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,

\[
0 = f(t, \dot{x}(t), x(t), z(t), u(t), \zeta), \quad t \in (0, T],
\]

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

\[
x(0) = x_0(\zeta),
\]
where \( x_0 : \mathbb{R}^{n_c} \rightarrow \mathbb{R}^{n_x} \) is continuously differentiable. For convenience, let \( y(t) \triangleq \dot{x}(t) \). It is assumed that Jacobian \( \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z} \) is nonsingular. Thus, description (1) represents an index-1 DAE.

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

For clarity, we only consider terminal constraints:

\[
\begin{align*}
g_m (u(t), \zeta) &\triangleq \Phi_m (x(T|u(t), \zeta), \zeta) \\
&= 0, \quad m = 1, \ldots, n_c, (2)
\end{align*}
\]

where \( \Phi_m : \mathbb{R}^{n_u} \times \mathbb{R}^{n_z} \rightarrow \mathbb{R} \) is continuously differentiable. Let \( \mathcal{F}_u \) and \( \mathcal{F}_z \) denote, respectively, the sets of \( u(t) \in \mathcal{U} \) and \( \zeta \in Z \) 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 \( \zeta \in \mathcal{F}_z \) such that the cost functional

\[
\begin{align*}
g_0 (u(t), \zeta) &\triangleq \Phi_0 (x(T|u(t), \zeta), \zeta) \\
&= 0
\end{align*}
\]

is minimized over \( \mathcal{F}_u \times \mathcal{F}_z \) where \( \Phi_0 : \mathbb{R}^{n_u} \times \mathbb{R}^{n_z} \rightarrow \mathbb{R} \) 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 \( \mathbb{X} \triangleq \{ \sigma \in \mathbb{R}^{n_u}, |\sigma| \in \mathcal{Y}, i = 1, \ldots, p \} \) with \( \sigma \triangleq [\sigma_t^1 \ldots, \sigma_T^p]^T \). Let us consider piecewise constant control functions with equidistant time grids:

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

where \( \sigma \in \mathbb{X}, \mu_{\ell} \triangleq \frac{T(t-1)}{p}, T p \) and

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

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

\[
0 = f(t, \nu^p(t|\sigma), \sigma_T^p(t|\sigma), \sigma_{\ell}, \zeta), (5a)
\]

and

\[
x^p(0|\zeta) = x_0(\zeta). (5b)
\]

The constraints in (2) are transformed correspondingly into:

\[
g^p_m (\sigma, \zeta) \triangleq g_m (\nu^p(|\sigma|, \zeta)) \geq 0, \quad m = 1, \ldots, n_c. (6)
\]

for \( m = 1, \ldots, n_c \). Let \( \Omega_p \) denote the set of all pairs \((\sigma, \zeta)\) in \( \mathbb{X} \times Z \) satisfying constraints in (6), i.e.

\[
\Omega_p \triangleq \{ (\sigma, \zeta) \in \mathbb{X} \times Z | u^p(|\sigma|) \in \mathcal{F}_u, \zeta \in \mathcal{F}_z \}.
\]

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

\[
g^p_0 (\sigma, \zeta) \triangleq g_0 (u^p(|\sigma|), \zeta) = \Phi_0 (x(T|u^p(|\sigma|), \zeta), \zeta), (7)
\]

over \( \Omega_p \).

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 \((\sigma, \zeta)\). 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 \( t \in \mu_{\ell} \). Let \( \tau_{\ell,0} \triangleq (T_{\ell-1}) / p \) and \( x^p_{\ell,0} \triangleq x^p(\tau_{\ell,0}|\sigma, \zeta) \) be the initial condition. Let \( 1 \leq q \in \mathbb{Z} \) denote the chosen number of integration steps, and divide the interval \( \mu_{\ell} \) 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 \( t_{\ell,n} = \tau_{\ell,n} \). Let \( x^p_{\ell,n} \in \mathbb{X} \) be the value of differential state at \( t = \tau_{\ell,n} \). From the \( r \)-stage Runge-Kutta scheme, we have the following collocation equations on this step, for \( \ell = 1, \ldots, r \):

\[
0 = f_{\ell,n} (y_{\ell,n,1}^p, \ldots, y_{\ell,n,r}^p, z_{\ell,n,1}^p | x^p_{\ell,n,0}, \sigma_{\ell}, \zeta),
\]

\[
\triangleq f \left( x^p_{\ell,n,0} + c_i h_i y_{\ell,n,i}^p, x^p_{\ell,n,0} + h \sum_{j=1}^{r} a_{ij} y_{\ell,n,j}^p, \sigma_{\ell}, \zeta \right),
\]

where \( c_i \in \mathbb{R} \) and \( a_{ij} \in \mathbb{R} \) are coefficients in Butcher table [12]; \( y_{\ell,n,1}^p \in \mathbb{X} \) and \( x_{\ell,n,1}^p \in \mathbb{R} \) denote, respectively, the \( i \)th stage values of the derivative of differential state and the algebraic state.

