An Iterative Method for Final Time Optimization in Nonlinear Optimal Control

Alberto De Marchi* Matthias Gerdts*

Abstract

This paper discusses a bilevel optimization approach for free finite final time optimal control problems and addresses a numerical method for their approximate solution. The core idea is to decouple the final time optimization from the optimal control and state trajectory. This is rigorously formulated as an equivalent bilevel problem seeking, at the upper level, the optimal final time and optimal control and corresponding state at the lower level. Standard solvers for nonlinear optimal control can deal with the latter, while the former is a box-constrained optimization problem with one scalar decision variable. The interface between the two levels is based on the Hamilton function associated to the problem and its relationship with the cost function. A method for solving the upper level problem is developed, that combines a tailored fast first-order method with a robust and guaranteed root-finding algorithm. Finally, numerical results demonstrate the robustness of the method and show its limitations.

1 Introduction.

This paper aims at solving nonlinear optimal control problems (OCPs) with free final time by decoupling the search of the optimal control from the final time optimization. This approach has been introduced in [11] for linear systems and extended to nonlinear OCPs in [12], and its bilevel perspective is based on the vanishing condition on the Hamilton function for free time OCPs [13], [18]. This allows to formulate a single-objective bilevel optimization problem, consisting of an upper and a lower level. The latter seeks the optimal control for an OCP with fixed final time, which is considered at and given by the upper level. This approach avoids any time or spatial transformation, which are commonly adopted for this class of problems, especially those arising, e.g., in automotive and robotic applications [3], [16], [19], [20]. Hence, the dynamics do not change and no local optima are introduced by the transformation. However, the equivalence of the bilevel reformulation is in general not guaranteed, unless the lower level problem admits a unique solution [8], [9].

The contribution of this paper is an extension of the bilevel approach to any method for solving the lower level problem, that is the fixed final time counterpart of the original problem. In this respect, this work differs from [12], which is based on successive linear-quadratic approximations of the lower level nonlinear problem and exploits the results in [11]. Herein, instead, the only necessary feature, needed by the upper level, is that it must be possible to evaluate the Hamilton function, along a solution given by the lower level, at the final time (for autonomous systems, at some time). This value corresponds to the sensitivity of the cost function with respect to the final time [11], and thus, at the upper level, it can be used within a gradient-based optimization method. This approach harks back to the descent methods discussed in [8]. Indeed, the upper level does not necessarily need the evaluation of the cost function, even though it could take advantage of it, as discussed in §3.2. Since the lower level is an optimization problem, it may share inexact or inaccurate information with the upper level, which could still properly work; for details see [10].

This paper is organized as follows. In §2 we formulate a bilevel optimization approach for free-time OCPs, which we propose to overcome difficulties with local minima that might be introduced by standard time transformation techniques for problems with free final time. The solution approach is briefly delineated. A method for dealing with the upper level problem is sketched in §3, with the corresponding procedures and the coupling with the lower level is discussed. The proposed approach is numerically validated on two standard problems in §4, showing effectiveness and limitations of the proposed algorithm.

2 Problem and Approach.

Let

$$\begin{aligned} J &: \mathbb{R}^{n_x} \times \mathbb{R}^{n_x} \times \mathbb{R}^{n_p} \times \mathbb{R} \to \mathbb{R} \\ \boldsymbol{f} &: \mathbb{R}^{n_x} \times \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \times \mathbb{R}^{n_p} \times \mathbb{R} \to \mathbb{R}^{n_x} \\ \boldsymbol{b} &: \mathbb{R}^{n_x} \times \mathbb{R}^{n_x} \times \mathbb{R}^{n_p} \times \mathbb{R} \to \mathbb{R}^{n_b} \\ \boldsymbol{c} &: \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \times \mathbb{R}^{n_p} \times \mathbb{R} \to \mathbb{R}^{n_c} \end{aligned}$$

be sufficiently smooth functions (n_x, n_u, n_p, n_b, n_c being some positive integers). Let also T_ℓ, $T_u \in \mathbb{R}$ be given bounds on the final time, such that $0 < T_\ell < T_u$.

