IEEJ Journal of Industry Applications
Vol.5 No.2 pp.154–166 DOI: 10.1541/ieejjia.5.154

Paper

Notes on Numerical Methods for Solving Optimal Control Problems

Francesco Biral[*a)] Non-member, Enrico Bertolazzi[*] Non-member
Paolo Bosetti[*] Non-member

(Manuscript received June 1, 2015, revised Nov. 29, 2015)

Recent advances in theory, algorithms, and computational power make it possible to solve complex, optimal control problems both for off-line and on-line industrial applications. This paper starts by reviewing the technical details of the solution methods pertaining to three general categories: dynamic programming, indirect methods, and direct methods. With the aid of a demonstration example, the advantages and disadvantages of each method are discussed, along with a brief review of available software. The main result that emerges is the indirect method being numerically competitive with the performance of direct ones based on non-linear programming solvers and interior point algorithms. The second part of the paper introduces an indirect method based on the Pontryagin Minimum Principle (PMP). It also presents a detailed procedure and software tools (named PINS) to formulate the problem, automatically generate the C++ code, and eventually obtain a numerical solution for several optimal control problems of practical relevance. The application of PMP relates to the analytical derivation of necessary conditions for optimality. This aspect—often regarded in the literature as a drawback—is here exploited to build a robust yet efficient numerical method that formally eliminates the controls from the resulting Boundary Value Problem, thus gaining robustness and a high convergence rate. The elimination of the control is obtained either via their explicit formulation function of state and Lagrange multipliers—when possible—or via an iterative numerical solution. The paper closes presenting a minimum time manoeuvre of a car using a fairly complex vehicle model which also includes tyre saturation.

Keywords: optimal control, direct methods, indirect methods

1. Introduction

Complex practical optimal control problems (OCPs) both for off-line and on-line applications can be solved thanks to recent advances in theory, the availability of more efficient optimisation and root finding algorithms, and the increase of hardware computational power. Solution methods for optimal control (OC) problems can be categorised in three different groups: Dynamic programming, indirect Methods, and direct Methods. However, more than 90% of software available and solution schemes proposed in the literature are based on direct methods that—in the essence—transform the OC into a large, non-linear optimisation problem (or Non Linear Programming, NLP). In particular pseudo-spectral state variables and controls approximation is recently wide spreading in many software implementation. One of the reason of direct approaches to be so popular is the availability of many robust and ready-to-use optimisation algorithms (NLP solvers), which are software that can easily handle different types of inequality constraints and do not require direct calculation of co-states (or adjoint) equations as indirect methods do. In the literature the number of review articles about numerical methods to solve optimal control problems is wide and the most recents [(1)(2)] largely cover the numerical methods pertaining to the direct and indirect approaches with discussion on numerical issues. From these, more or less openly, it emerges that the indirect methods are not computationally efficient as the indirect ones and less appealing due to the need of more equations manipulation especially to derive the co-state equations.

Despite the general opinion emerging from the literature, the authors of this paper believe that robust and fast converging algorithms based on indirect method are yet possible, with the advantage of providing solutions with better accuracy and of being suitable for symbolic manipulation to obtain analytical or semi-analytical solutions.

Therefore, we think that the previous review articles could be complemented with more insight on indirect methods and comparisons with direct ones to understand if there is a substantial difference between the two families of methods both in term of type of problem which is being solved and the accuracy of the solution obtained.

With this goals in mind the contribution of the paper is two fold. First it is not a pure description of the theory related to the different solution methods, as it is done in many review papers, but it also presents and discusses the numerical implementations by means of a demonstration example and providing the source code for further analysis of interested readers. The second contribution is an in-depth analysis of an indirect approach which is most of the time overlooked because of historical reasons and essentially the lack of robust and fast root-finding solvers.

In this work it is shown that the indirect method is flexible, fast and accurate which also provides, differently than NLP, the possibility to find analytical solutions.

a) Correspondence to: francesco.biral@unitn.it
[*] Dep. Industrial Engineering, University of Trento
Via Sommarive, 9 – I-38123 Povo, Trento, Italy

The work is essentially structured into two parts. The first reviews the three different families of solution methods of OCPs and compares an implementation of each of them with a simple, yet significative, optimal control problem. Solution quality and performance results are discussed together with other implementation issues.

The second part of the paper discusses the details of a proposed indirect method that was successfully applied to many industrial engineering problems.

2. Optimal Control Problem Definition

An optimal control problem is a constrained optimisation problem with a dynamical system as constraint. Let us consider the following initial value problem (IVP):

$$\dot{x}(t) = f\,(x(t), u(t), t)\,, \quad x(0) = x_0, \quad t \in [0, T] \qquad \cdots\cdots (1)$$

where $x(t) \in \mathbb{R}^n$ are the states of the dynamical system and $u(t) \in \mathcal{U} \subset \mathbb{R}^m$ are the controls. If the control vector $u(t)$ is known and sufficiently regular (e.g. piecewise continuous) and $f(x, u, t)$ is regular enough as a function of (x, u, t) (e.g. continuous in (x, u, t), and Lipschitz in x) then IVP has an unique solution that depends on the control history $u(t)$. In more complex problems, the initial condition $x(0) = x_0$ may be replaced with a general boundary condition $b(x(0), x(T)) = \mathbf{0}$. Furthermore, path constraints $c(x(t), u(t), t) \geq \mathbf{0}$ may also be present. However, here, to simplify the explanation and comparison among the three families of solution methods for OCP, the path constraints are here omitted and introduced later on in the article in Sect. 5.

Let us now define a *feasible control* a control history $\bar{u}(t) \in \mathcal{U}$, which determines a function $\bar{x}(t)$ (i.e. a history of states) for the dynamical system (1) that satisfies the set of boundary conditions, and—when defined—the path constraints. The pair $(\bar{u}(t), \bar{x}(t))$ is called *feasible manoeuvre*.

Now, lets introduce a *performance functional index*:

$$J[u] = \mathcal{M}(x(T)) + \int_0^T \ell(x(t), u(t), t)\,\mathrm{d}t \cdots\cdots (2)$$

This functional $J[u]$ evaluates to a scalar, where function $x(t)$ is the solution of the ODE (1) given a *control manoeuvre* $\bar{u}$.

The term $\mathcal{M}(\cdot)$ is called *final cost* or *Mayer's term*, while $\ell(\cdot)$ is named *running cost* or *Lagrange's term*.

Under these definitions, the solution of an *optimal control problem* is a *feasible manoeuvre*, or pair of control history u and state x, which minimises the performance index $J[u]$ when the dynamical system transits from the initial state to the final one satisfying the path constraints.

3. Methods to Solve Optimal Control Problems

This section summaries the available methods to solve optimal control problems, also providing references to the available software. It is intended to provide a sufficiently wide and clear overview of the differences among the available formulations of optimal control problems and solution techniques, rather than being a thoroughly report of the state-of-the-art. For a more extensive survey of numerical methods for optimal control see [1][2]. In short OCPs solution methods can be classified in three main families: dynamic programming, indirect method based on calculus of variations and direct methods. Figure 1 is one possible visualisation of both OCPs formulations and solution methods. Each of these three groups are discussed in details and an example of implementation for the demonstration problem is provided. Comparisons, numerical issues and theoretical discussion are presented based on the result obtained.

3.1 Dynamic Programming Dynamic programming (DP) methodology was developed in the fifties and sixties of the 19th century, most prominently by Richard Bellman [3]. Dynamic programming can be easily applied to systems having discrete states and control spaces. In case of continuous time, DP for (1) with performance index (2) and final condition $b(x(T), T) = \mathbf{0}$, can be formulated by introducing the function describing the cost-to-go to the end when starting at a given state:

$$V(y, t) = \min_{u} \left\{ \mathcal{M}(x(T; u)) + \int_t^T \ell(x(s; u), u(s), s)\,\mathrm{d}s \right\} \qquad \cdots\cdots (3)$$

where function $x(s; u)$ is the solution of the ODE (1) in the range $[t, T]$ with initial data y and feasible control $u(s)$. Obviously $V(x_0, 0)$ is equal to the performance index $J[u]$ when u is the optimal control of the problem while $V(x, T) = \mathcal{M}(x, T)$ for all x. Function $V(x, t)$ is called *value function* and satisfies a partial differential equation in the state space, the Hamilton-Jacobi-Bellman (HJB) equation:

$$\frac{\partial}{\partial t} V(x, t) + \min_{u \in \mathcal{U}} H(x, \nabla_x V(x, t), u, t) = \mathbf{0}, \quad t \in (0, T) \qquad \cdots\cdots (4)$$

$$V(x, T) = \mathcal{M}(x), \quad x \in \mathbb{R}^n$$

where the function H is the Hamiltonian:

$$H(x, \lambda, u, t) = \ell(x, u, t) + \lambda \cdot f(x, u, t). \cdots\cdots (5)$$

The *principle of optimality* states that each sub-trajectory of an optimal trajectory is an optimal trajectory too. This means that Eq. (4) can be solved over a shorter horizon starting from the end $V(x, T) = \mathcal{M}(x)$ and recursively find the complete solution proceeding backwards. Having solved the PDE (4), the optimal feedback control u_{fb} for the state x at time t is then obtained from:

$$u_{\mathrm{fb}}(x, t) = \arg\min_{u \in \mathcal{U}} H(x, \nabla_x V(x, t), u, t). \cdots\cdots (6)$$

The optimal solution is obtained integrating (1) with the optimal feedback control (6), i.e.

$$\dot{x}(t) = f\,(x(t), u_{\mathrm{fb}}(x, t), t)\,, \quad x(0) = x_0, \quad t \in [0, T] \qquad \cdots\cdots (7)$$