For convenience, let

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

be the stage variable and \( \xi^T_n \triangleq \left[ (x_{\ell,n,1}^p)T, (\sigma_{\ell})T, (\zeta)T \right]T \). Given \( \xi_n \) and an initial guess \( k_n^{(0)} \), \( k_n \), satisfying equation

\[
0 = F_n^p \left( k_n^{(0)} | \xi_n \right)
\]

\[
\triangleq \begin{bmatrix}
    f_{n,1} (y_{n,1}^p, \ldots, y_{n,r}^p, z_{n,1}^p | k_n^{(0)} \\
    \vdots \\
    f_{n,r} (y_{n,1}^p, \ldots, y_{n,r}^p, z_{n,r}^p | k_n^{(0)})
\end{bmatrix},
\]

can be computed by Newton iterations:

\[
k_n^{(i)} = k_n^{(i-1)} - (M_n^{(i-1)})^{-1} F_n^p (k_n^{(i-1)} | \xi_n),
\]

\[
i = 1, \ldots, L, (9a)
\]
where \(1 \leq L \in \mathbb{Z}\) is a fixed iteration number and
\[
M_{n}[i-1] = \frac{\partial F_n^T}{\partial k_n} \left( k_n \xi_n^T \right) \bigg|_{k_n = k_n[i-1]}.
\]
(9b)

Then, by quadrature, we have the approximation to the differential state at \(t = \tau_{n+1,0}\).
\[
x^\ell_{n+1,0} = x^\ell_{n,0} + h \sum_{j=1}^{r} b_j y^\ell_{n,j},
\]
(10)
where \(b_j\) is the coefficient in Butcher table [12].

In sequential case, \(x^\ell_{n,0}\) is not an independent variable for optimization as in multiple shooting case. We can derive \(x^\ell_{0,0}\) from \(x^\ell_{0,0}\) by (10) recursively with \(n \) from 0 to \(q-1\). Then, with initialization \(x_{0,0} = x_0(\zeta), x^\ell_{q,0}\) can be computed by executing this \(q\)–step recursion repeatedly, for \(\ell = 1, \ldots, p\), and following continuous conditions \(x^\ell_{0,0} = x^\ell_{q,0}\), for \(\ell = 1, \ldots, p-1\). As a result, the terminal state can be approximated by \(x^\ell(T) = x^\ell_{q,0}\). The objective and constraint functions \(g^\ell_m(\sigma, \zeta)\), 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, \(k_n^\ell\) is a function of \(\xi_n^\ell\) from (8). Then, we can obtain from the implicit function theorem that
\[
\frac{\partial k_n^\ell}{\partial \xi_n^\ell} = -\left( M_n^\ell \right)^{-1} \frac{\partial F_n^T}{\partial \xi_n^\ell}.
\]
(11)

Clearly, from (10) and (11), we have
\[
\frac{\partial x_{n+1,0}^\ell}{\partial x_{n}^\ell} = h \sum_{j=1}^{r} b_j \frac{\partial y^\ell_{n,j}}{\partial x_{n}^\ell} + \left[ I_{n_c} \quad 0 \quad 0 \right],
\]
(12)
where \(I_{n_c} \in \mathbb{R}^{n_c \times n_c}\) is an identity matrix. Then, from chain rule and the forward propagation formula of algorithmic differentiation, gradient of the objective or constraint function with respective to \((\sigma, \zeta)\) can be derived [3]. Note that a prerequisite of applying (11) is that the Newton iterations (9) converge. Decomposition of \(M_n^\ell\) 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 \(G \in \mathbb{R}^{n_x \times (n_x+n_c)}\), for \(1 \leq n_d \leq 1 + n_c\), be a directional matrix, and \(x^\ell_{n,0} \in \mathbb{R}^{n_x \times n_c}\) be the corresponding adjoint variable matrix. Similar to [6], we introduce the following adjoint equation,
\[
x^\ell_{n,0} = x^\ell_{n+1,0} - \frac{\partial x^\ell_{n+1,0}}{\partial x^\ell_{n,0}},
\]
(13a)
for \(\ell = p, \ldots, 1\) and \(n = q-1, \ldots, 0\), with terminal condition
\[
x^\ell_{q,0} = \frac{\partial G_{[0]} g^p_{[0]} \cdots g^p_{[n_c]} T}{\partial x^\ell_{q,0}},
\]
(13b)
and continuous condition
\[
x^\ell_{q-1} = x^\ell_{0,0}, \quad \ell \neq 1,
\]
(13c)
where \(\frac{\partial x^\ell_{n+1,0}}{\partial x^\ell_{n,0}}\) is computed from (12).