*Bundeswehr University Munich, 85577 Neubiberg, Germany. (e-mail: {alberto.demarchi, matthias.gerdts}@unibw.de)

Let us consider the following free-time optimal control problem:

PROBLEM 2.1. (OOCP) *Minimize* $J(\boldsymbol{x}(0), \boldsymbol{x}(T), \boldsymbol{p}, T)$ *subject to the constraints*

$$\boldsymbol{f}(\dot{\boldsymbol{x}}(t), \boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{p}, T) = \boldsymbol{0} \ , \qquad t \in [0, T] \ , \tag{2.1}$$

$$\boldsymbol{b}(\boldsymbol{x}(0), \boldsymbol{x}(T), \boldsymbol{p}, T) \leq \boldsymbol{0} \ , \tag{2.2}$$

$$\boldsymbol{c}(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{p}, T) \leq \boldsymbol{0} \ , \qquad t \in [0, T] \ , \tag{2.3}$$

$$T_\ell \leq T \leq T_u \ . \tag{2.4}$$

Herein, the controls $\boldsymbol{u}(\cdot)$ are assumed to be measurable, with the corresponding state trajectories $\boldsymbol{x}(\cdot)$ being absolutely continuous functions of time (in general, for DAEs, not all components of $\boldsymbol{x}(\cdot)$ can be absolutely continuous [13]). Problem 2.1 represents a fairly general class of problems, in that it possibly has dynamics described by differential-algebraic equations (2.1), free boundary conditions (2.2), mixed state and control constraints (2.3), free parameters $\boldsymbol{p}$ and final time T. Considering autonomous OCPs only is not a restriction [13], [15]; for an extension in this direction see [12]. For the sake of simplicity and without loss of generality, let us collect in $\boldsymbol{\lambda}$ the Lagrange multipliers associated to the OOCP and denote $H = H(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{p}, \boldsymbol{\lambda}, T)$ the corresponding Hamilton function [13], [18].

2.1 Lower Level. Assume a final time $\hat{T} \in [T_\ell, T_u]$ is given by the upper level. Then, a fixed time OCP can be formulated; the lower level problem (LLP) reads:

PROBLEM 2.2. (LLP) *Minimize* $J(\boldsymbol{x}(0), \boldsymbol{x}(T), \boldsymbol{p}, T)$ *subject to the constraints* (2.1)–(2.3) *and* $T = \hat{T}$.

This problem is to be solved, with different values of $\hat{T}$, at each and every iteration of the upper level. One could take advantage of previous solutions as initial guesses for warm starting the lower level optimization.

2.2 Upper Level. For a given final time $\hat{T}$, let us denote $\hat{\boldsymbol{x}}$, $\hat{\boldsymbol{u}}$, $\hat{\boldsymbol{p}}$ and $\hat{\boldsymbol{\lambda}}$ a solution to the corresponding LLP. Then, let us introduce the reduced cost function $\tilde{J}$ and the reduced Hamilton function $\tilde{H}$, defined by:

$$\tilde{J}(\hat{T}) := J(\hat{\boldsymbol{x}}(0), \hat{\boldsymbol{x}}(\hat{T}), \hat{\boldsymbol{p}}, \hat{T}) \ , \tag{2.5}$$

$$\tilde{H}(\hat{T}) := H(\hat{\boldsymbol{x}}(\hat{T}), \hat{\boldsymbol{u}}(\hat{T}), \hat{\boldsymbol{p}}, \hat{\boldsymbol{\lambda}}(\hat{T}), \hat{T}) \ , \tag{2.6}$$

for any $\hat{T} \in [T_\ell, T_u]$. In the case the LLP turns out to be an autonomous OCP, the Hamilton function attains a constant value along a solution and definition (2.6) could be relaxed [11], [13]. By constructing the reduced cost function (2.5), the underlying OCP is hidden and an optimization problem with a single decision variable, namely the final time, appears; it reads:

PROBLEM 2.3. (ULP) *Minimize* $\tilde{J}(T)$ *subject to the constraint* (2.4).

We stress that the bilevel optimization problem consisting of LLP and ULP, namely Problems 2.2 and 2.3, is in general not equivalent to Problem 2.1; if the lower level problem admits a unique solution, then the two formulations are equivalent [8], [9]. In fact, in our case, it suffices to solve the lower level problem to global optimality, exploiting the fact that lower and upper levels share the same objective function. Then, in particular, a solution to the ULP is also an optimal final time for the OOCP. The key result which supports the bilevel perspective is drawn in [11, Theorem 10]; it reads:

$$\tilde{J}'(T) = \tilde{H}(T) \ , \quad \forall T \in [T_\ell, T_u] \ . \tag{2.7}$$

This provides a relatively simple way to couple the lower to the upper level and adopt gradient-based techniques for facing the ULP. In §3 we propose a method to solve the ULP and sketch a suitable procedure for the bilevel solver.

