Firstly, the integration step size and order are fixed to ensure a deterministic run time. For $\ell = 1, \ldots, p$, consider DAE (5) defined over interval $t \in \mu_\ell^p$. Let $\tau_{0,0}^\ell \triangleq \frac{T(\ell-1)}{p}$ and $x_{0,0}^\ell \triangleq x^p(\tau_{0,0}^\ell|\sigma, \zeta)$ be the initial condition. Let $1 \leq q \in \mathbb{Z}$ denote the chosen number of integration steps, and divide the interval μ_ℓ^p into q subintervals of equal length $h \triangleq T/p/q$. Then, for $n = 0, \ldots, q-1$, we consider in the following a single integration step with t from $\tau_{n,0}^\ell \triangleq \tau_{0,0}^\ell + nh$ to $\tau_{n+1,0}^\ell$. Let $x_{n,0}^\ell \in \mathbb{R}^{n_x}$ be the value of differential state at $t = \tau_{n,0}^\ell$. From the r-stage Runge-Kutta scheme, we have the following collocation equations on this step, for $i = 1, \ldots, r$:

$$\begin{aligned} 0 &= f_{n,i}^\ell (y_{n,1}^\ell, \ldots, y_{n,r}^\ell, z_{n,i}^\ell | x_{n,0}^\ell, \sigma_\ell, \zeta) \\ &\triangleq f\left(\tau_{n,0}^\ell + c_i h, y_{n,i}^\ell, x_{n,0}^\ell + h \sum_{j=1}^{r} a_{ij} y_{n,j}^\ell, z_{n,i}^\ell, \sigma_\ell, \zeta\right), \end{aligned}$$

where $c_i \in \mathbb{R}$ and $a_{ij} \in \mathbb{R}$ are coefficients in Butcher table [12]; $y_{n,i}^\ell \in \mathbb{R}^{n_x}$ and $z_{n,i}^\ell \in \mathbb{R}^{n_z}$ denote, respectively, the ith stage values of the derivative of differential state and the algebraic state.

For convenience, let

$$k_n^\ell \triangleq \left[(y_{n,1}^\ell)^T, \ldots, (y_{n,r}^\ell)^T, (z_{n,1}^\ell)^T, \ldots, (z_{n,r}^\ell)^T\right]^T$$

be the stage variable and $\xi_n^\ell \triangleq \left[(x_{n,0}^\ell)^T, \sigma_\ell^T, \zeta^T\right]^T$. Given ξ_n^ℓ and an initial guess $k_n^{\ell[0]}$, k_n^ℓ satisfying equation

$$\begin{aligned} 0 &= F_n^\ell (k_n^\ell | \xi_n^\ell) \\ &\triangleq \begin{bmatrix} f_{n,1}^\ell (y_{n,1}^\ell, \ldots, y_{n,r}^\ell, z_{n,1}^\ell | \xi_n^\ell) \\ \vdots \\ f_{n,r}^\ell (y_{n,1}^\ell, \ldots, y_{n,r}^\ell, z_{n,r}^\ell | \xi_n^\ell) \end{bmatrix} \end{aligned} \tag{8}$$

can be computed by Newton iterations:

$$k_n^{\ell[i]} = k_n^{\ell[i-1]} - \left(M_n^{\ell[i-1]}\right)^{-1} F_n^\ell \left(k_n^{\ell[i-1]} | \xi_n^\ell\right), \qquad i = 1, \ldots, L, \tag{9a}$$

where $1 \le L \in \mathbb{Z}$ is a fixed iteration number and

$$M_n^{\ell[i-1]} \triangleq \frac{\partial F_n^\ell}{\partial k_n^\ell}\left(k_n^\ell | \xi_n^\ell\right)\Big|_{k_n^\ell = k_n^{\ell[i-1]}}. \tag{9b}$$

Then, by quadrature, we have the approximation to the differential state at $t = \tau_{n+1,0}^\ell$,

$$x_{n+1,0}^\ell = x_{n,0}^\ell + h\sum_{j=1}^r b_j y_{n,j}^\ell, \tag{10}$$

where b_j is the coefficient in Butcher table [12].