3.1.1 Applicability of DP The construction of the optimal feedback control (6) depends on the availability of the *value function* $V(x, t)$. Unless exceptional cases where value function is analytically computable, the function $V(x, t)$ must be approximated using numerical technique. For example using function expansion, finite difference or finite elements or other numerical techniques for PDE. This approach is feasible only for PDE with low dimensional space or equivalently for low dimension of state space. An exception is

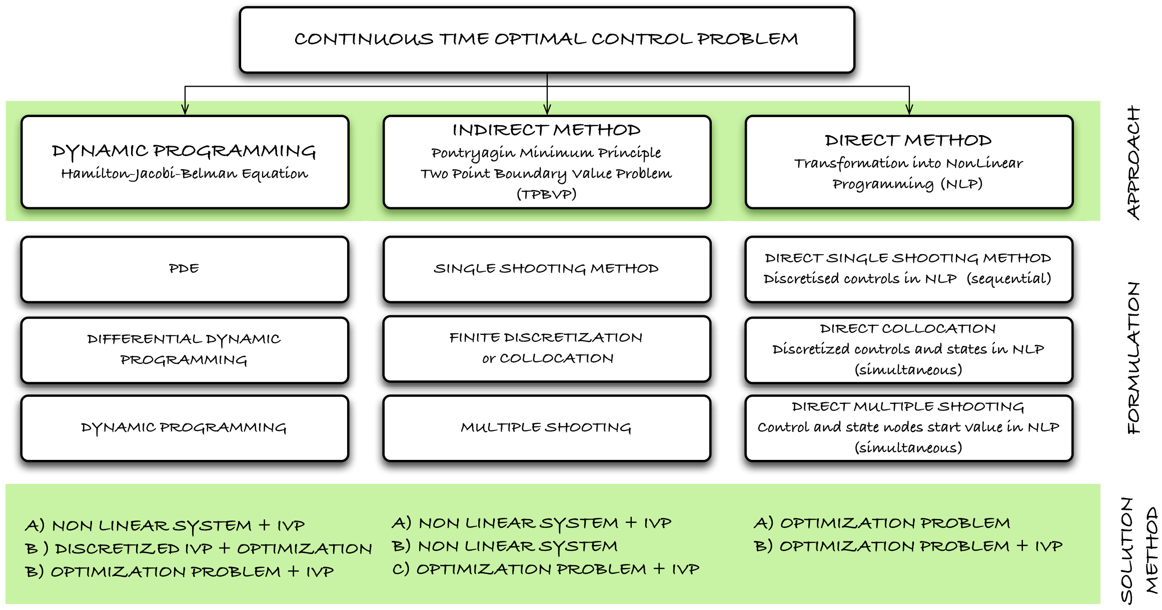
CONTINUOUS TIME OPTIMAL CONTROL PROBLEM
DYNAMIC PROGRAMMING
Hamilton-Jacobi-Belman Equation
INDIRECT METHOD
Pontryagin Minimum Principle
Two Point Boundary Value Problem (TPBVP)
DIRECT METHOD
Transformation into NonLinear Programming (NLP)
APPROACH
PDE
SINGLE SHOOTING METHOD
DIRECT SINGLE SHOOTING METHOD
Discretised controls in NLP (sequential)
DIFFERENTIAL DYNAMIC PROGRAMMING
FINITE DISCRETIZATION or COLLOCATION
DIRECT COLLOCATION
Discretized controls and states in NLP (simultaneous)
DYNAMIC PROGRAMMING
MULTIPLE SHOOTING
DIRECT MULTIPLE SHOOTING
Control and state nodes start value in NLP (simultaneous)
FORMULATION
A) NON LINEAR SYSTEM + IVP
B) DISCRETIZED IVP + OPTIMIZATION
B) OPTIMIZATION PROBLEM + IVP
A) NON LINEAR SYSTEM + IVP
B) NON LINEAR SYSTEM
C) OPTIMIZATION PROBLEM + IVP
A) OPTIMIZATION PROBLEM
B) OPTIMIZATION PROBLEM + IVP
SOLUTION METHOD

Fig. 1. Classification of different methods to solve optimal control problems and related formulations and solution algorithms

when the performance functional index (2) is linear quadratic with linear dynamical system

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{A}(t)\boldsymbol{x}(t) + \boldsymbol{B}(t)\boldsymbol{u}(t), \quad \boldsymbol{x}(0) = \boldsymbol{x}_0, \quad t \in [0, T] \quad \cdots\cdots\cdots (8)$$

$$J[\boldsymbol{u}] = \frac{\boldsymbol{x}(T)^T\boldsymbol{C}\boldsymbol{x}(T)}{2} + \frac{1}{2}\int_0^T \boldsymbol{x}^T\boldsymbol{D}(t)\boldsymbol{x} + \boldsymbol{u}^T\boldsymbol{E}(t)\boldsymbol{u}\,\mathrm{d}t \quad \cdots\cdots\cdots (9)$$

In this case it is well known that the value function takes the form $V(\boldsymbol{x}, t) = \frac{1}{2}\boldsymbol{x}^T\boldsymbol{P}(t)\boldsymbol{x}$ where the matrix function $\boldsymbol{P}(t)$ satisfy the Riccati ODE

$$\boldsymbol{P}' + \boldsymbol{P}\boldsymbol{A} + \boldsymbol{A}^T\boldsymbol{P} + \boldsymbol{D} - \boldsymbol{P}\boldsymbol{B}\boldsymbol{E}^{-1}\boldsymbol{B}^T\boldsymbol{P} = \boldsymbol{0}, \quad \boldsymbol{P}(T) = \boldsymbol{C} \quad \cdots\cdots\cdots (10)$$

and thus the cost for the solution of the problem is very low.

3.1.2 Discrete Version of DP When DP is applied to discrete time systems with continuous state spaces, some approximations have to be made, usually by discretization. The DP solution methods examine all the feasible state trajectory candidates that satisfy the necessary condition, by breaking down the global problem into local subproblems for every (reachable) discrete state and time instant. However, the computation time scales with the number of feasible state trajectory candidates, which in turn increases exponentially with the number of states and control variables, and increases linearly with the quantization resolution of the continuous states. The computation time can be reduced by decomposing the problem into cascaded of subproblems, or by approximating the problem [(4)], or again by applying Differential Dynamic Programming (DDP), which uses a sequential quadratic approximation of the value function as central role to incrementally obtain a solution of the optimal control problem [(5)]. An efficient implementation of the deterministic DP approach for optimal control of non-linear, time-variant, constrained, discrete-time approximations of continuous-time dynamic models is presented in [(6)]. The related free software can be downloaded from http://www.idsc.ethz.ch/research-guzzella-onder [(7)] and it is used for solving the demonstration example in Sect. 4.1.

3.2 Pontryagin Minimum Principle: Indirect Method

Another class of analytical optimisation method relates to Pontryagin's minimum principle [(8)] to derive the necessary conditions for optimality [(9)]. The method uses the Hamiltonian function (5) introduced for the Hamilton-Jacobi-Bellman equation and the global optimal control problem is reduced to the solution of the following system of $2N_x$ equations given in the form of a two-point boundary value problem (BVP):

$$\dot{\boldsymbol{x}}(t) = +\partial_{\boldsymbol{\lambda}}^T\mathcal{H}(\boldsymbol{x}(t), \boldsymbol{\lambda}(t), \boldsymbol{u}(t), t) \quad \cdots\cdots\cdots (11)$$

$$\dot{\boldsymbol{\lambda}}(t) = -\partial_{\boldsymbol{x}}^T\mathcal{H}(\boldsymbol{x}(t), \boldsymbol{\lambda}(t), \boldsymbol{u}(t), t) \quad \cdots\cdots\cdots (12)$$

where $\boldsymbol{\lambda}$ are additional Lagrange multiplier functions. The boundary conditions are the original ones augmented with the transversality conditions (because the conditions, not necessarily on the state variables only, are given both at the start and the end of the time horizon)

$$\boldsymbol{x}(0) = \boldsymbol{x}_0, \quad \boldsymbol{\lambda}(T) = \partial_{\boldsymbol{x}}^T\mathcal{M}(\boldsymbol{x}(T)) \quad \cdots\cdots\cdots (13)$$

Control $\boldsymbol{u}(t) = \boldsymbol{u}_{\mathrm{fb}}(\boldsymbol{x}(t), \boldsymbol{\lambda}(t), t)$ is determined by a local optimization of the Hamiltonian at each time instant:

$$\boldsymbol{u}_{\mathrm{fb}}(\boldsymbol{x}, \boldsymbol{\lambda}, t) = \underset{\boldsymbol{u}\in\mathcal{U}}{\arg\min}\, H(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{u}, t). \quad \cdots\cdots\cdots (14)$$

which let one choose explicitly the decision between the actual cost and the equivalent cost of the system dynamics. The problem (11–12–14) provides only *necessary conditions* that can be solved numerically (only in few cases analytically) to obtain candidate solution of the optimal control problem. The solution need to be checked with the condition obtained for the second variation. The controls may not appear directly in the BVP as unknowns since the can be solved using the condition (14) and substituted back into the differential equations, es explained in next sections.