3 Methods.

This Section discusses an approach to couple the two levels of the bilevel optimization problem formulated above and proposes a method for solving the upper level problem. For the bilevel approach introduced above to properly work, some information must be shared among the levels. The coupling between the two levels is based on the reduced quantities (2.5)–(2.6) and their relationship (2.7). Hence, the method proceeds as follows: (i) the upper level communicates an estimate of the optimal final time, say $\hat{T}$, to the lower level; (ii) the corresponding LLP is solved and the reduced quantities, say $\hat{J} := \tilde{J}(\hat{T})$ and $\hat{H} := \tilde{H}(\hat{T})$, are evaluated and sent back to the upper level, at which (iii) the optimal final time estimate is updated. Overall, at most three scalar values are exchanged at each and every iteration of the bilevel solver; thus, the communication load is clearly marginal. Indeed, the cost function evaluation is not strictly needed for the upper level to work, as considered in §3.2 from a general perspective. Suitable features for the lower level are summarized in §3.1. Regarding the implementation of the proposed methods, the reverse communication mode provides a favorable environment for setting up the bilevel solver and interfacing the two levels.

3.1 Lower Level. It is assumed a standard solver, e.g. CasADi [1], BOCOP [4] and OCPID-DAE1 [14], is available and capable of solving the LLP (global convergence to local optima is expected). Moreover, one must be able to evaluate the Hamilton function H,

possibly *a posteriori*, along a solution to the LLP. These two requirements are to evaluate the reduced quantities defined in (2.5)–(2.6) and to be exchanged with the upper level.

3.2 Upper Level. Let a nonempty closed interval $\mathbb{X}$, a function $g : \mathbb{X} \to \mathbb{R}$ and an initial guess $x_s \in \mathbb{X}$ be given. For clarity, consider $\mathbb{X} := [x_\ell, x_u]$. Function g satisfies $g(x) = f'(x)$, $x \in \mathbb{X}$, for a given unknown function $f : \mathbb{X} \to \mathbb{R}$. We assume function g be sufficiently smooth in $\mathbb{X}$ and unknown function f be convex in $\mathbb{X}$. Even though these conditions guarantee global convergence to a global optimum, they can hardly be satisfied in practice. In fact, Lipschitz continuity of g might be restrictive too and convexity of f might only hold locally around a solution. Let us consider the following box-constrained scalar optimization problem:

PROBLEM 3.1. *Minimize $f(x)$, by only evaluating g, subject to the constraint $x \in \mathbb{X}$.*

In order to solve this problem, we propose to combine a first-order optimization method and a root-finding algorithm. In particular, the former is tailored for the specific problem based on the fast proximal gradient method [2], see Algorithm 1. The iterative procedure begins with an initial guess $x_s \in \mathbb{X}$ and achieves optimal rate of convergence with marginal overhead computation with respect to other first-order methods [2], [17]. It adjusts the stepsize based on an estimate L_g of the Lipschitz constant of g in $\mathbb{X}$ and on the sequence of gradient evaluations, through a safety factor $\kappa \geq 1$. This is because the former is not always known or not easily computable. Also, in order to generate a sequence of estimates in the domain of g, an additional projection is performed, compare [2]. We highlight that the proximal operator reduces to a simple projection, because the feasible set $\mathbb{X}$ is a closed nonempty interval [17]. Termination criteria for Algorithm 1 may comprise a maximal number of iterations and check $|x_{i+1} - x_i| < \delta_x$ and $|g_i| < \delta_g$, δ_x and δ_g being small positive tolerances. Evaluating function f in Algorithm 1 would make it possible to adopt a backtracking stepsize rule or to have a monotone algorithm, improving robustness in both cases [2]. Finally, we point out that the evaluation of g might be inexact, since it results from an optimization problem, namely the LLP. Depending on this oracle's accuracy, it may be preferable to adopt a classical gradient method instead of a fast one, because the latter suffers from error accumulation [10].

Starting from an initial guess, Algorithm 1 generates a sequence of pairs (x_k, g_k), $k = 1, 2, \ldots$, where $x_k \in \mathbb{X}$ and $g_k = g(x_k)$. In general, assuming function g be continuous in an interval $\mathbb{Z} := [x_\ell^0, x_u^0] \subseteq \mathbb{X}$,

Algorithm 1 `FPG`: a Fast Projected Gradient method.

procedure FPG(g, x_s, $\mathbb{X}$, L_g, κ)
  $x_1 \leftarrow x_s$, $L_0 \leftarrow L_g$, $\alpha_1 \leftarrow 1$, $z_1 \leftarrow x_s$
  for $j = 1, 2, \ldots$ **do**
    $g_j \leftarrow g(x_j)$
    $L_j \leftarrow \max\left(L_{j-1}, \kappa|g_j|\right)$ ▷ stepsize
    $z_{j+1} \leftarrow \mathrm{proj}_{\mathbb{X}}\left(x_j - (g_j/L_j)\right)$ ▷ prox-grad
    $\alpha_{j+1} \leftarrow \left(1 + \sqrt{1 + 4\alpha_j^2}\right)/2$
    $y_{j+1} \leftarrow z_{j+1} + (z_{j+1} - z_j)(\alpha_j - 1)/\alpha_{j+1}$
    $x_{j+1} \leftarrow \mathrm{proj}_{\mathbb{X}}\left(y_{j+1}\right)$ ▷ projection
  end for
  return x_{j+1}
end procedure