In sequential case, $x_{n,0}^\ell$ is not an independent variable for optimization as in multiple shooting case. We can derive $x_{q,0}^\ell$ from $x_{0,0}^\ell$ by (10) recursively with n from 0 to $q-1$. Then, with initialization $x_{0,0}^1 = x_0(\zeta)$, $x_{q,0}^p$ can be computed by executing this q-step recursion repeatedly, for $\ell = 1, \ldots, p$, and following continuous conditions $x_{0,0}^{\ell+1} = x_{q,0}^\ell$, for $\ell = 1, \ldots, p-1$. As a result, the terminal state can be approximated by $x^p(T|\sigma,\zeta) \approx x_{q,0}^p$. The objective and constraint functions $g_m^p(\sigma,\zeta)$, for $m = 0, \ldots, n_c$, can, then, be evaluated.

3.2 Gradient Evaluation by Adjoint Sensitivity Propagation

Traditional methods for gradient evaluation is to integrate the variational or adjoint system of (5) [2]. However, these two systems are DAEs too, and it is computationally expensive to integrate them. To avoid additional Newton iterations for integration, we can use the implicit function theorem [3, 6]. Specifically, k_n^ℓ is a function of ξ_n^ℓ from (8). Then, we can obtain from the implicit function theorem that

$$\frac{\partial k_n^\ell}{\partial \xi_n^\ell} = -\left(M_n^\ell\right)^{-1}\frac{\partial F_n^\ell}{\partial \xi_n^\ell}. \tag{11}$$

Clearly, from (10) and (11), we have

$$\frac{\partial x_{n+1,0}^\ell}{\partial \xi_n^\ell} = h\sum_{j=1}^r b_j \frac{\partial y_{n,j}^\ell}{\partial \xi_n^\ell} + \begin{bmatrix} I_{n_x} & 0 & 0 \end{bmatrix}, \tag{12}$$

where $I_{n_x} \in \mathbb{R}^{n_x \times n_x}$ is an identity matrix. Then, from chain rule and the forward propagation procedure of algorithmic differentiation, gradient of the objective or constraint function with respective to (σ,ζ) can be derived [3]. Note that a prerequisite of applying (11) is that the Newton iterations (9) converge. Decomposition of M_n^ℓ required in (11) can be obtained from (9) directly.

As gradients can also be derived by integrating the adjoint system, a corresponding method was proposed in [6] by combining (11) with adjoint sensitivity propagation. This method is more efficient when there are fewer constraints than optimization variables. Let us revisit its computational procedure here. Let $\bar{G} \in \mathbb{R}^{n_d \times (1+n_c)}$, for $1 \le n_d \le 1 + n_c$, be a directional matrix, and $\bar{x}_{n,0}^\ell \in \mathbb{R}^{n_d \times n_x}$ be the corresponding adjoint variable matrix. Similar to [6], we introduce the following adjoint equation,

$$\bar{x}_{n,0}^\ell = \bar{x}_{n+1,0}^\ell \frac{\partial x_{n+1,0}^\ell}{\partial x_{n,0}^\ell}, \tag{13a}$$

for $\ell = p, \ldots, 1$ and $n = q-1, \ldots, 0$, with terminal condition

$$\bar{x}_{q,0}^p = \frac{\partial \bar{G}[g_0^p, \ldots, g_{n_c}^p]^T}{\partial x^p(T)}, \tag{13b}$$

and continuous condition

$$\bar{x}_{q,0}^{\ell-1} = \bar{x}_{0,0}^\ell, \qquad \ell \neq 1, \tag{13c}$$

where $\frac{\partial x_{n+1,0}^\ell}{\partial x_{n,0}^\ell}$ is computed from (12).

Proposition 1 *Given initial condition $x^p(0) = x_0(\zeta)$, and parameter $\omega \triangleq [\sigma_1^T, \ldots, \sigma_p^T, \zeta^T]^T$. If the differential state $x^p(T)$ of DAE (5) is approximated by $x_{q,0}^p$, which is computed from the IRK scheme (8)-(10) with Newton iterations (9) converged, then, for a given $\bar{G} \in \mathbb{R}^{n_d \times (n_c+1)}$, the directional derivative of the objective and constraint function vector $[g_0^p, \ldots, g_{n_c}^p]^T$ with respect to ω can be approximated by*

$$\frac{\mathrm{d}\bar{G}[g_0^p, \ldots, g_{n_c}^p]^T}{\mathrm{d}\omega} \approx \bar{\omega}, \tag{14a}$$

where $\bar{\omega} = [\bar{\sigma}_1, \ldots, \bar{\sigma}_p, \bar{\zeta}]$ can be derived by

$$\begin{cases} \bar{\sigma}_\ell = \displaystyle\sum_{n=q-1}^{0} \bar{x}_{n+1,0}^\ell \frac{\partial x_{n+1,0}^\ell}{\partial \sigma_\ell}, \quad \ell = p, \ldots, 1, \\ \bar{\zeta} = \bar{x}_{0,0}^1 \dfrac{\partial x_0}{\partial \zeta} + \displaystyle\sum_{\ell=p}^{1}\sum_{n=q-1}^{0} \bar{x}_{n+1,0}^\ell \frac{\partial x_{n+1,0}^\ell}{\partial \zeta} \\ \quad + \dfrac{\partial \bar{G}[g_0^p, g_1^p, \ldots, g_{n_c}^p]^T}{\partial \zeta}. \end{cases} \tag{14b}$$