Indirect methods can treat easily different boundary condition in dynamical system, for example (1) can be substituted with

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{f}\left(\boldsymbol{x}(t), \boldsymbol{u}(t), t\right), \quad \boldsymbol{b}(\boldsymbol{x}(0), \boldsymbol{x}(T)) = \boldsymbol{0}, \cdots\cdots (15)$$

and, thus, boundary conditions (13) becomes

$$\begin{cases} \boldsymbol{b}(\boldsymbol{x}(0), \boldsymbol{x}(T)) = \boldsymbol{0}, \\ \partial^T_{\boldsymbol{x}(0)}\boldsymbol{b}(\boldsymbol{x}(0), \boldsymbol{x}(T)) \cdot \boldsymbol{\omega} = \boldsymbol{\lambda}(0), \\ \partial^T_{\boldsymbol{x}(T)}\boldsymbol{b}(\boldsymbol{x}(0), \boldsymbol{x}(T)) \cdot \boldsymbol{\omega} = -\partial^T_{\boldsymbol{x}}\mathcal{M}(\boldsymbol{x}(T)) - \boldsymbol{\lambda}(T), \end{cases}$$
$$\cdots\cdots\cdots\cdots\cdots\cdots\cdots (16)$$

where $\boldsymbol{\omega}$ is a multiplier that can be easily eliminated when, for example, boundary conditions are separated, i.e. when $b_i(\boldsymbol{x}(0), \boldsymbol{x}(T))$ is a function of $\boldsymbol{x}(0)$ or $\boldsymbol{x}(T)$ alone. This formulation is further generalised in Sect. 5 but here it is sufficient to shortly introduce different solution methods and provide a simple demonstrative example in Sect. 6.1. Solution methods span from single shooting, to multiple shooting and different types of collocation being the last one the most robust and fast converging. An example of indirect method solver is the one explained in [10].

3.3 Non Linear Programming: Direct Method The essence of a direct method is the discretisation of the original optimal control problem [11] which is then transcribed to a nonlinear programming problem (NLP) solved numerically using a well-established optimisation method [12][13]. The category of the direct methods is quite broad and encompasses very different techniques. In particular, methods differs for the variables to be discretised (i.e. control and states) and how to approximate the continuous time dynamics. In the case of the *shooting* and *multiple shooting* the control are parameterised with piecewise linear functions and the differential equations are solved via numerical integration. These approaches make use of robust and available ordinary differential equations solvers but need sensitivity analysis to compute the jacobians of the continuity and boundary conditions with respect to the initial and intermediate conditions. In the case of state and control parameterisation (*direct collocation*), both states and controls are approximated with polynomial functions, therefore the continuous time differential equations are converted into algebraic constraints. These constraints are then imposed in the NLP formulation, which avoid the sensitivity issues of direct shooting methods at the expense of a larger NLP. A class of direct collocation is the *spectral method* where the optimal control problem is transcribed to a NLP by parameterising the state and control using global orthogonal polynomials (*i.e. spectral*) and collocating the differential—algebraic equations using nodes obtained from a Gaussian quadrature. The use of global polynomials together with Gaussian quadrature collocation points is known to provide accurate approximations that converge exponentially for problems whose solutions are smooth. In case the solution is non smooth the domain is divided in segments and different approximation are used in each of them (*pseudo-spectral*). Over the last few years, pseudo-spectral methods for solving optimal control problems have rapidly widespread in real-world applications making it possible to solve highly complicated nonlinear optimal control problems in real-life applications [14]–[16].

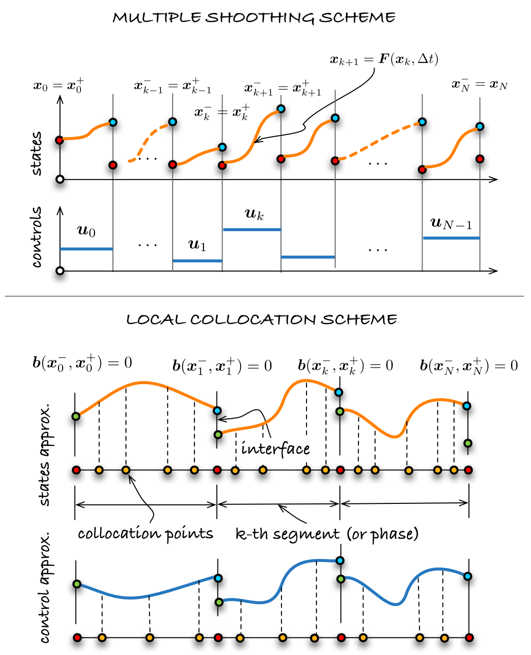

Fig. 2. Upper figure shows the scheme of the concept of the Multiple shooting. The lower figure visualises the concept of the pseudo-spectral method

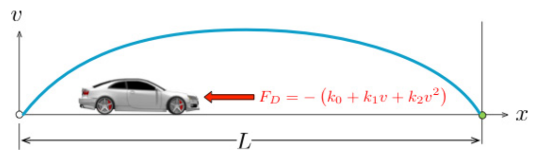

Fig. 3. Sketch of the simple model of the longitudinal dynamics of a vehicle that has to travel stop-to-stop a maximum distance L in given time T

Figure 2 visualises the essence of the two families of NLP methods. A list of the most known available software in this category are ACADO [17], GPOPS-II [18], PSOPT, MISER, SOCS, DIRCOL, PROPT [19], RIOTS [20], MUSCOD-II [21], DIDO [22].

4. Demonstration Example of Optimal Control Solution

In this section, as an example, a simple, yet numerically difficult, non linear optimal control problem is used to compare the numerical solution methods pertaining to the three families above described. The proposed problem, under some assumptions, has an analytical solution that it is used to compare the accuracy of the numerical methods. The scripts and codes developed and used in this article to find the numerical solutions are available at https://github.com/mechatronix/TS-OCS.

The demonstration example is a simple model of the longitudinal dynamics of a vehicle with the aerodynamic downforce. Let us consider the vehicle as a point mass that has to be moved of a maximum distance in a given fixed time T from initial zero velocity to final zero velocity:

$$\min\,\{x(0) - x(T)\} = \min\left(-\int_0^T v\,\mathrm{d}t\right) \cdots\cdots\cdots (17)$$

$$\text{subject to:}\quad \dot{x} = v, \quad \dot{v} = u - k_0 - k_1 v - k_2 v^2 \cdots\cdots\cdots (18)$$

$$\text{b.c.}\quad x(0) = v(0) = v(T) = 0, \cdots\cdots\cdots\cdots\cdots\cdots (19)$$

control limits: $|u| \le g + k_3 v^2$ ················ (20)

The control u is limited between two limits which are function of the square of the velocity.

For the proposed problem the Hamiltonian (5) becomes:

$$H(x, v, \lambda, \mu, u) = -v + \lambda v + \mu\left(u - k_0 - k_1 v - k_2 v^2\right)$$

and the control satisfies (14) so that optimal control $u^\star$ is

$$u^\star = \underset{|u|\le g+k_3v^2}{\arg\min}\, H(x, v, \lambda, \mu, u) = -\left(g + k_3 v^2\right)\operatorname{sign}(\mu) \qquad (21)$$

The BVP (11–12) becomes:

$$\begin{cases} \dot{x} = \partial_\lambda H = v \\ \dot{v} = \partial_\mu H = u - k_0 - k_1 v - k_2 v^2 \\ \quad = -\left(g + k_3 v^2\right)\operatorname{sign}(\mu) - k_0 - k_1 v - k_2 v^2 \\ \dot{\lambda}(t) = -\partial_x H = 0 \\ \dot{\mu}(t) = -\partial_v H = 1 - \lambda + \mu\,(k_1 + 2vk_2) \end{cases} \qquad (22)$$

while boundary condition (of type (16)) with

$$\boldsymbol{b}(x(0), v(0), x(T), v(T)) = \left(x(0) \quad v(0) \quad v(T)\right)^T,$$
$$\boldsymbol{b}(x(0), v(0), x(T), v(T)) \cdot \boldsymbol{\omega} = \omega_1 x(0) + \omega_2 v(0) + \omega_3 v(T)$$

becomes

$$x(0) = v(0) = v(T) = 0, \quad \lambda(T) = 0,$$
$$\lambda(0) = \omega_1, \quad \mu(0) = \omega_2, \quad \mu(T) = \omega_3$$

in this case ω_1, ω_2 and ω_3 may take any values and are eliminated, thus, $\lambda(0), \mu(0)$ and $\mu(T)$ are free. Boundary conditions reduce to

$$x(0) = v(0) = v(T) = \lambda(T) = 0. \qquad (23)$$

The optimal control problem, assuming parameters $k_2 = 0$ and $k_3 = 0$, has an analytical solution that can be derived by solving Eq. (22) with boundary condition (23). From Eq. (21) and differential equations of Lagrange multipliers one understands that the control depends on the sign of the multiplier μ which is monotonically increasing with time and therefore has only one switching point t_s. The solution can be easily found by matching the solution obtained from forward integration starting from initial conditions with the one obtained by backward integration starting from final conditions. The set of four algebraic constraints is augmented with the algebraic equation that set the Lagrange multiplier μ to be equal to zero at t_s (i.e. switching condition). The solution for the optimal control renders as follow:

$$u(t) = \begin{cases} 1 & t \le t_s, \\ -1 & t > t_s, \end{cases} \quad t_s = \frac{1}{k_1}\ln\frac{g_- + g_+ e^{k_1 T}}{2g}, \qquad (24)$$

where t_s is switching time, $g_+ = g + k_0$ and $g_- = g - k_0$. The exact solution for $t \le t_s$ is given by

$$s(t) = k_1^{-2} g_-(k_1 t + \mathrm{e}^{-k_1 t} - 1),$$
$$v(t) = k_1^{-2} g_-(1 - \mathrm{e}^{-k_1 t}),$$

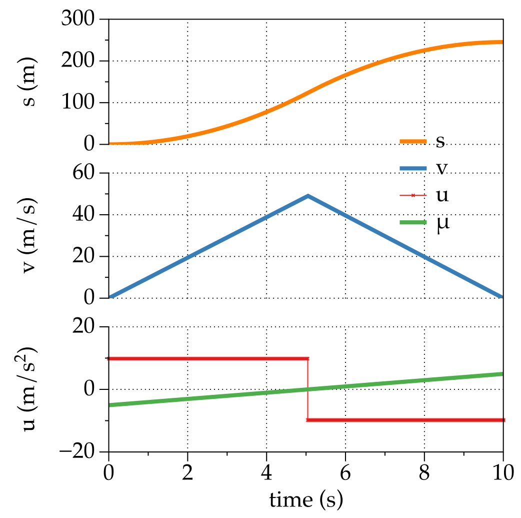

Fig. 4. The exact solution for an horizon of $T = 10s$, and with parameters $g = 9.81$, $k_0 = g \cdot 10^{-2}$, $k_1 = g \cdot 10^{-5}$, $k_2 = 0$ and $k_3 = 0$

and for $t > t_s$

$$s(t) = k_1^{-2}\left(g_+ + \mathrm{e}^{-k_1 t}\left(g_- - 2\,g\mathrm{e}^{k_1 t_s}\right) + k_1\,(2gt_s - tg_+)\right),$$
$$v(t) = k_1^{-1} g_+ \mathrm{e}^{k_1(T-t)},$$

while multiplier $\lambda(t)$ and $\mu(t)$ satisfy

$$\lambda(t) = 0, \quad \mu(t) = \frac{1}{k_1}\left(\frac{2\,g\,\mathrm{e}^{k_1(t-T)}}{g_-\mathrm{e}^{-k_1 T} + g_+} - 1\right)$$

Figure 4 show the exact solution for a particular choice of T, k_0 and k_1.