Algorithm 2 `posZero`: a positive-slope zero-crossing detector.

procedure POSZERO($\{x_i, g_i\}_{i=1}^n$)
  $\mathcal{I} \leftarrow \left\{ (a,b) \;\middle|\; \begin{array}{l} a, b = 1, 2, \ldots, n\ , \\ x_a < x_b\ , \\ g_a < 0 < g_b \end{array} \right\}$
  $\mathbb{Z} \leftarrow \bigcup_{(a,b)\in\mathcal{I}} [x_a, x_b]$
  return $\mathbb{Z}$
end procedure

root-finding algorithms guarantee to find an $x^\star \in \mathbb{Z}$ such that $g(x^\star) = 0$ if $\mathbb{Z}$ is a change-of-sign interval for g, that is, if $g(x_\ell^0)\ g(x_u^0) < 0$. Therefore, as a precondition for executing the root-finding algorithm, one has to detect a suitable change-of-sign interval $\mathbb{Z}$, possibly based on the sequence $\{(x_k, g_k)\}$ only. The procedure reported in Algorithm 2 generates a set which is the union of those change-of-sign intervals for g with positive mean slope. This restriction stems from the sufficient optimality condition for Problem 3.1, namely $f''(x^\star) = g'(x^\star) > 0$, valid for sufficiently smooth problems. As soon as the set $\mathbb{Z}$ contains an interval, the root-finding method can be executed; this provides a robust and guaranteed way for solving the first-order necessary optimality conditions, namely $g(x^\star) = 0$. Nonetheless, it is not guaranteed to end up in a local minimum, since local maxima may be in $\mathbb{Z}$ as well. In the following we consider and adopt the Brent's method, which is a derivative-free root-finding method and usually exhibits superlinear convergence [5]; it combines linear interpolation and inverse quadratic interpolation with bisection. A publicly available implementation using reverse communication has been adopted [7].

The overall strategy for the upper level is sketched in Algorithm 3. First of all, it is checked if $\mathbb{X}$ is a positive-slope zero-crossing interval, then also the initial

Algorithm 3 Upper level.

procedure UPPERLEVEL(g, x_s, $\mathbb{X}$, L_g, κ, δ_x, δ_h)
  $x_\ell,\ x_u \leftarrow \mathbb{X}$
  $i \leftarrow 0$, $x_i \leftarrow x_\ell$, $g_i \leftarrow g(x_\ell)$ ▷ lower bound
  $i \leftarrow 1$, $x_i \leftarrow x_u$, $g_i \leftarrow g(x_u)$ ▷ upper bound
  $\mathbb{Z} \leftarrow \texttt{posZero}(\{(x_k, g_k)\}_{k=0}^{i})$
  if $\mathbb{Z} = \emptyset$ **then**
    $i \leftarrow 2$, $x_i \leftarrow x_s$, $h_i \leftarrow g(x_s)$ ▷ guess
    $\mathbb{Z} \leftarrow \texttt{posZero}(\{(x_k, g_k)\}_{k=0}^{i})$
  end if
  while $\mathbb{Z} = \emptyset$ **do**
    $x_{i+1} \leftarrow \texttt{FPG}(g, x_s, \mathbb{X}, L_g, \kappa)$ ▷ FPG
    $g_{i+1} \leftarrow g(x_{i+1})$
    if $|x_{i+1} - x_i| < \delta_x$ **or** $|g_{i+1}| < \delta_h$ **then**
      $x^\star \leftarrow x_{i+1}$
      return $x^\star$
    end if
    $i \leftarrow i + 1$
    $\mathbb{Z} \leftarrow \texttt{posZero}(\{(x_k, g_k)\}_{k=0}^{i})$
  end while
  $x^\star \leftarrow \texttt{Brent}(g, \mathbb{Z}, \delta_x, \delta_h)$ ▷ Brent's method
  return $x^\star$
end procedure

guess x_s is considered. As long as no suitable change-of-sign interval is detected, namely $\mathbb{Z} = \emptyset$, the tailored first-order method in Algorithm 1 is executed and the sequence $\{(x_k, g_k)\}$ extended. At each and every of these iterations, Algorithm 2 is called to detect and, if possible, build $\mathbb{Z}$; this ensures the set $\mathbb{Z}$ is either empty or a nonempty interval in $\mathbb{X}$. Some cases are possible. (i) If f has a minimum in $\mathbb{X}$, then g exhibits a positive-slope zero-crossing, hence $\mathbb{Z}$ is nonempty and $x^\star \in \mathbb{Z}$ is found by the root-finding algorithm. (ii) If f has a maximum in $\mathbb{X}$, then g exhibits a negative-slope zero-crossing, hence $\mathbb{Z}$ is empty and the projected gradient method terminates at $x^\star \in \{x_\ell, x_u\}$, depending on the initial guess x_s. (iii) If f is monotone increasing (decreasing) in $\mathbb{X}$, then g is positive (negative), $\mathbb{Z}$ is empty and the projected gradient method terminates at $x^\star = x_\ell$ (x_u). These cases are investigated in the next Section with numerical examples on two standard problems.