In (14b), for $\ell = p, \ldots, 1$ and $n = q-1, \ldots, 0$, $\bar{x}_{n+1,0}^\ell$ can be propagated backwards following adjoint equation (13); $\frac{\partial x_{n+1,0}^\ell}{\partial \sigma_\ell}$ and $\frac{\partial x_{n+1,0}^\ell}{\partial \zeta}$ are derived from (12).

Proof. Take $\bar{\sigma}_\ell$, $\ell = p, \ldots, 1$, for example. Following the tradition of backward algorithmic differentiation [13], $\bar{x}_{n+1,0}^\ell = \frac{\mathrm{d}\bar{G}[g_0^p,\ldots,g_{n_c}^p]^T}{\mathrm{d}x_{n+1,0}^\ell}$ for $n = q-1, \ldots, 0$. For $\ell = p$ and $n = q-1$, we have (13b). Then, considering integration formulas (9) and (10), and partial derivatives in (12), we know adjoint equation (13a) holds true from chain rule. The equality in (13c) is obvious from continuity. Let $\bar{\sigma}_\ell = 0$ initially. Then, $\bar{\sigma}_\ell$ can be computed by $\bar{\sigma}_\ell + = \bar{x}_{n+1,0}^\ell \frac{\partial x_{n+1,0}^\ell}{\partial \sigma_\ell}$ with $n = q-1, \ldots, 0$. Here, '$a + = b$' denotes assigning $a + b$ to a. Hence, the first equality in (14b) holds true. The second one can be proved similarly. □

Then, gradients can be derived following two steps: 1) integrating the nominal system (5) forwards; 2) propagating the adjoint system (13) backwards.

3.3 Acceleration by a Predictor-Corrector Strategy

For sequential solution of Problem 2, L Newton iterations for integration of (5) are executed in each iteration for the solution of ω. If ω does not change two much between two iterations, those inner L iterations are unnecessary. Let $\Delta\sigma_\ell \triangleq \sigma_\ell - \sigma_\ell^*$, $\Delta\zeta \triangleq \zeta - \zeta^*$ and $\Delta x_{n,0}^\ell \triangleq x_{n,0}^\ell - x_{n,0}^{\ell*}$, where v^* denotes cached value of v, which is obtained from the last optimization iteration. In analogy to the lifted implicit integrator proposed in [5], initial estimate of $k_n^{\ell[0]}$ in

(9) can be updated by the following tangential prediction,

$$k_n^{\ell[0]} = k_n^{\ell*} + \frac{\partial k_n^{\ell*}}{\partial \sigma_\ell}\Delta\sigma_\ell + \frac{\partial k_n^{\ell*}}{\partial \zeta}\Delta\zeta + \frac{\partial k_n^{\ell*}}{\partial x_{n,0}^\ell}\Delta x_{n,0}^\ell, \quad (15)$$

where $k_n^{\ell*}$, $\frac{\partial k_n^{\ell*}}{\partial \sigma_\ell}$ and $\frac{\partial k_n^{\ell*}}{\partial \zeta}$ are evaluated at $\xi_n^{\ell*}$. Unlike in multiple shooting methods, $x_{n,0}^\ell$ here cannot be obtained directly. However, since $\Delta x_{0,0}^1 = 0$, $x_{n,0}^\ell$ with respect to current ω can be determined step-by-step from the initial condition (5b) by integration. After that, $k_n^{\ell[0]}$ can be corrected in further by a few Newton iterations (one Newton iteration in the limit).

In many NLP algorithms, ω changes not only between two NLP iterations but also in procedures like step-size trial and feasibility restoration [11]. To circumvent this complexity, we introduce a heuristic rule in Algorithm 1 before Newton iterations (9), which will make the tangential prediction (15) more reliable.