4.1 Numerical Solution with HJB Method The demonstration example is first solved using the dynamic programming approach based on the algorithm explained in[6]. The numerical method evaluates the region of feasible states and the cost-to-go function backward starting from the final condition using the forward Euler approximation for the given dynamical system:

$$\begin{cases} x_{k+1} = x_k + hv_k \\ v_{k+1} = v_k + h\left(u_{k+1/2} - k_0 - k_1 v_k - k_2 v_k^2\right) \end{cases}$$

that can be synthesised as:

$$\boldsymbol{y}_{k+1} = \boldsymbol{F}(\boldsymbol{y}_k, u_{k+1/2}, h), \qquad \boldsymbol{y} = \begin{pmatrix} x \\ v \end{pmatrix} \qquad (25)$$

The cost-to-go function is initialised as follows:

$$\mathcal{J}_N(x^i, v^i) = p(v^i) \qquad (26)$$

where $p(v)$ is a penalty to enforce the final condition on the velocity. Then, with a backward iteration for $k = N-1, \ldots, 0$, $\forall \boldsymbol{x}^i \in \mathcal{X}_k$, the cost-to-go is evaluated:

$$\mathcal{J}_k(\boldsymbol{y}^i) = -h\,v(\boldsymbol{y}^i, u^\star, h) + \mathcal{J}_{k+1}(F(\boldsymbol{y}^i, u^\star, h))$$

where $u^\star$ is calculated by solving the local minimisation

$$u^\star = \underset{u\in\mathcal{U}_k}{\arg\min}\{-v_k(u_{k+1/2}) + \mathcal{J}_{k+1}(\boldsymbol{F}(\boldsymbol{y}_k, u_{k+1/2}, h)\}$$

The algorithm simulates the system over one time step by applying all possible control candidates. A multilinear interpolation is used to find the values of the cost-to-go at the

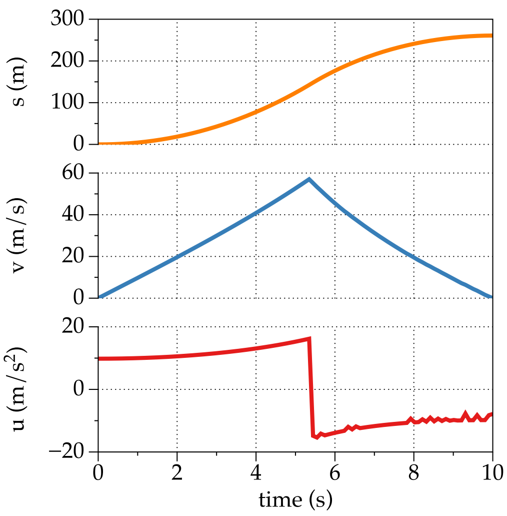

Fig. 5. Numerical solution obtained with DPM with state variables discretised with $N_x = 201$ points, control discretised with $N_u = 101$ points and time discretised with $N_t = 101$ points. Final velocity was constrained to lie in the range $[0, 0.1]$ enforced via the penalty $p(v_N)$

node *k-th* which is set to a high value if the state is not reachable. Finally, with a forward evaluation the optimal control policy $\pi = \{u_0(\boldsymbol{x}), u_1(\boldsymbol{x}), \ldots, u_{N_1}(\boldsymbol{x})\}$ that respects the initial and final conditions is searched.

Figure 5 shows the solution obtained with the DP using algorithm available at[23]. The method was setup with the `level-set` option to get a more accurate solution yet in less time. Nevertheless it takes about 270 seconds to find the solution. The small control oscillations, induced by the penalty function to enforce the final velocity to be zero, could be reduced increasing the time and states quantization at the price of higher computational costs.

4.2 Numerical Solution with Direct Method This methods numerically solve the Non Linear Programming (NLP) problem resulting by the direct discretisation of (17-20). For example using finite difference the following NLP is obtained:

- Minimize $-\sum_{k=1}^{N} v_{k-1/2}$
- With constraints

$$\begin{cases} x_{k+1} = x_k + h\bar{v}_{k+1/2} \\ v_{k+1} = v_k + h\left(u_{k+1/2} - k_0 - k_1\bar{v}_{k+1/2} - k_2\bar{v}_{k+1/2}^2\right) \\ x_0 = 0, \quad v_0 = 0, \quad v_N = 0, \\ -g - k_3\bar{v}_{k+1/2}^2 \le u_{k+1/2} \le g + k_3\bar{v}_{k+1/2}^2 \end{cases} \qquad (27)$$

where $\bar{v}_{k+1/2} = (v_k + v_{k+1})/2$, N is the number of interval of size $h = T/N$ where the interval $[0, T]$ split. With f_k we denote the finite difference approximation of function $f(t)$ evaluated at $t_k = kh$. The unknowns of the problem are collected in $z \in \mathbb{R}^{3N+2}$

$$z = (x_0, \ldots, x_N, v_0, \ldots, v_N, u_{1/2}, u_{3/2}, \ldots, u_{N-1/2})^T \qquad (28)$$

and thus problem (27) can be written as the minimisation of $f(z)$ with constraints $h_k(z) = 0$ and $g_k(z) \ge 0$ where

$$f(z) = -\Sigma_{k=1}^{N} \bar{v}_{k-1/2} \qquad (29)$$

$$h_k(z) = x_{k+1} - x_k - h\bar{v}_{k+1/2} \qquad (30)$$

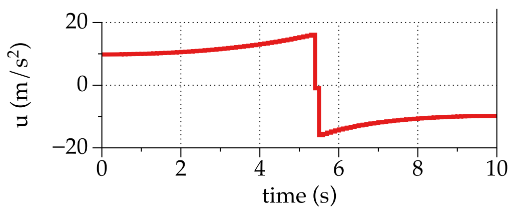

Fig. 6. NLP solution (only control u) using IPOPT in Matlab with $N = 100$ and also providing the jacobian. Dent in the control solution at the jump location is due to control discretisation

$$h_{k+N}(z) = v_{k+1} - v_k - h(u_{k+1/2} - k_0 - k_1\bar{v}_{k+1/2} - k_2\bar{v}_{k+1/2}^2) \qquad (31)$$

$$h_{2N+1}(z) = x_0, \quad h_{2N+2}(z) = v_0, \quad h_{2N+3}(z) = v_N \qquad (32)$$

$$g_k(z) = g + k_3\bar{v}_{k+1/2}^2 + u_{k+1/2} \ge 0, \qquad (33)$$

$$g_{k+N}(z) = g + k_3\bar{v}_{k+1/2}^2 - u_{k+1/2} \ge 0. \qquad (34)$$

This problem can be solved using available NLP solver. For reference the state of art nonlinear optimisation code are IPOPT[12], KNITRO[24], LOQO[25] and WORHP[26]. In this case IPOPT was used to find the numerical solution via its Matlab interface. Figure 6 show the obtained solution which is qualitatively identical to the exact solution.

4.3 Numerical Solution with Indirect Method This methods numerically solve BVP obtained using the Pontryagin maximum principle (21-22-23). For example using finite difference denoting with f_k the finite difference approximation of function $f(t)$ evaluated at $t_k = kh$ the following nonlinear system is obtained:

$$\begin{cases} x_{k+1} = x_k + h\bar{v}_{k+1/2} \\ v_{k+1} = v_k + h\left(u_{k+1/2} - k_0 - k_1\bar{v}_{k+1/2} - k_2\bar{v}_{k+1/2}^2\right) \\ \lambda_{k+1} = \lambda_k \\ \mu_{k+1} = \mu_k + h\left(1 - \bar{\lambda}_{k+1/2} + \bar{\mu}_{k+1/2}(k_1 + 2\bar{v}_{k+1/2}k_2)\right) \\ x_0 = 0, \quad v_0 = 0, \quad v_N = 0, \quad \lambda_N = 0, \end{cases} \qquad (35)$$

where $\bar{v}_{k+1/2} = (v_k + v_{k+1})/2$, $\bar{\mu}_{k+1/2} = (\mu_k + \mu_{k+1})/2$, $\bar{\lambda}_{k+1/2} = (\lambda_k + \lambda_{k+1})/2$, and N is the number of interval of size $h = T/N$ where the interval $[0, T]$ split. Moreover

$$\begin{aligned} u_{k+1/2} &= \underset{|u| \le g + k_3\bar{v}_{k+1/2}^2}{\arg\min}\; H(\bar{x}_{k+1/2}, \bar{v}_{k+1/2}, \bar{\lambda}_{k+1/2}, \bar{\mu}_{k+1/2}, u). \\ &= -\mathrm{sign}\,(\bar{\mu}_{k+1/2})\,(g + k_3\bar{v}_{1/2}^2) \end{aligned} \qquad (36)$$