4 Numerical Results.

The methods proposed in §3 are tested with two examples on different scenarios. The aim is to validate the approach, show its effectiveness and disclose its limitations and drawbacks. The two case problems involve the point-to-point motion of either a vehicle or a trolley with a load, with a cost functional based on both the control effort and the final time. These OCPs can easily be casted into the form of Problem 2.1. In particular, an additional differential state is introduced to express the cost in Mayer form [13].

Algorithm 3 is adopted with $L_g = \delta_x = \delta_h = 10^{-6}$ (SI units are omitted). The influence of parameter κ is discussed in §4.1. In view of the bilevel approach, inputs g, x_s and $\mathbb{X}$ are nothing but the reduced Hamilton function $\tilde{H}$, Eq. (2.6), an initial guess T_s of the optimal final time and a set $[T_\ell, T_u]$ of feasible final times, Eq. (2.4). Each evaluation of function g involves solving the corresponding LLP, formulated in §2.1, and evaluating the associated Hamiltonian. The LLPs are solved by the software package OCPID-DAE1 [14], with a direct single shooting approach, on a discretization of equidistant grid points, with piecewise linear control approximation; optimality and feasibility tolerance are set to 10^{-8} and 10^{-10} respectively; gradient information is obtained through the sensitivity DAE.

4.1 Vehicle. Let us consider the longitudinal motion of a vehicle, along a straight horizontal line, whose dynamics read

$$\dot{x}_1 = x_2 \ , \tag{4.8}$$

$$\dot{x}_2 = u - c_1 x_2 - c_2 x_2^2 \ , \tag{4.9}$$

with $c_1 = c_2 = 0.01$. Herein, states x_1 and x_2 and control u denote the vehicle position and velocity and the control thrust, respectively. We aim at minimizing the cost functional

$$J(u, T) = \int_0^T w_u u^2(t) \mathrm{d}t + T \ , \tag{4.10}$$

with $w_u = 0.61803399$[1], while satisfying boundary conditions $x_1(0) = x_2(0) = x_2(T) = 0$, $x_1(T) = 100$, and control bounds $u(t) \in [-1, 1]$, $t \in [0, T]$. For the LLPs, a discretization of 101 equidistant grid points and a fixed-step fourth-order Runge-Kutta integration method are adopted. The reduced cost and Hamiltonian (2.5)–(2.6) for this example are depicted in Fig. 1, for $T \in [21, 40]$. Notice that without drag forces, namely $c_1 = c_2 = 0$, the minimum feasible time to satisfy both the boundary conditions and the control bounds is $T = 20$. The reduced cost $\tilde{J}$ exhibits a minimum, around $T \approx 23$, and the reduced Hamiltonian $\tilde{H}$ crosses zero with positive slope, accordingly. In a sense, this is the problem from the upper level perspective, see §3.2. Condition (2.7) is numerically tested by means of a finite difference approximation of $\tilde{H}$ based on $\tilde{J}$; it matches

[1] This is approximately the reciprocal of the golden section. Life is too short for boring numbers.

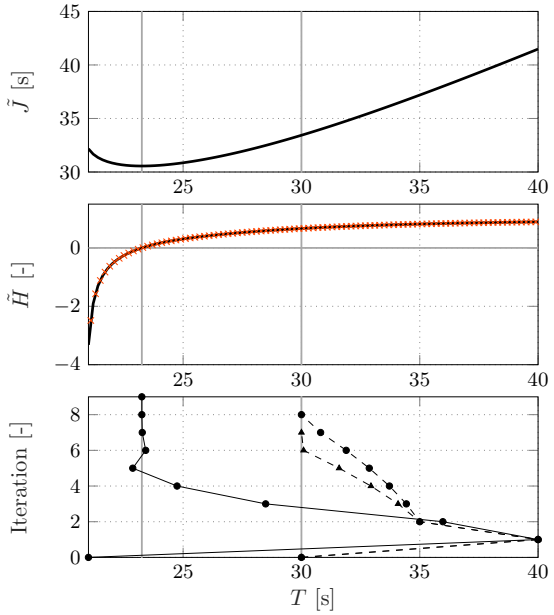

Figure 1: Vehicle — Reduced cost $\tilde{J}$, reduced Hamiltonian $\tilde{H}$ with finite difference approximation (cross), and final time T during iterations, with $\kappa = 1$ (triangle) and $\kappa = (1+\sqrt{5})/2$ (circle), for $T \in [21, 40]$ (solid) and $T \in [30, 40]$ (dashed).