**Proposition 1** Given initial condition \(x^p(0) = x_0(\zeta)\), and parameter \(\omega \triangleq [\sigma_1^T, \ldots, \sigma_p^T, \zeta]^T\). If the differential state \(x^\ell(T)\) of DAE (5) is approximated by \(x^\ell_{n,0}\), which is computed from the IRK scheme (8)-(10) with Newton iterations (9) converged, then, for a given \(G \in \mathbb{R}^{n_x \times (n_x+n_c)}\) the directional derivative of the objective and constraint function vector \([g^p, \ldots, g^p_{n_c}]^T\) with respect to \(\omega\) can be approximated by
\[
\frac{dG_{[0]} g^p \cdots g^p_{n_c} T}{d\omega} \approx \omega,
\]
(14a)
where \(\omega = [\sigma_1, \ldots, \sigma_p, \zeta]\) can be derived by
\[
\left\{ \begin{array}{l}
\sigma_\ell = \sum_{n=0}^{q-1} x^\ell_{n+1,0} \frac{\partial x^\ell_{n+1,0}}{\partial \sigma_\ell}, \quad \ell = p, \ldots, 1, \\
\zeta = x^\ell_{0,0} \frac{\partial x^\ell_{0,0}}{\partial \zeta} + \sum_{n=0}^{q-1} \sum_{n=q}^{p-1} x^\ell_{n+1,0} \frac{\partial x^\ell_{n+1,0}}{\partial \zeta} + \frac{\partial G_{[0]} g^p_{[0]} \cdots g^p_{[n_c]} T}{\partial \zeta}
\end{array} \right.
\]
(14b)

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

**Proof.** Take \(\sigma_\ell, \ell = p, \ldots, 1\), for example. Following the tradition of backward algorithmic differentiation [13],
\[
x^\ell_{n+1,0} = \frac{dG_{[0]} g^p_{[0]} \cdots g^p_{[n_c]} T}{d\sigma_{\ell}} \quad \text{for} \quad n = q-1, \ldots, 0.
\]
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 \(\sigma_0 = 0\) initially. Then, \(\sigma_\ell\) can be computed by \(\sigma_\ell^+ = x^\ell_{n+1,0} \frac{\partial x^\ell_{n+1,0}}{\partial \sigma_{\ell}}\) with \(n = q-1, \ldots, 0\). Here, \(\delta a + b = b\) denotes assigning \(a + b = a\). 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 \(\omega\). If \(\omega\) does not change too 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 \(\Delta x^\ell_{n,0} \triangleq x^\ell_{n,0} - x_{n,0}^\ell\), where \(\zeta^*\) denotes cached value of \(\zeta\), which is obtained from the last optimization iteration. In analogy to the lifted implicit integrator proposed in [5], initial estimate of \(k^\ell_{0,0}\) in
(9) can be updated by the following tangential prediction,
\[ k^f[n] = k^f + \frac{\partial k^f}{\partial x^f} \Delta x + \frac{\partial k^f}{\partial \omega} \Delta \omega + \frac{\partial k^f}{\partial y^f} \Delta y + \frac{\partial k^f}{\partial x^f} \Delta x^f. \tag{15} \]
where \( k^f, \frac{\partial k^f}{\partial x^f} \) and \( \frac{\partial k^f}{\partial \omega} \) are evaluated at \( x^f \). Unlike in multiple shooting methods, \( x^f \) here cannot be obtained directly. However, since \( \Delta x^f = 0 \), \( x^f \) with respect to current \( \omega \) can be determined step-by-step from the initial condition (5b) by integration. After that, \( k^f \) can be corrected in further by a few Newton iterations (one Newton iteration in the limit).

In many NLP algorithms, \( \omega \) 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 feasibility restoration [11].

**Remark 1**
The correction procedure in Algorithm 2 is with respect to prediction \( k^f \). 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 \( k^f \). It was proved in [5] that \( \Delta \omega \) with \( \Delta k^f \) for \( \ell = 1, \ldots, p \) and \( n = 0, \ldots, q - 1 \), 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].