The unknowns of the problem are collected in $z \in \mathbb{R}^{4N}$:

$$z = (x_0, \ldots, x_N, v_0, \ldots, v_N, \lambda_0, \ldots, \lambda_N, \mu_0, \ldots, \mu_N)^T$$

Nonlinear system (35) contains discontinuous function $\mathrm{sign}(x)$ which make it hard or impossible to solve. A simple working strategy is to smooth the problem by approximating problem (36) with interior point approach

$$u^\star \approx \underset{u \in \mathbb{R}}{\arg\min}\left(H(x, v, \lambda, \mu, u) + b(u)\right)$$

where $b(u)$ is a barrier function:

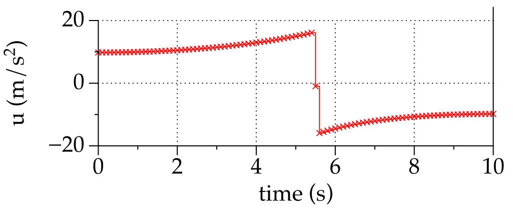

Fig. 7. Solution using PINS with $N = 100$

$$b(u) = -\varepsilon(g + k_3 v^2) \log \cos\left(\frac{\pi}{2}\frac{u}{g + k_3 v^2}\right) \cdots\cdots\cdots (37)$$

and the minima satisfy $\frac{\partial}{\partial u}\Big(H(x, v, \lambda, \mu, u) + b(u)\Big) = 0$, *i.e.*

$$0 = \mu + \varepsilon\frac{\pi}{2}\tan\left(\frac{\pi}{2}\frac{u}{g + k_3 v^2}\right) \cdots\cdots\cdots (38)$$

so that control u as a function of μ and v can be easily computed

$$u(\mu, v) = -\frac{2}{\pi}\left(g + k_3 v^2\right)\arctan\left(\frac{2\mu}{\pi\epsilon}\right) \cdots\cdots\cdots (39)$$

The nonlinear system (35) with Eq. (39), i.e. $u_{k+1/2} = u(\mu_{k+1/2}, v_{k+1/2})$ is easier to solve than the original nonlinear system.

Figure 7 shows the solution obtained using the indirect solver PINS described in Sect. 5.

Remark 1 Normally controls cannot be solved explicitly like in Eq. (39). Control are computed as the minimization respect to $\boldsymbol{u}$ of an Hamiltonian $H(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{u})$ with $\boldsymbol{x}$ the states and $\boldsymbol{\lambda}$ the multipliers. Using penalties and/or barriers this minimization is transformed into a nonlinear system that must be solved: $\partial_{\boldsymbol{u}} H(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{u}) = \boldsymbol{G}(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{u}) = \boldsymbol{0}$. Thus, the computation of $\boldsymbol{u}(\boldsymbol{x}, \boldsymbol{\lambda})$ is reduced to the computation of the solution of the nonlinear system $\boldsymbol{G}(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{u}) = \boldsymbol{0}$. The computation of the partial derivatives $\partial_{\boldsymbol{x}}\boldsymbol{u}(\boldsymbol{x}, \boldsymbol{\lambda})$ and $\partial_{\boldsymbol{\lambda}}\boldsymbol{u}(\boldsymbol{x}, \boldsymbol{\lambda})$ needed for the iterative solution of the nonlinear system generated by the indirect approach can be computed using implicit function theorem from the identity $\boldsymbol{G}(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{u}(\boldsymbol{x}, \boldsymbol{\lambda})) = \boldsymbol{0}$. In the previous example the identity $\boldsymbol{G}(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{u}) = \boldsymbol{0}$ is the single Eq. (38) with $\boldsymbol{x} = (x, v)^T$ and $\boldsymbol{\lambda} = (\lambda, \mu)^T$.

4.4 Discussion of Solution Methods

Despite the fact that DP is more suitable to solve mixed integer and continuous optimal control problems and provides a sufficient solution for the optimality it is less adopted as solution method due to the curse of dimensionality. Additionally, it requires the optimal cost to be sufficiently smooth, which is not always the case. Our demonstration test clearly shows that to get a solution with similar accuracy than the direct and indirect method it takes a computational time which is at least two order higher.

On the other hand, direct methods are recognised as having many advantages that make them appealing. The direct transcription and the subsequent NLP allows to easily handle the inequality constraints via a variety of off-the-shelf robust optimisation software. The minimisation problem is widely studied and in general more robust compared to the solution of a non linear system. However, contrary to common opinion emerging from the literature, the indirect method has at least the same numerical efficiency and computational speed of the direct method. However, it must be said that statement is true if the finite difference approximation of the BVP is used as solution method (or other collocation approximation) and the resulting non linear system is solved with a robust non linear solver. For example, the standard Matlab non linear system solver `lsqnonlin()` did not converge for the nonlinear system in (35) with more than 100 points. Instead using STRSCNE [27] it took about 11s for 10000 mesh points which is comparable to the solution obtained with IPOPT for the same problem (i.e. about about 8s).

Table 1. Computational time among different solution method and solver adopted. IPOPT can use analytic Hessian or approximate it using using BFGS. Matlab FMINCON do not converge for $N > 100$. Newton solver is a simplified Matlab version of HYNESS solver described in Sect. 5.1. PINS is C++ indirect solver described in Sect. 5

Method	Solver	cpu time (s) $N = 100$	cpu time (s) $N = 1000$	cpu time (s) $N = 10000$
Direct	IPOPT [12] (analytic)	0.06	0.4	≈6
Direct	IPOPT [12] (BFGS)	0.17	2.8	≈31
Direct	fmincon	2.8	—	—
Indirect	lsqnonlin	0.33	1.4	≈16
Indirect	STRSCNE [27]	0.19	1.6	≈21
Indirect	Newton Solver	0.1	0.6	≈6
Indirect	PINS	0.05	0.2	0.85

It is worth it to point out that he dependency on a good solver it is also an issue for the NLP based approaches. The optimisation problem (29–34) if solved with Matlab `fmincon()` takes 5 seconds for 100 mesh points and for 1000 mesh points does not converge. On the contrary, the state of art IPOPT solver finds a solution in few seconds about 8s with a mesh of 10000 points. Table 1 reports all the results of the numerical tests performed using different solvers and solution methods. The interested readers may experience the above results by running the scripts used for the test and that can be downloaded from https://github.com/mechatronix/TS-OCS.

The considerations above suggest that there is not a real difference in the type of problem that is solved between the two families of methods, but it is the solver adopted that really makes the difference. To further support this result here below it is proved that the discretised optimisation problem for the NLP is almost equal to the discretised BVP resulting from the Pontryagin Minimum Principle.

The NLP (29–34) can be transformed—like in IPOPT—by a combination of Langrange multipliers and penalties to search for the stationary points of $f(z)$ defined as

$$f(z) = -\frac{\lambda_0 x_0 + \mu_0 v_0 - \mu_N v_N}{h} - \sum_{k=0}^{N-1}\left(v_{k+1/2} + b(w_{k+1/2})\right) - \sum_{k=0}^{N-1}\Big(\lambda_{k+1/2} h_k(z) + \mu_{k+1/2} h_{k+N}(z)\Big) \cdots\cdots\cdots (40)$$

where $h_k(z)$ and $h_{k+N}(z)$ are defined in Eqs. (30)–(31) and $b(u)$ is a barrier defined in (37) used as in the interior point method to force $u_{k+1/2} \in (-1, 1)$. The stationary points of $f(z)$ satisfy $\nabla f(z) = \boldsymbol{0}$ and thus

$$2\partial_{x_0} f(z) = (\lambda_{1/2} - \lambda_0)/(h/2) \cdots\cdots\cdots (41)$$

$$\partial_{x_k} f(z) = (\lambda_{k+1/2} - \lambda_{k-1/2})/h \tag{42}$$

$$\partial_{x_N} f(z) = -\lambda_{N-1/2}/h \tag{43}$$

$$2\partial_{v_0} f(z) = (\mu_{1/2} - \mu_0)/(h/2) + \lambda_{1/2} - 1 - \mu_{1/2}(k_1 + 2v_{1/2}k_2) \tag{44}$$

$$\partial_{v_k} f(z) = (\mu_{k+1/2} - \mu_{k-1/2})/h + \bar{\lambda}_k - 1 - k_1\bar{\mu}_k - 2k_2(\overline{\mu v})_k \tag{45}$$

$$2\partial_{v_N} f(z) = (\mu_N - \mu_{N-1/2})/(h/2) + \lambda_{N-1/2} - 1 - \mu_{N-1/2}(k_1 + 2v_{N-1/2}k_2) \tag{46}$$

$$\partial_{\lambda_{k+1/2}} f(z) = (x_{k+1} - x_k)/h - \bar{v}_{k+1/2} \tag{47}$$

$$\partial_{\mu_{k+1/2}} f(z) = (v_{k+1} - v_k)/h + (k_0 + k_1\bar{v}_{k+1/2} + k_2\bar{v}^2_{k+1/2} - u_{k+1/2}) \tag{48}$$

$$\partial_{w_{k+1/2}} f(z) = -\mu_{k+1/2} - \frac{\pi}{2}\varepsilon\tan\left(\frac{\pi}{2}w_{k+1/2}\right) \tag{49}$$

$$x_0 = v_0 = v_N = 0 \tag{50}$$

$$u_{k+1/2} = \left(g + k_3 v^2_{k+1/2}\right) w_{k+1/2} \tag{51}$$

where $(\overline{\mu v})_k = (\mu_{k+1/2}\bar{v}_{k+1/2} + \mu_{k-1/2}\bar{v}_{k-1/2})/2$. Equations (41)–(43) can be interpreted as a finite difference approximation of the ODE $\lambda' = 0$ with $\lambda(T) = 0$. Equations (44)–(46) can be interpreted as a finite difference approximation of the ODE $\mu' = \mu(k_1 + 2k_2 v) + \lambda - 1$ with free boundary conditions. Equations (47)–(48) with (50) can be interpreted as a finite difference approximation of $x' = v$ and $v' = u - k_0 - k_1 v - k_2 v^2$ with $x(0) = v(0) = v(T) = 0$. Solution of Eq. (49) with (51) results in

$$u_{k+1/2} = -\frac{2}{\pi}\left(g + k_3 v^2_{k+1/2}\right)\arctan\left(\frac{2}{\pi}\frac{\mu_{k+1/2}}{\epsilon}\right) \tag{52}$$

is close to (39) except that the sign function is approximated using a simpler expression.