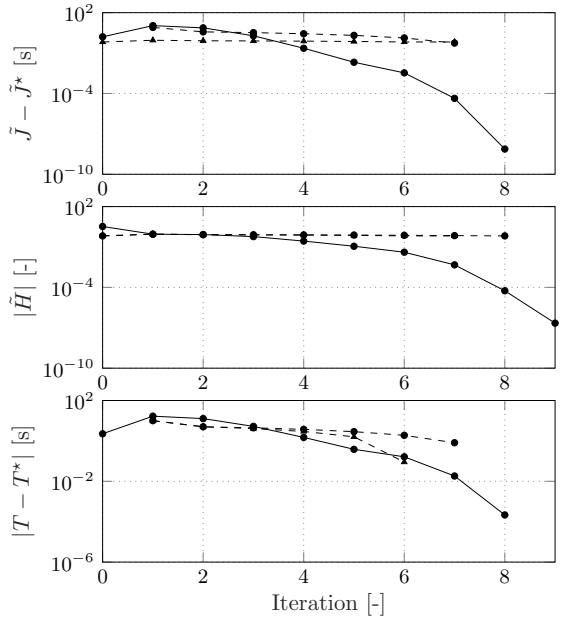

Figure 2: Vehicle — Reduced cost $\tilde{J}$, reduced Hamiltonian $\tilde{H}$ and final time T during iterations, with $\kappa = (1+\sqrt{5})/2$ (circle) and $\kappa = 1$ (triangle), for $T \in [21, 40]$ (solid) and $T \in [30, 40]$ (dashed).

very well, see Fig. 1. Also, the sequence of final time T during iterations are reported in different scenarios. For $T \in [21, 40]$, the Brent's method is immediately called and converges to the (unconstrained) minimum $T^\star = 23.2554$. Instead, for $T \in [30, 40]$, the reduced Hamiltonian is always positive and thus only the tailored gradient method is executed and converges to the (constrained) minimum $T^\star = 30$. In this case, lower values of κ show faster convergence but may be less robust, while higher values may be more conservative, see Fig. 1 (bottom) and Fig. 2.

For the sake of comparison, the original free final time problem was also solved with OCPID-DAE1 [14] through time transformation. We are interested in the number of iterations it takes to solve the problem. The results are reported in Table 1. Starting from different initial guesses, it ends up with the optimal solution in all cases, running through 32–37 iterations. Setting the guess as the fixed final time, it takes a comparable number of iterations. Hence, for this simple problem, the bilevel approach requires in total around 6 times more iterations (for the OCP solver) and returns the same solution. Then, in this case there is no clear benefit.

4.2 Trolley–load. Let us consider a trolley moving on an horizontal straight line, with a load mass and a massless rod. This system can be modelled by an index-three DAE [6, Example 10]; in the following, computations are performed considering the system with Gear-Gupta-Leimkuhler (GGL) stabilization. The dynamics read

$$\dot{x}_1 = x_4 - 2x_8(x_1 - x_2) \,, \tag{4.11}$$
$$\dot{x}_2 = x_5 + 2x_8(x_1 - x_2) \,, \tag{4.12}$$
$$\dot{x}_3 = x_6 - 2x_8 x_3 \,, \tag{4.13}$$
$$\dot{x}_4 = [u - 2x_7(x_1 - x_2)]/m_1 \,, \tag{4.14}$$
$$\dot{x}_5 = 2x_7(x_1 - x_2)/m_2 \,, \tag{4.15}$$
$$\dot{x}_6 = -g - 2x_7 x_3/m_2 \,, \tag{4.16}$$
$$0 = (x_1 - x_2)^2 + x_3^2 - \ell^2 \,, \tag{4.17}$$
$$0 = (x_1 - x_2)(x_4 - x_5) + x_3 x_6 \,, \tag{4.18}$$

with $g = 9.81$, rod length $\ell = 0.61803399$, trolley mass $m_1 = 10$ and load mass $m_2 = 2.71828182 \approx \mathrm{e}$. Herein, states x_1, x_2 and x_3 define the system configuration, x_4, x_5 and x_6 the velocity and x_7, x_8 are algebraic variables used to impose joint constraints; input u is the control force acting on the trolley. We aim at minimizing the

TABLE 1
Vehicle — results with OCPID-DAE1.