Algorithm 1 Tangential prediction

```
Input: $\omega$, $\omega^*$, $k_n^{\ell*}$, $\frac{\partial k_n^{\ell*}}{\partial \xi_n^\ell}$ and $\Delta x$
Output: $k_n^\ell$, $L_r$
 1: $\Delta\sigma_\ell \leftarrow \sigma_\ell - \sigma_\ell^*$,    $\Delta\zeta \leftarrow \zeta - \zeta^*$
 2: if $\frac{\|\omega-\omega^*\|^2}{\|\omega^*\|^2} < \epsilon^2$ then
 3:     $k_n^\ell \leftarrow k_n^{\ell*} + \frac{\partial k_n^{\ell*}}{\partial \sigma_\ell}\Delta\sigma_\ell + \frac{\partial k_n^{\ell*}}{\partial \zeta}\Delta\zeta + \frac{\partial k_n^{\ell*}}{\partial x_{n,0}^\ell}\Delta x$, and $L_r \leftarrow 1$
 4: else
 5:     $k_n^\ell \leftarrow k_n^{\ell*}$,    $L_r \leftarrow L$
 6: end if
```

In Algorithm 1, '$\leftarrow$' denotes assignment, ω^* denotes the parameter cached, and $0 < \epsilon < 1$ is a real constant. When the new parameter ω is in a small neighborhood (controlled by ϵ) of the cached one ω^*, the integrator switches to the predictor-corrector mode with the number of Newton iterations L_r set to one.

Then, a function evaluation algorithm with gradient computation, which employs the predictor-corrector strategy for solving the index-1 DAE (5), can be derived as Algorithm 2. In the input and output arguments of Algorithm 2, $\ell \in \{1,\dots,p\}$, $n \in \{0,\dots,q-1\}$ and $m \in \{0,\dots,n_c\}$. In Steps 29 and 34, e_m, for $m = 0,\dots,n_c$, is the mth column of a $(1+n_c)\times(1+n_c)$ identity matrix. The input and output arguments, and computational steps marked by '[PC]' are only required in the predictor-corrector mode. If $\epsilon = 0$ in Algorithm 1 and the aforementioned arguments and steps are removed, this algorithm is reduced to the one in [6], which is based on standard IRK integration. In the following, we call the algorithm with predictor-corrector strategy as Scheme II while the standard one Scheme I.

Remark 1 *The correction procedure in Algorithm 2 is with respect to prediction k_n^ℓ. Thus, this predictor-corrector strategy is different from Scheme A in Algorithm 2 in [5], where both the prediction and correction are with respect to $k_n^{\ell*}$. It was proved in [5] that $\Delta\omega$ with Δk_n^ℓ, for $\ell = 1,\dots,p$ and $n = 0,\dots,q-1$, computed by the latter algorithm constitutes a Newton step in simultaneous collocation method [9]. Hence, it has similar convergency properties as simultaneous collocation method. It is not direct to prove the equivalence between Algorithm 2 and simultaneous collocation method. But, numerical experiments demonstrate that Algorithm 2 converges faster than Scheme A in Algorithm 2 in [5].*