Therefore direct and indirect methods approximates the same problem. One may see that the discretised part of the dynamical system is coincident for the two methods except for the calculation of the control $u_{k+1/2}$.

Furthermore, for the indirect method the discretisation is second order for both states and multipliers instead for the direct method the ordinary differential equations of the Lagrange multipliers at the boundary are approximated with finite difference of the first order. Therefore one will expect the indirect method to be more accurate than the direct ones. This is true for *smooth* controls when multipliers are not affine functions, but, when control are discontinuous the error is dominated by the resolution of the mesh and the accuracy is the same.

Indeed this is the result achieved for our demonstration example as explained next. The accuracy of the direct and indirect method can be calculated using the analytical solution derived in Sect. 6.1. The accuracy index is the discrete scaled 2-Norm of the difference between exact and numerical solution of the control obtained with direct and indirect method respectively obtained in Sect. 4.2 and Sect. 4.3:

$$\text{error} = \|\boldsymbol{u}^h - \boldsymbol{u}\|_2 = \left(\frac{1}{N}\sum_{k=1}^{N}|u^h_{k-1/2} - u_{k-1/2}|^2\right)^{1/2} \tag{53}$$

where $\boldsymbol{u}^h$ is the numerical solution and $\boldsymbol{u}$ is the exact solution

Table 2. Direct and Indirect method accuracy versus the number of mesh points

N	Direct	Indirect	N	Direct	Indirect
100	0.9570	0.9570	1600	0.1491	0.1491
200	0.6597	0.6597	3200	0.0374	0.0374
400	0.4424	0.4424	6400	0.06977	0.06977
800	0.2788	0.2788	10000	0.05381	0.05380
1000	0.2342	0.2342	12800	0.01196	0.01197

sampled on the computational grid. Table 2 shows that direct and indirect methods are practically indistinguishable for the proposed test.

Finally, it can be said that the indirect method may provide the analytic feedback law for the optimal control $\boldsymbol{u}(t) = \Pi(\boldsymbol{x}, \boldsymbol{\lambda}, t)$. This is in general not always true if inequalities are present and the dynamical system is non-linear. However, under some conditions semi-analytic solution can be derived as in [28][29] and exploited in real life applications. Finally, when inequalities are enforced as penalty/barrier functions the controls can always be explicitly calculated even in the case of singular arc without additional effort (at least with a local sub-iterative solution scheme). In the demonstration example it was shown that the explicit solution of the controls an their formal substitution in the equations of the approximated problem reduces the computational time and the number of iterates.

5. An Efficient Symbolic-numeric Indirect Method

The first part of the paper clearly shown that the indirect method can be adopted to solve optimal control problems with the same numerical efficiency of the direct approaches provided that the solution method is based on the approximation of the BVP with finite difference or collocation and a robust non linear solver is available.

The section explains how it is possible to obtain robust and accurate solution of optimal control problem, even for real–time applications, using the Indirect Method in combination with penalty functions to implement path constraints on controls and states and with a finite different approximation. The concepts herein presented are implemented in the code named PINS which is available for free for academic use under request to the authors.

A more general definition of the optimal control problem can be the following. Let us consider the domain $[\zeta_0, \zeta_{n_f}]$ divided in n_f intervals or phases with $n_f - 1$ discontinuity or interface points ζ_k such that $\zeta_0 < \zeta_1 < \zeta_2 < \cdots < \zeta_{n_f}$. The objective is to minimise the following cost function in the Bolza form:

$$J[\boldsymbol{u}] = \mathcal{M}(\boldsymbol{x}^-(\zeta_{n_f})) + \sum_{k=1}^{n_f}\int_{\zeta_{k-1}}^{\zeta_k}\mathcal{L}(\boldsymbol{x}(\zeta), \boldsymbol{u}(\zeta), \zeta)\,\mathrm{d}\zeta \tag{54}$$

subject to the following constraints:

$$\boldsymbol{A}(\boldsymbol{x}(\zeta), \zeta)\boldsymbol{x}'(\zeta) = \boldsymbol{f}(\boldsymbol{x}(\zeta), \boldsymbol{u}(\zeta), \zeta), \tag{55}$$

$$\boldsymbol{q}(\boldsymbol{x}^-(\zeta_k), \boldsymbol{x}^+(\zeta_k), \zeta_k) = 0, \tag{56}$$

$$\boldsymbol{b}(\boldsymbol{x}^+(\zeta_0), \boldsymbol{x}^-(\zeta_{n_f})) = 0, \tag{57}$$

for $\zeta \in (\zeta_{k-1}, \zeta_k)$ and $k = 1, 2, \ldots, n_f - 1$ where the vector $\boldsymbol{b}$ defines the boundary conditions, the vector $\boldsymbol{q}$ the interface conditions. $\boldsymbol{A}$ is a non-singular matrix with continuous and piecewise differentiable entries and it corresponds to the mass matrix of the multibody model considered in many problems. $\mathcal{M}(\cdot)$ is named *Mayer* term and $\mathcal{L}(\cdot)$ the *Lagrange* term. The Lagrange term $\mathcal{L}(\cdot)$ includes the *running cost* $\ell(\cdot)$ and path constraints $\boldsymbol{c}(\boldsymbol{x}(\zeta), \boldsymbol{u}(\zeta), \zeta) \geq \boldsymbol{0}$ that are approximated as penalty or barrier functions $\mathcal{P}(\cdot)$ [10]:

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{u}, \zeta) = \ell(\boldsymbol{x}, \boldsymbol{u}, \zeta) + \sum_{i=1}^{n_c} \mathcal{P}_i(c_i(\boldsymbol{x}, \boldsymbol{u}, \zeta)) \cdots\cdots\cdots (58)$$

Additional constraints of different types could be included in this formulation using a suitable transformations. For example integral constraints are included by adding additional states and corresponding boundary conditions. Let us now make use the Theorem of Lagrange Multipliers and transform the problem into an unconstrained problem as follows:

$$\begin{aligned} J[\boldsymbol{u}] = {} & \mathcal{M}(\boldsymbol{x}^-(\zeta_{n_f})) + \boldsymbol{\omega} \cdot \boldsymbol{b}(\boldsymbol{x}^+(\zeta_0), \boldsymbol{x}^-(\zeta_{n_f})) \\ & + \sum_{k=1}^{n_f} \int_{\zeta_{k-1}}^{\zeta_k} \mathcal{L}(\boldsymbol{x}, \boldsymbol{u}, \zeta) + \boldsymbol{\lambda}(\zeta) \cdot \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, \zeta) \, \mathrm{d}\zeta \\ & - \sum_{k=1}^{n_f} \int_{\zeta_{k-1}}^{\zeta_k} \boldsymbol{a}(\boldsymbol{x}(\zeta), \boldsymbol{\lambda}(\zeta), \zeta) \cdot \boldsymbol{x}'(\zeta) \, \mathrm{d}\zeta \\ & + \sum_{k=1}^{n_f - 1} \boldsymbol{\eta}_k \cdot \boldsymbol{q}(\boldsymbol{x}^-(\zeta_k), \boldsymbol{x}^+(\zeta_k), \zeta_k) \cdots\cdots\cdots (59) \end{aligned}$$

where $\boldsymbol{a}(\boldsymbol{x}, \boldsymbol{\lambda}, \zeta) = \boldsymbol{A}(\boldsymbol{x}, \zeta)^T \boldsymbol{\lambda}$ and $\boldsymbol{\lambda}(\zeta)$, $\boldsymbol{\omega}$, $\boldsymbol{\eta}$ are respectively the piecewise continuous, the point wise and the interface node Lagrange multipliers. Let us now define the hamiltonian function $\mathcal{H}$ and the vector $\mathcal{B}$ that collects all the algebraic conditions involving the state $\boldsymbol{x}$ at the boundaries:

$$\begin{aligned} \mathcal{H}(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{u}, \zeta) &= \mathcal{L}(\boldsymbol{x}, \boldsymbol{u}, \zeta) + \boldsymbol{\lambda} \cdot \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, \zeta) \\ \mathcal{B}(\boldsymbol{x}_0, \boldsymbol{x}_f, \boldsymbol{\omega}) &= \mathcal{M}(\boldsymbol{x}_f) + \boldsymbol{\omega} \cdot \boldsymbol{b}(\boldsymbol{x}_0, \boldsymbol{x}_f) \cdots\cdots\cdots (60) \end{aligned}$$