guess	fixed time	free time	
T [s]	iterations [-]	iterations [-]	$T^\star$ [s]
21	32	36	23.2554
23	31	32	23.2554
25	32	34	23.2554
30	37	33	23.2554
40	33	37	23.2554

cost functional

$$J(\boldsymbol{x}, u, T) = \int_0^T w_u u^2(t)\mathrm{d}t + T + w_x x_1^2(T) + w_v x_4^2(T) \ , \tag{4.19}$$

with $w_x = 100$, $w_v = 100$, $w_u = 3.14159265 \approx \pi$, while satisfying boundary conditions

$$x_1(0) = x_2(0) = 1 \ , \qquad x_3(0) = -\ell \ , \tag{4.20}$$

$$x_4(0) = x_5(0) = 0 \ , \qquad x_6(0) = 0 \ , \tag{4.21}$$

$$x_1(T) = x_2(T) \ , \qquad x_4(T) = x_5(T) \ , \tag{4.22}$$

and control and final time bounds $u(t) \in [-1, 1]$, $t \in [0, T]$, $T \in [0.05, 12]$. For the LLPs, a discretization of 201 equidistant grid points and a implicit integration method are adopted. In Algorithm 3 we set $\kappa = (1 + \sqrt{5})/2$.

The reduced cost and Hamiltonian (2.5)–(2.6) for this example are depicted in Fig. 3. The reduced cost $\tilde{J}$ exhibits a (global) minimum around $T \approx 9$ and a (global) maximum around $T \approx 0.5$; as expected, the reduced Hamiltonian $\tilde{H}$ crosses zero with positive and negative slope, respectively. Notice that $T = 0.05$ is a (local) constrained minimizer for $\tilde{J}$. Once again, condition (2.7) is numerically tested by means of a finite difference approximation of $\tilde{H}$ based on $\tilde{J}$; it matches very well, see Fig. 3. At both, the lower and the upper bound of the final time box constraint, the reduced Hamilton function is positive; hence, for this problem, the initial guess for the final time is always tested by Algorithm 3. Then, for different initial guesses, the sequence of reduced cost $\tilde{J}$, reduced Hamiltonian $\tilde{H}$ and final time T during iterations are reported in Fig. 3. In all cases the bilevel solver converges to the (unconstrained) minimum $T^\star = 9.1914$.

The comparison results for this example are reported in Table 2. Starting from different initial guesses, the direct single shooting code, with time transformation, ends up with the (global) optimal solution only in one case, namely with initial guess $T_s = 4$, running through 43 iterations. For $T_s \in \{8, 9, 10\}$, it converges to the (local, constrained) minimum at $T = 0.05$, after 27–31 iterations. On the other hand, for the fixed final time problem, it takes 10–12 iterations to converge. Hence, for this problem, the bilevel approach requires in total around 3 times more iterations (for the OCP solver) but it exhibits a more reliable and robust behavior, which is a valuable benefit.

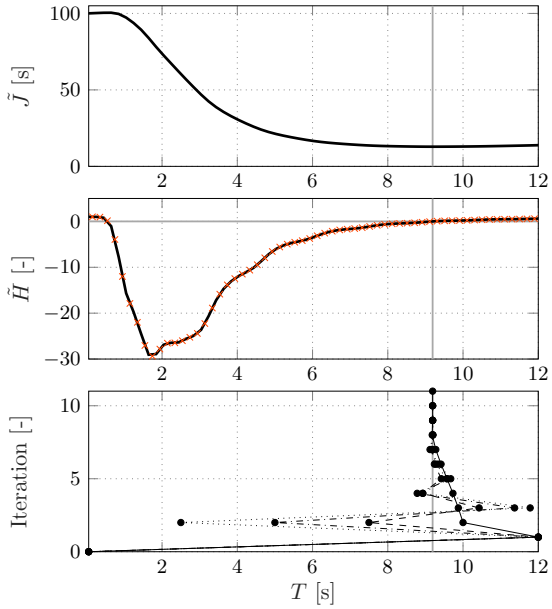

Figure 3: Trolley — Reduced cost $\tilde{J}$, reduced Hamiltonian $\tilde{H}$ with finite difference approximation (cross), and final time T during iterations, with initial guess $T_s = 2.5$ (dotted), $T_s = 5$ (dash-dotted), $T_s = 7.5$ (dashed) and $T_s = 10$ (solid).

References

[1] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings and M. Diehl, *CasADi: a software framework for nonlinear optimization and optimal control*, Math. Progr. Comp. (2018).

[2] A. Beck and M. Teboulle, *Gradient-based algorithms with applications to signal-recovery problems*, in Convex Optimization in Signal Processing and Communications, Cambridge University Press, Cambridge, 2009.

[3] P. Bosetti and F. Biral, *Application of optimal control theory to milling process*, in IECON 2014 - 40th Annual Conference of the IEEE Industrial Electronics Society (2014), pp. 4896–4901.