Algorithm 2 Objective and constraint evaluation with adjoint sensitivity propagation

```
Input: $x_0(\zeta)$, $\omega$ and $k_n^{\ell*}$;        [PC] $x_{n,0}^{\ell*}$, $\frac{\partial k_n^{\ell*}}{\partial \xi_n^\ell}$ and $\omega^*$
Output: $g_m^p$, $\frac{\mathrm{d}g_m^p}{\mathrm{d}\omega}$ and $k_n^{\ell*}$;        [PC] $x_{n,0}^{\ell*}$, $\frac{\partial k_n^{\ell*}}{\partial \xi_n^\ell}$ and $\omega^*$
 1: $x_{0,0}^1 \leftarrow x_0(\zeta)$
 2: [PC] $\Delta x \leftarrow 0$
 3: for $\ell = 1,\dots,p$ do
 4:     for $n = 0,\dots,q-1$ do
 5:         compute $k_n^\ell$ and set $L_r$ following Algorithm 1
 6:         for $i = 1,\dots,L_r$ do
 7:             compute $M_n^\ell = \frac{\partial F_n^\ell}{\partial k_n^\ell}$ and its decomposition
 8:             $k_n^\ell \leftarrow k_n^\ell - \left(M_n^\ell\right)^{-1} F_n^\ell$
 9:         end for
10:         $x_{n+1,0}^\ell \leftarrow x_{n,0}^\ell + h\sum_{i=1}^r b_i y_{n,i}^\ell$,    $k_n^{\ell*} \leftarrow k_n^\ell$
11:         [PC] $\Delta x \leftarrow x_{n+1,0}^\ell - x_{n+1,0}^{\ell*}$
12:         [PC] $x_{n+1,0}^{\ell*} \leftarrow x_{n+1,0}^\ell$
13:         compute $\frac{\partial k_n^\ell}{\partial \xi_n^\ell}$ from (11) with the decomposition of $M_n^\ell$
            computed from step 7
14:         [PC] $\frac{\partial k_n^{\ell*}}{\partial \xi_n^\ell} \leftarrow \frac{\partial k_n^\ell}{\partial \xi_n^\ell}$
15:         compute $\frac{\partial x_{n+1,0}^\ell}{\partial \xi_n^\ell}$ from (12)
16:     end for
17:     if $\ell < p$ then
18:         $x_{0,0}^{\ell+1} \leftarrow x_{q,0}^\ell$
19:     end if
20: end for
21: $x^p(p) \leftarrow x_{q,0}^p$, and compute $g_m^p$
22: $\bar{x}_{q,0}^p \leftarrow \frac{\partial [g_0^p,\dots,g_{n_c}^p]^T}{\partial x_{q,0}^p}$,    $\bar{\zeta} \leftarrow \frac{\partial [g_0^p,\dots,g_{n_c}^p]^T}{\partial \zeta}$
23: for $\ell = p,\dots,1$ do
24:     $\bar{\sigma}_\ell \leftarrow 0$
25:     for $n = q-1,\dots,0$ do
26:         $\bar{\sigma}_\ell \leftarrow \bar{\sigma}_\ell + \bar{x}_{n+1,0}^\ell \frac{\partial x_{n+1,0}^\ell}{\partial \sigma_\ell}$,    $\bar{\zeta} \leftarrow \bar{\zeta} + \bar{x}_{n+1,0}^\ell \frac{\partial x_{n+1,0}^\ell}{\partial \zeta}$
27:         $\bar{x}_{n,0}^\ell \leftarrow \bar{x}_{n+1,0}^\ell \frac{\partial x_{n+1,0}^\ell}{\partial x_{n,0}^\ell}$
28:     end for
29:     $\frac{\mathrm{d}g_m^p}{\mathrm{d}\sigma_\ell} \leftarrow \bar{\sigma}_\ell^T e_m$
30:     if $\ell > 1$ then
31:         $\bar{x}_{q,0}^{\ell-1} \leftarrow \bar{x}_{0,0}^\ell$
32:     end if
33: end for
34: $\bar{\zeta} \leftarrow \bar{\zeta} + \bar{x}_{0,0}^1 \frac{\partial x_0}{\partial \zeta}$,    $\frac{\mathrm{d}g_m^p}{\mathrm{d}\zeta} \leftarrow \bar{\zeta}^T e_m$
35: [PC] $\omega^* \leftarrow \omega$
```

3.4 Complexity Analysis

In order to show the efficiency of Algorithm 2, we analyze the computational complexity of Schemes I and II formally in this subsection.

Assume that LU decomposition is used to solve the linear equation (9). Suppose that LU decomposition for a $n \times n$ matrix requires approximately $\frac{2}{3}n^3$ flops and back substitution requires, accordingly, $2n^2$. Then, one Newton iteration in (9) requires approximately $\frac{2}{3}n_k^3 + 2n_k^2$ flops, where $n_k \triangleq r(n_x + n_z)$ is the dimension of stage variable k_n^ℓ. As the first two steps for sensitivity propagation, computations in (11) and (12) require, respectively, $2n_k^2 n_\xi$ and $2rn_x n_\xi$

flops, where $n_\xi \triangleq n_x + n_u + n_\zeta$. The backward propagation of adjoint variables in (13) requires approximately $2pqn_x^2(n_c+1)$ flops. If $n_c+1 < n_\omega$, where $n_\omega \triangleq pn_u + n_\zeta$ represents the number of optimization variables, this propagation procedure is more efficient than forward sensitivity propagation, where $2pqn_x^2 n_\omega$ flops are needed approximately. For Scheme II, additional $3pqn_k n_\xi$ flops are used for tangential prediction. The major computational costs of Schemes I and II are compared in Table 1. As the majority of computation is spent in Newton iterations, the computational speed of Scheme II is increased by a factor of $L-1$ in the limit compared with Scheme I.