Let us know perform the first variation obtaining the *necessary* conditions for the solution of the problem (59) that must satisfy the following Boundary Value Problem (BVP):

$$\begin{aligned} & \boldsymbol{A}(\boldsymbol{x}, \zeta)\boldsymbol{x}' - \partial_{\boldsymbol{\lambda}}^T \mathcal{H}(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{u}, \zeta) = \boldsymbol{0}, \cdots\cdots\cdots (61) \\ & \boldsymbol{a}_x(\boldsymbol{x}, \boldsymbol{\lambda}, \zeta)\boldsymbol{x}' - \boldsymbol{A}(\boldsymbol{x}, \zeta)^T \boldsymbol{\lambda}' - \partial_{\boldsymbol{x}}^T \mathcal{H}(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{u}, \zeta) = \boldsymbol{0}, \\ & \cdots\cdots\cdots (62) \\ & \boldsymbol{b}(\boldsymbol{x}(\zeta_0), \boldsymbol{x}(\zeta_f)) = \boldsymbol{0}, \cdots\cdots\cdots (63) \\ & \boldsymbol{q}(\boldsymbol{x}^-(\zeta_k), \boldsymbol{x}^+(\zeta_k), \zeta_k) = \boldsymbol{0}, \cdots\cdots\cdots (64) \\ & \partial_{\boldsymbol{x}_0}^T \mathcal{B}(\boldsymbol{x}(\zeta_0), \boldsymbol{x}(\zeta_f), \boldsymbol{\omega}) - \boldsymbol{A}(\boldsymbol{x}(\zeta_0), \zeta_0)^T \boldsymbol{\lambda}(\zeta_0) = \boldsymbol{0}, \\ & \cdots\cdots\cdots (65) \\ & \partial_{\boldsymbol{x}_f}^T \mathcal{B}(\boldsymbol{x}(\zeta_0), \boldsymbol{x}(\zeta_f), \boldsymbol{\omega}) + \boldsymbol{A}(\boldsymbol{x}(\zeta_f), \zeta_f)^T \boldsymbol{\lambda}(\zeta_f) = \boldsymbol{0}, \\ & \cdots\cdots\cdots (66) \\ & \boldsymbol{Q}(\boldsymbol{x}^-(\zeta_k), \boldsymbol{\lambda}^-(\zeta_k), \boldsymbol{x}^+(\zeta_k), \boldsymbol{\lambda}^+(\zeta_k), \zeta_k) = \boldsymbol{0}, \cdots\cdots (67) \end{aligned}$$

where $\boldsymbol{a}_x(\boldsymbol{x}, \boldsymbol{\lambda}, \zeta) = \partial_{\boldsymbol{x}}^T \boldsymbol{a}(\boldsymbol{x}, \boldsymbol{\lambda}, \zeta) - \partial_{\boldsymbol{x}} \boldsymbol{a}(\boldsymbol{x}, \boldsymbol{\lambda}, \zeta)$ and

$$\boldsymbol{u}(\zeta) = \arg\min_{\boldsymbol{v} \in \mathcal{U}} \mathcal{H}(\boldsymbol{x}(\zeta), \boldsymbol{\lambda}(\zeta), \boldsymbol{v}) \cdots\cdots\cdots (68)$$

states the Pontryagin Minimum Principle. Vector function $\boldsymbol{Q}$ contains the additional interface conditions after removing multiplier $\boldsymbol{\eta}_k$. In fact the additional interface conditions are resumed in equations of the form

$$\begin{pmatrix} \partial_{\boldsymbol{x}^-}^T \boldsymbol{q}(\boldsymbol{x}^-, \boldsymbol{x}^+, \zeta) \\ \partial_{\boldsymbol{x}^+}^T \boldsymbol{q}(\boldsymbol{x}^-, \boldsymbol{x}^+, \zeta) \end{pmatrix} \boldsymbol{\eta} = \begin{pmatrix} \boldsymbol{A}(\boldsymbol{x}^-)^T \boldsymbol{\lambda}^- \\ -\boldsymbol{A}(\boldsymbol{x}^+)^T \boldsymbol{\lambda}^+ \end{pmatrix} \cdots\cdots\cdots (69)$$

where $\boldsymbol{x}^-$ and $\boldsymbol{x}^+$ are the vector of states $k = 1, \ldots, n_f - 1$ at left and right interface nodes. The multiplier $\boldsymbol{\eta}$ can be computed and eliminated from (69) as a function $\boldsymbol{\eta}(\boldsymbol{x}^-, \boldsymbol{\lambda}^-, \boldsymbol{x}^+, \boldsymbol{\lambda}^+, \zeta)$ if the matrix that multiply $\boldsymbol{\eta}$ is full rank. This elimination is done symbolically by the software.

Having used the barrier function to implement the inequality constraints on the controls it is guaranteed that $\boldsymbol{u}$ is bounded in a compact set $\mathcal{U}$ and the problem is 'smooth'.

The optimal controls are the ones that satisfies the condition (68) where $\mathcal{H}(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{u}, \zeta)$ contains the barrier function for the controls. Let us know assume that the barrier function is: $-\epsilon \log \operatorname{dist}(\boldsymbol{u}, \mathbb{R}^m \backslash \mathcal{U})$ where $\operatorname{dist}(\boldsymbol{u}, \mathcal{A}) = \inf\{|\boldsymbol{u} - \boldsymbol{u}^*|, \boldsymbol{u}^* \in \mathcal{A}\}$ which is small and positive for $\boldsymbol{u} \in \mathcal{U}$ and is ∞ for $\boldsymbol{u} \notin \mathcal{U}$. Let us note that as ϵ becomes small the problem (68) approximates the following minimization:

$$\boldsymbol{u} = \arg\min_{\boldsymbol{v} \in \mathcal{U}} \widetilde{\mathcal{H}}(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{v}) \cdots\cdots\cdots (70)$$

where now $\widetilde{\mathcal{H}}$ is the Hamiltonian of the *constrained* optimal control problem (*i.e.* without the barrier function for the control bounds but pure inequalities). Equation (70) is part of the Pontryagin Minimum Principle [8].

5.1 Numerical Solution Approach for Indirect Method The use of penalty or barrier functions transform the BVP (61) into a smooth problem therefore any numerical method can be used to approximate the solution of the BVP (61) such as a second order finite difference (as it was done in Sect. 6.1 but higher order discretisation or collocation can be adopted). Additionally, it is assumed that the controls can be explicitly solved with respect to $\boldsymbol{x}$ and $\boldsymbol{\lambda}$ since for each node of the discretisation the optimal control can be obtained by solving for $\boldsymbol{u}$ in the following:

$$\partial_{\boldsymbol{u}}^T \mathcal{H}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{\lambda}) = 0 \cdots\cdots\cdots (71)$$

This is possible because the penalty functions used to implement the inequalities of the control vector $\boldsymbol{u}$ makes the system unconstrained and therefore the first variation provides the set of Eq. (71).

The approximation of the BVP (61) with finite difference yields a large non linear system of equations $\boldsymbol{\Phi}(\boldsymbol{Z})$. Newton method can be used to solve the non linear system, however, being the Newton method not globally convergent a good initial starting point with globalisation techniques are mandatory. Line search allows the method to enlarge the solution attraction basin. By introducing the notations $\boldsymbol{J}(\boldsymbol{Z}) = \partial_{\boldsymbol{Z}} \boldsymbol{\Phi}(\boldsymbol{Z})$ a basic scheme can be sketched as follows:

- compute direction $\boldsymbol{d}_j$ by solving: $\boldsymbol{J}(\boldsymbol{Z}_j)\boldsymbol{d}_j = -\boldsymbol{\Phi}(\boldsymbol{Z}_j)$
- find dumping factor α_j that satisfies Armijo condition on merit function: $m_j(\alpha) = \|\boldsymbol{J}(\boldsymbol{Z}_j)^{-1} \boldsymbol{\Phi}(\boldsymbol{Z}_j + \alpha_j \boldsymbol{d}_j)\|^2$
- update $\mathbf{Z}$: $\mathbf{Z}_{j+1} = \mathbf{Z}_j + \alpha_j \boldsymbol{d}_j$

The merit function makes the algorithm affine invariant which ensure a good robustness of the approach. Finally, to improve robustness in particular when the starting point $\mathbf{Z}_0$ is far from the solution, non-monotone iterations are allowed. A custom designed, robust and fast non linear solver that exploits the problem structures of the discretised BVP (61) and solves in sub-iterations at each node the equations of controls has been implemented based on the above concepts [10][30][31].

Additionally, by using an symbolic algebra computational environment such as Maple ©, Mathematica © or CasADi (https://github.com/casadi) one may derive symbolically the necessary equations and the jacobians required by the Newton method. The symbolic equations improve accuracy and convergence rate as one may easily prove by de-activating the symbolic jacobian in the tests that uses IPOPT and fminc().

An Optimal Control Suite (https://github.com/mechatronix/TS-OCS) has been developed based on the above approach that is made of a Maple package for equation generation (called *XOptima*), a C++ library for Numerical Solution (called *Mechatronix*) and a embedded ruby interpreter (called *PINS*). Table 2 shows the performance that can be achieved with this software.

6. Successful Applications of Indirect Method

The approach and software implemented in Sect. 5 has been extensively used in different engineering fields such as robot motion planning [10][32], car and racing motorcycles dynamics and in machining applications.

In particular the applications in vehicle dynamic field can be divided between off-line trajectory optimisation problems [30][31][33]–[35] and on-line receding horizon based implementation both for driving support [10][36]–[41] and autonomous driving [42]–[45]. Details can be found in the cited papers. In manufacturing field we have proposed a prototype for a supervising controller for machining applications where the controller relies on the definition of an optimal control problem that aims at calculating the sequence of controls (feed rate and tool speed) that minimises a multi-objective target function [46]. The indirect approach was also used to implement an algorithm for CNC kernels that aims at solving the axes interpolation [47].