[4] Team Commands, Inria Saclay, *BOCOP: an open source toolbox for optimal control*, 2017. Online: www.bocop.org

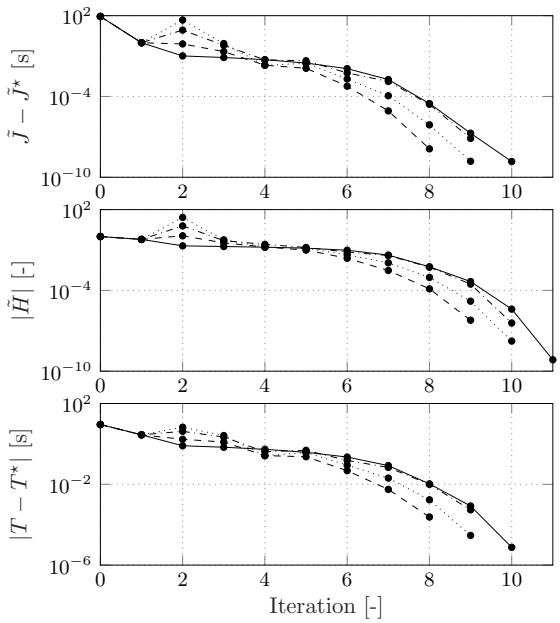

Figure 4: Trolley — Reduced cost $\tilde{J}$, reduced Hamiltonian $\tilde{H}$ and final time T during iterations, with initial guess $T_s = 2.5$ (dotted), $T_s = 5$ (dash-dotted), $T_s = 7.5$ (dashed) and $T_s = 10$ (solid).

Table 2
Trolley — results with OCPID-DAE1.

guess	fixed time	free time	
T [s]	iterations [-]	iterations [-]	$T^\star$ [s]
4	12	43	9.1914
8	11	27	0.05
9	10	29	0.05
10	11	31	0.05

[5] R. P. Brent, *An algorithm with guaranteed convergence for finding a zero of a function*, Computer Journal, 14:4 (1971), pp. 422–425.

[6] M. Burger and M. Gerdts, *A Survey on Numerical Methods for the Simulation of Initial Value Problems with DAEs*, in Surveys in Differential-Algebraic Equations IV, Springer International Publishing, Cham, 2017.

[7] J. Burkardt, *ZERO_RC: Nonlinear Equation Solver, Reverse Communication*, 2013. Online: https://people.sc.fsu.edu/~jburkardt/m_src/zero_rc

[8] B. Colson, P. Marcotte and G. Savard, *An overview of bilevel optimization*, Ann. Oper. Res., 153 (2007), pp. 235–256.

[9] S. Dempe, *Foundations of Bilevel Programming*, Kluwer Academic Publishers, Dordrecht, 2002.

[10] O. Devolder, F. Glineur and Y. Nesterov, *First-order methods of smooth convex optimization with inexact oracle*, Math. Program., 146 (2014), pp. 37–75.

[11] A. De Marchi and M. Gerdts, *Free finite horizon LQR: a bilevel perspective and its application to model predictive control*, Automatica, 100 (2019), pp. 299–311.

[12] ———, *A Bilevel Approach for Nonlinear Optimal Control Problems with Free Final Time*, submitted for publication, (2018). DOI: 10.5281/zenodo.2602459.

[13] M. Gerdts, *Optimal Control of ODEs and DAEs*, De Gruyter, Berlin, Boston, 2011.

[14] ———, *OCPID-DAE1 — optimal control and parameter identification with differential-algebraic equations of index 1*, User's guide, Bundeswehr University Munich, 2013. Online: www.optimal-control.de

[15] A. Locatelli, *Optimal control: An introduction*, Birkhäuser, Basel, 2001.

[16] R. Lot and F. Biral, *A Curvilinear Abscissa Approach for the Lap Time Optimization of Racing Vehicles*, in IFAC Proceedings Volumes, 47:3 (2014), pp. 7559–7565.

[17] N. Parikh and S. Boyd, *Proximal Algorithms*, Foundations and Trends® in Optimization, 1:3 (2014), pp. 127–239.

[18] L. S. Pontryagin, V. G. Boltyanskii, R. V. Gamkrelidze and E. Mishchenko, *The mathematical theory of optimal processes* in International series of monographs in pure and applied mathematics, Interscience Publishers, 1962.

[19] L. Van den Broeck, M. Diehl and J. Swevers, *Time Optimal MPC for mechatronic applications*, in Proceedings of the IEEE Conference on Decision and Control (2009), pp. 8040–8045.

[20] R. Verschueren, S. De Bruyne, M. Zanon, J. V. Frasch and M. Diehl, *Towards time-optimal race car driving using nonlinear MPC in real-time*, in Proceedings of the IEEE Conference on Decision and Control (2014), pp. 2505–2510.