### 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 \( \frac{2}{3} n^3 + 2n^2 \) flops, where \( n_k = \sqrt{r(n_x + n_z)} \). The dimension of stage variable \( k^f \). As the first two steps for sensitivity propagation, computations in (11) and (12) require, respectively, \( 2n_k n_x \) and \( 2n_k n_z \) flops.
flops, where \( n_g \triangleq n_x + n_u + n_c \). The backward propagation of adjoint variables in (13) requires approximately \( 2pqn_x^2(n_c + 1) \) flops. If \( n_c + 1 < n_u \), where \( n_u \triangleq n_m + n_c \) represents the number of optimization variables, this propagation procedure is more efficient than forward sensitivity propagation, where \( 2pqn_x^2n_c \) flops are needed approximately. For Scheme II, additional \( 3pqn_u n_c \) 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 \( x_{n,0}^T(n_c + 1)(n_x + 1) \) to store all the adjoint variables, \( n_k(n_k + n_c) \) to store intermediate variables and \( pqn_k \) to store \( k_d^T \). Besides this, \( pqn_u n_c \) memory is needed to store all the \( \frac{\partial x^0}{\partial a} \). For Scheme II, additional memory \( n_x + n_u + pqn_k + pqn_x n_c \) is used to store \( \Delta x_{n,0}^T, \omega^T, x_{n,0}^T \) and \( \omega^T \). Clearly, Scheme II requires more memory.

### 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) 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 \( l_a = 0.2m \), and three forearms of length \( l_f = 0.6m \). The mass \( m_r = 5 \times 10^{-2} \)kg and inertia \( J_r = 0.1\text{kgm}^2 \) of the robot are concentrated in its nacelle and motor-arms, respectively. Let \( p \triangleq [p_1, p_2, p_3]^T \in \mathbb{R}^3 \) denote the position of the nacelle, \( \vartheta \triangleq [\vartheta_1, \vartheta_2, \vartheta_3]^T \in \mathbb{R}^3 \) be the angular position of arms, and \( (\alpha_1, \alpha_2, \alpha_3) = (0, \frac{2\pi}{3}, \frac{2\pi}{3}) \) denote the triple of mounting angles of arms. To ensure that each length of the forearms is invariant during moving of nacelle, the geometry of the robot is constrained by

\[
|p - R_i a_i|^2 - l_f^2 = 0, \quad i = 1, 2, 3,
\]

where \( R_i \) is the rotation matrix,

\[
|\cos \alpha_i - \sin \alpha_i 
\sin \alpha_i & \cos \alpha_i 
0 & 0 & 1 \rangle \quad \text{and} \quad a_i = l_a \frac{\cos \vartheta_i}{0 \sin \vartheta_i}.
\]

The robot has kinetic energy \( K = \frac{1}{2} m_r \dot{p}^T \dot{p} + \frac{1}{2} J_r \dot{\vartheta}^T \dot{\vartheta} + \dot{p}^T f_e \) and potential energy \( V = m_{cg} g z + T^T \vartheta \), where \( z \in \mathbb{R}^3 \) is the Lagrange multiplier, \( f_e \triangleq [f_{x_1}, f_{x_2}, f_{x_3}]^T \) is the equality constraint (16) and \( T \triangleq [T_1, T_2, T_3]^T \in \mathbb{R}^3 \) is the motor torque. Denote \( q \triangleq [\vartheta^T, p^T]^T \) 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:

\[
\begin{bmatrix}
M_T & -\frac{\partial \xi^T}{\partial q} \\
-\frac{\partial \xi}{\partial q} & 0_{3 \times 3}
\end{bmatrix}
\begin{bmatrix}
\dot{q}
\dot{\xi}
\end{bmatrix}
= \begin{bmatrix}
\rho \frac{\partial \xi}{\partial q} \\
\rho \frac{\partial \xi}{\partial q}
\end{bmatrix}
\begin{bmatrix}
\dot{q}
\end{bmatrix},
\]

where \( M_T \triangleq \text{diag}([J_r, J_f, J_f, m_r, m_r, m_r]) \). 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 \( T \) is bounded, which satisfies

\[
-5\text{Nm} \leq T_i \leq 5\text{Nm}, \quad i = 1, 2, 3.
\]

The control objective is to force the nacelle moving from the initial position \( p_0 = [0, 0, -0.6] \) (with zero velocity and acceleration) to terminal position \( p_f = [0.15, 0.15, -0.7] \) in \( T = 0.225 \) second consuming little energy. Specifically, the cost functional is

\[
g_0 = \int_0^T \{(1 - \rho) ||p(t) - p_f||^2 + \rho ||T(t)||^2 \} dt,
\]

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 (\( z \)) in all. The nacelle is required to reach position \( p_f \) at terminal time. Thus, there is only one constraint

\[
p(T) - p_f = 0
\]
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 \( p_0 \) to \( p_f \) 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-dotted 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.
5 Conclusions

In this paper, function evaluation and gradient computation are investigated for the solution of optimal control problems 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 an 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.

References