Table 1 also presents the memory usage in Schemes I and II, which includes n_x floating-point numbers to store the current value of $x_{n,0}^\ell$, $(n_c+1)(n_x+1)$ to store all the adjoint variables, $n_k(n_k+n_\xi)$ to store intermediate variables and pqn_k to store k_n^ℓ. Besides this, $pqn_x n_\xi$ memory is needed to store all the $\frac{\partial x_{n+1,0}^\ell}{\partial \xi_n^\ell}$. For Scheme II, additional memory $n_x + n_\omega + pqn_x + pqn_k n_\xi$ is used to store $\Delta x_{n,0}^\ell$, ω^*, $x_{n,0}^{\ell *}$ and $\frac{\partial k_n^{\ell *}}{\partial \xi_n^\ell}$. Clearly, Scheme II requires more memory.

Table 1: Comparisons of computational complexity and memory usage for function evaluation

Scheme	I	II
Complex. (flops)	$pq[L\frac{2}{3}n_k^3 + (n_\xi + L)2n_k^2 + 2(r+n_c+1)n_x n_\xi]$	$pq[\frac{2}{3}n_k^3 + (n_\xi + 1)2n_k^2 + 2(r+n_c+1)n_x n_\xi + 3n_k n_\xi]$
Memory (floating-point numbers)	$n_x + (n_x+1)(n_c+1) + n_k(n_k+n_\xi) + pq(n_k + n_x n_\xi)$	$2n_x + (n_x+1)(n_c+1) + n_k(n_k+n_\xi) + n_\omega + pq(n_x+n_k)(1+n_\xi)$

4 Numerical Example

For the solution of Problem 2, we choose NLP solver `Ipopt` (version 3.12.0) [11], which is open-source and implements an interior-point filter line-search algorithm in C++. A member function is implemented to evaluate the objective and constraint functions with their gradients based on Algorithm 2. Then, approximate Hessian of Lagrangian is provided by a limited-memory quasi-Newton method (L-BFGS) integrated in `Ipopt` [11]. The solution algorithm is implemented in C++, and compiled by `g++` of version 5.2.1. The program is running in a 32-bit version of `ubuntu` 15.10 with an Intel i7-4790 processor at 3.60GHz.

To verify Schemes I and II, we investigate the optimal control problem for a Delta robot [14], which is a constrained mechanical system. The model of the robot is much simplified as in [15]. Its three arms are of length $l_a = 0.2$m, and three forearms of length $l_f = 0.6$m. The mass $m_r = 5 \times 10^{-2}$kg and inertia $J_r = 0.1$kgm^2 of the robot are concentrated in its nacelle and motor-arms, respectively. Let $\mathbf{p} \triangleq [p_1, p_2, p_3]^T \in \mathbb{R}^3$ denote the position of the nacelle, $\boldsymbol{\vartheta} \triangleq [\vartheta_1, \vartheta_2, \vartheta_3]^T \in \mathbb{R}^3$ be the angular position of arms, and $(\alpha_1, \alpha_2, \alpha_3) = (0, \frac{2\pi}{3}, \frac{4\pi}{3})$ denote the triple of mounting angles of arms. To ensure the length of each forearms is invariant during moving of nacelle, the geometry of the robot is constrained by

$$f_{c_i} = \|\mathbf{p} - \mathbf{R}_i \mathbf{a}_i\|^2 - l_f^2 = 0, \quad i = 1,2,3, \tag{16}$$

where

$$\mathbf{R}_i = \begin{bmatrix} \cos\alpha_i & -\sin\alpha_i & 0 \\ \sin\alpha_i & \cos\alpha_i & 0 \\ 0 & 0 & 1 \end{bmatrix} \text{ and } \mathbf{a}_i = l_a \begin{bmatrix} \cos\vartheta_i \\ 0 \\ \sin\vartheta_i \end{bmatrix}.$$

The robot has kinetic energy $\mathcal{K} = \frac{1}{2} m_r \dot{\mathbf{p}}^T \dot{\mathbf{p}} + \frac{1}{2} J_r \dot{\boldsymbol{\vartheta}}^T \dot{\boldsymbol{\vartheta}} + \mathbf{z}^T \mathbf{f}_c$ and potential energy $\mathcal{V} = m_r g p_3 + \mathbf{T}^T \boldsymbol{\vartheta}$, where $\mathbf{z} \in \mathbb{R}^3$ is the Lagrange multiplier, $\mathbf{f}_c \triangleq [f_{c_1}, f_{c_2}, f_{c_3}]^T$ is the equality constraint (16) and $\mathbf{T} \triangleq [T_1, T_2, T_3]^T \in \mathbb{R}^3$ is the motor torque. Denote $\mathbf{q} \triangleq [\boldsymbol{\vartheta}^T, \mathbf{p}^T]^T$ as the position of the robot. From the Lagrange's equation, and by differentiating (16) twice with respect to t, we obtain the following equation of the robot:

$$\begin{bmatrix} \mathbf{M}_r & -\frac{\partial \mathbf{f}_c}{\partial \mathbf{q}}^T \\ -\frac{\partial \mathbf{f}_c}{\partial \mathbf{q}} & \mathbf{0}_{3\times 3} \end{bmatrix} \begin{bmatrix} \ddot{\mathbf{q}} \\ \mathbf{z} \end{bmatrix} = \begin{bmatrix} -\frac{\partial \mathcal{V}}{\partial \mathbf{q}} \\ \frac{\partial}{\partial \mathbf{q}} \left(\frac{\partial \mathbf{c}}{\partial \mathbf{q}} \dot{\mathbf{q}} \right) \dot{\mathbf{q}} \end{bmatrix}, \tag{17}$$

where $\mathbf{M}_r \triangleq \operatorname{diag}([J_r, J_r, J_r, m_r, m_r, m_r])$. Description (17) is an index-1 DAE, which is set up directly without the need to solve equations in (16) as in [14].

The motor torque $\mathbf{T}$ is bounded, which satisfies

$$-5\text{Nm} \le T_i \le 5\text{Nm}, \quad i = 1,2,3. \tag{18}$$

The control objective is to force the nacelle moving from the initial position $\mathbf{p}_0 = [0, 0, -0.6]$m (with zero velocity and acceleration) to terminal position $\mathbf{p}_f = [0.15, 0.15, -0.7]$m in $T = 0.225$ second consuming little energy. Specifically, the cost functional is

$$g_0 = \int_0^T \left\{ (1-\rho)\|\mathbf{p}(t) - \mathbf{p}_f\|^2 + \rho \|\mathbf{T}(t)\|^2 \right\} \mathrm{d}t, \tag{19}$$

where $\rho = 0.1$. Since g_0 has two integration terms, we introduce two auxiliary differential equations to rewrite it into Mayer form. Then, DAE (17) has 14 differential states and 3 algebraic ones ($\mathbf{z}$) in all. The nacelle is required to reach position $\mathbf{p}_f$ at terminal time. Thus, there is only one constraint

$$\mathbf{p}(T) - \mathbf{p}_f = 0$$

in our problem.

Although Gaussian quadrature was used in IRK methods in [5, 15], we choose in this paper the 3-stage Radau IIA quadrature (order 5), which is stiffly accurate and much preferred in [1, 9]. In our experiments, $p = 5$ and $q = 2$. Solution results after 470 iterations of `Ipopt` are shown in Figs. 1 and 2, where the blue solid curves and the blue dot ones represent results obtained, respectively, from Scheme I and Scheme II with $L = 5$ and $\epsilon = 0.1$ in Algorithm 1.

We can observe that the nacelle moves smoothly from $\mathbf{p}_0$ to $\mathbf{p}_f$ in 0.225 second, and the error of terminal position is negligible. The joint angles of arms also change smoothly from their initial values to their terminal ones. The motor torque resides on its upper bound (denoted by black dash-dot line) until 0.045 second (cf. T_3 in Fig. 2). The control torques computed by Schemes I and II are in considerable agreement. In Table 2, average solution time based on Schemes I and II is compared. Clearly, solution time is much saved for Scheme II.

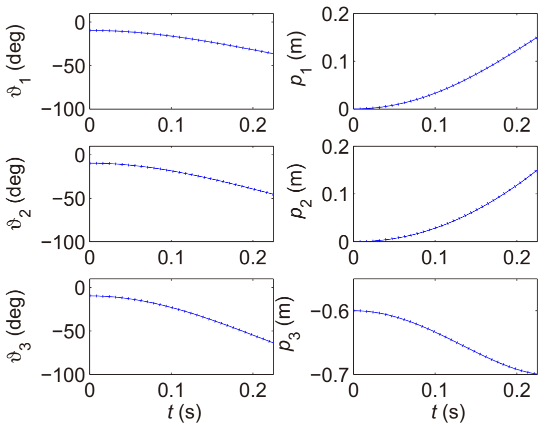

Fig. 1: Trajectories of angular ϑ and position $\mathbf{p}$

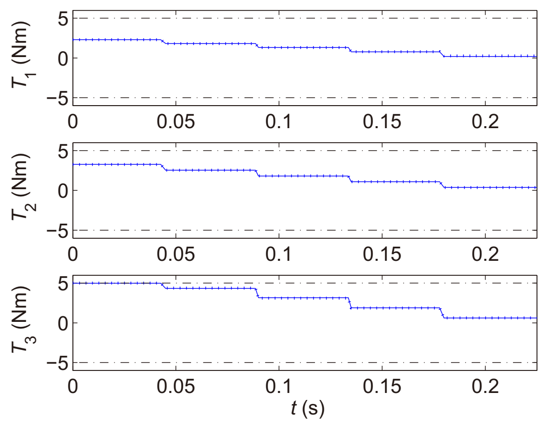

Fig. 2: Trajectories of motor torque $\mathbf{T}$

5 Conclusions

In this paper, function evaluation and gradient computation are investigated for the solution of optimal control problem involving an index-1 DAE. An efficient algorithm based on IRK integration is proposed by applying the implicit function theorem and a predictor-corrector technique. Then, this algorithm is integrated with a NLP solver `Ipopt` [11] to form a sequential optimal control solver. Different from algorithms in [3, 5], gradients in our algorithm are derived by propagating adjoint sensitivities. Hence, it is computationally more efficient than those in [3, 5] when the problem has fewer constraints than optimization variables. Compared with [6], the integration procedure in our algorithm requires fewer Newton iterations. Numerical experiments for the point-to-point optimal control of a Delta robot demonstrate the effectiveness of our algorithm. Algorithmic analysis and experimental results show that our algorithm incorporating the predictor-corrector technique is more efficient than [6] in the price of more memory usage.

Table 2: Averaged solution time (milliseconds) for one iteration of `Ipopt`

Scheme	I	II
CPU time (w/o function evaluations)	0.668	0.515
CPU time for function evaluations	9.90	2.32
Total CPU time	10.6	2.84

References

[1] K. E. Brenan, S. L. Campbell, and L. R. Petzold, *Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations*. Philadelphia: SIAM, 1996.

[2] L. Petzold, S. Li, Y. Cao, and R. Serban, Sensitivity analysis of differential-algebraic equations and partial differential equations, *Computers & Chemical Engineering*, 30:1553–1559, 2006.

[3] R. Quirynen, M. Vukov, and M. Diehl, Auto generation of implicit integrators for embedded NMPC with microsecond sampling times, in *Proceedings of 4th IFAC nonlinear model predictive control conference*, 2012: 175–180.

[4] B. Houska, H. J. Ferreau, and M. Diehl, ACADO Toolkit—an open source framework for automatic control and dynamic optimization, *Optimal Control Applications and Methods*, 32(3):298–312, 2011.

[5] R. Quirynen, S. Gros, and M. Diehl, Lifted implicit integrators for direct optimal control, in *Proceedings of 2015 IEEE 54th Annual Conference on Decision and Control*, 2015: 3212–3217.

[6] R. Pytlak, Runge-Kutta based procedure for the optimal control of differential-algebraic equations, *Journal of Optimization Theory and Applications*, 97(3):675–705, 1998.

[7] R. Pytlak, Numerical procedure for optimal control of higher index DAEs, *Discrete and Continuous Dynamical Systems—Series A (DCDS-A)*, 29(2):647–670, 2011.

[8] R. Pytlak and T. Zawadzki, On solving optimal control problems with higher index DAEs, *Optimization Methods & Software*, 29:1139–1162, 2014.

[9] L. T. Biegler, *Nonlinear Programming: Concepts, Algorithms, and Applications to Chemical Processes*. Philadelphia: SIAM, 2010.

[10] R. Quirynen, S. Gros, B. Houska, and M. Diehl, Lifted collocation integrators for direct optimal control in ACADO Toolkit, `http://www.optimization-online.org/DB_FILE/2016/05/5468.pdf`

[11] A. Wächter and L. T. Biegler, On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming, *Mathematical Programming, Ser. A*, 106:25–57, 2006.

[12] E. Hairer and G. Wanner, *Solving Ordinary Differential Equations: II Stiff and Differential-Algebraic Problems*. Berlin: Springer-Verlag, 1991.

[13] A. Griewank, *Evaluating Derivatives, Principles and Techniques of Algorithmic Differentiation*. Philadelphia: SIAM, 2000.

[14] A. Codourey, Dynamic modeling of parallel robots for computed-torque control implementation, *The International Journal of Robotics Research*, 17(12):1325–1336, 1998.

[15] R. Quirynen, S. Gros, and M. Diehl, Inexact Newton based lifted implicit integrators for fast nonlinear MPC, in *Proceedings of 5th IFAC nonlinear model predictive control conference*, 2015: 32–38.