6.1 An Example from Vehicle Dynamics

The indirect method proposed in Sect. 5 is here applied on a more complex vehicle dynamic problem to prove its performance. The proposed problem is the minimum time manoeuvre of a four wheel vehicle (here for sake of space modelled as single track) to travel along a $U-$shaped curve. The model includes a simplified version of the combined lateral longitudinal tyre forces as in Pacejka [48]. Therefore tyres' sideslip and longitudinal slip dynamics have to be accounted. The dynamics is written using the curvilinear coordinate s as independent variable becomes:

$$\dot{s}(s)n'(s) = \sin(\xi)u + \cos(\xi)v \quad\cdots\cdots (72)$$

$$\dot{s}(s)\xi'(s) = \Omega - \frac{C(s)}{1 - C(s)n}(\cos(\xi)u - \sin(\xi)v) \quad\cdots\cdots (73)$$

$$m\dot{s}(s)u'(s) = 2\left(c_\delta F_{x_f} - s_\delta F_{y_f} + F_{x_r}\right) + m\Omega v - k_v u^2 \quad\cdots\cdots (74)$$

$$m\dot{s}(s)v'(s) = 2\left(s_\delta F_{x_f} + c_\delta F_{y_f} + F_{y_r}\right) - m\Omega u \quad\cdots\cdots (75)$$

$$I_{zz}\dot{s}(s)\Omega'(s) = 2\left(L_f(s_\delta F_{x_f} + c_\delta F_{y_f}) - L_r F_{y_r}\right) \quad\cdots\cdots (76)$$

$$\sigma_{y,r}\dot{s}(s)\alpha_r'(s) = u\arctan((v - \Omega L_r)/u) - u\alpha_r \quad\cdots\cdots (77)$$

$$\sigma_{y,f}\dot{s}(s)\alpha_f'(s) = u\arctan\left(\frac{L_f\Omega c_\delta - s_\delta u + c_\delta v}{L_f\Omega s_\delta + c_\delta u + s_\delta v)}\right) - u\alpha_f \quad\cdots\cdots (78)$$

$$\sigma_{x,r}\dot{s}(s)\kappa_r'(s) = u(\kappa_{\text{ro}} - \kappa_r) \quad\cdots\cdots (79)$$

$$\sigma_{x,f}\dot{s}(s)\kappa_f'(s) = u(\kappa_{\text{fo}} - \kappa_f) \quad\cdots\cdots (80)$$

$$\dot{s}(s)\delta'(s) = v_\delta \quad\cdots\cdots (81)$$

$$c_\delta = \cos(\delta), \quad s_\delta = \sin(\delta) \quad\cdots\cdots (82)$$

$$\dot{s}(s) = \frac{\cos(\xi(s))u(s) - \sin(\xi(s))v(s)}{1 - C(s)n(s)} \quad\cdots\cdots (83)$$

where $0 \le s \le S$ with S the length of the track and the states are $\boldsymbol{x} = [n, \xi, u, v, \Omega, \alpha_r, \alpha_f, \kappa_r, \kappa_f, \delta]$ where independent s, with $n(s)$, $\xi(s)$ are the curvilinear coordinates, $C(s)$ is the road middle line curvature, $u(s)$, $v(s)$ are components of absolute centre of mass velocity expressed in moving frame, $\Omega(s)$ is the yaw rate, $\alpha_r(s)$ and $\alpha_f(s)$ represent the side slip angles and $\kappa_r(s)$ and $\kappa_f(s)$ represent the longitudinal slips, lastly, $\delta(s)$ is the steering angle.

The system input $\boldsymbol{u} = [v_\delta, \kappa_{ro}, \kappa_{fo}]$ are the steering angle rate $v_\delta(s)$ and and target longitudinal slip $\kappa_{ro}(s)$ and $\kappa_{fo}(s)$. Having chosen the target longitudinal slips as control we implicitly assume that a low level controller is used to track those references. The optimal control formulates as finding the controls $\boldsymbol{u}$ in order to minimise the time T spent to travel along the road:

$$J[\boldsymbol{u}] = T = \int_0^S w_T \frac{\mathrm{d}s}{\dot{s}(s)} \quad\cdots\cdots (84)$$

subject to the following boundary conditions:

$$n(0) = \xi(0) = v(0) = \Omega(0) = \delta(0) = 0, \quad\cdots\cdots (85)$$

$$\alpha_r(0) = \alpha_f(0) = 0,\ u(0) = U_0,\ n(S) = \xi(S) = 0, \quad\cdots\cdots (86)$$

and constraints:

$$|n(s)| \le n_{\max}, \qquad |\delta(s)| \le \delta_{max}, \qquad |v_\delta| \le v_{\delta_{\max}},$$

$$|\kappa_{\text{ro}}(s)| \le \kappa_{\text{ro}}^{\max}, \qquad |\kappa_{\text{fo}}(s)| \le \kappa_{\text{fo}}^{\max}.$$

The functions $F_{x_f} = F_{x_f}(\kappa_f, \alpha_f)$, $F_{y_f} = F_{y_f}(\kappa_f, \alpha_f)$, $F_{x_r} = F_{x_r}(\kappa_r, \alpha_r)$ and $F_{y_r} = F_{y_r}(\kappa_r, \alpha_r)$ are a simplified version of the combined Pacejka tyre model as follows:

$$F_X(\kappa, \alpha) = F_Z\mu_X \sin(C_X \arctan(B_X\kappa)) \cos\left(C_{Xa}\arctan\left(B_{X_1}\alpha\Big/\sqrt{B_{X_2}^2\kappa^2 + 1}\right)\right) \quad\cdots\cdots (87)$$

$$F_Y(\kappa, \alpha) = F_Z\mu_Y \sin(C_Y \arctan(B_Y\kappa)) \cos\left(C_{Yk}\arctan\left(B_{Y_1}\alpha\Big/\sqrt{B_{Y_2}^2\kappa^2 + 1}\right)\right) \quad\cdots\cdots (88)$$

where $X = x_r$ or x_f and $Y = y_r$ or y_f and $F_Z = mgL_i/(L_f + L_r)$ with $i = f, r$ for rear and front axes respectively. In the test the track is described by the curvature $C(s)$ a piecewise constant function:

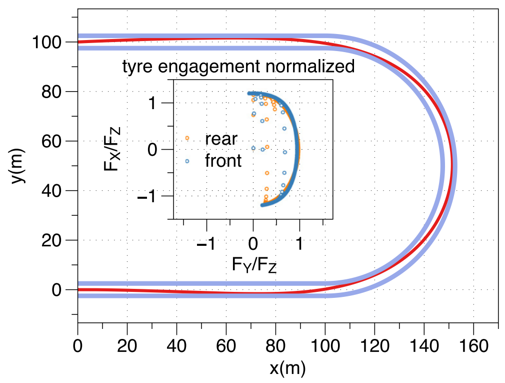

Fig. 8. Trajectory of the vehicle center of mass and related tyres' engagements as lateral and longitudinal forces normalised with vertical loads

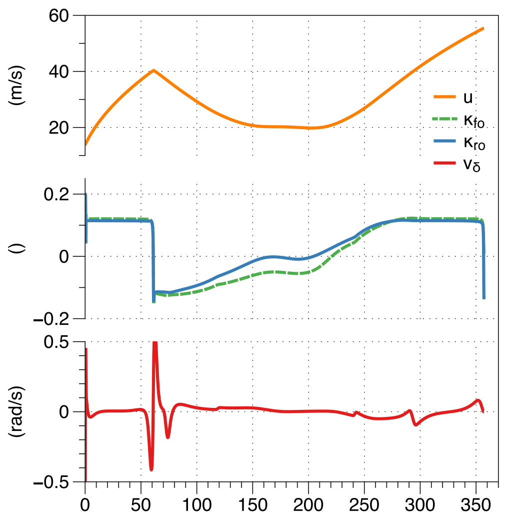

Fig. 9. Top graph shows forward speed profile and below the controls

Table 3. Vehicle dynamics test case parameters

$m = 2100$	$g = 9.82$	$I_{zz} = 3900$	$k_v = 0$	$L_f = 1.3$
$L_r = 1.5$	$\sigma_{x,f} = 0.5$	$\sigma_{x,r} = 0.5$	$\sigma_{y,f} = 0.3$	$\sigma_{y,r} = 0.3$
$\kappa_{\mathrm{ro}}^{\max} = 0.2$	$\kappa_{\mathrm{fo}}^{\max} = 0.2$	$\delta_{max} = 75°$	$v_{\delta\max} = 60°$	$n_{\max} = 2.5$
$\mu_{x,f} = 1.20$	$C_{x,f} = 1.69$	$C_{x_{a,f}} = 1.09$	$B_{x,f} = 11.7$	$B_{x_{1,f}} = 12.4$
$B_{x_{2,f}} = -10.8$	$\mu_{x,r} = 1.20$	$C_{x,r} = 1.69$	$C_{xa,r} = 1.09$	$B_{x,r} = 11.1$
$B_{x_{1,r}} = 12.4$	$B_{x_{2,r}} = -10.8$	$\mu_{y,f} = 0.935$	$C_{y,f} = 1.7$	$C_{y_{k,f}} = 1.08$
$B_{y,f} = 8.86$	$B_{y_{1,f}} = 6.46$	$B_{y_{2,f}} = 4.20$	$\mu_{y,r} = 0.961$	$C_{y,r} = 1.7$
$C_{y_{k,r}} = 1.08$	$B_{y,r} = 7.0$	$B_{y_{1,r}} = 6.46$	$B_{y_{2,r}} = 4.20$	$S = 357.08$

$$C(s) = \begin{cases} 0 & \text{for } s \in [0, 100] \\ 1/50 & \text{for } s \in [100, 100 + 50\pi] \\ 0 & \text{for } s \in [100 + 50\pi, 200 + 50\pi] \end{cases} \quad \cdots\cdots\cdots\cdots (89)$$

In general $C(s)$ can be a piecewise linear function and the resulting road shape is a piecewise clothoid that can be computed fast and accurately [49]. The parameter used in the simulation are found in Table 3 while $U_0 = 50/3.6$ [m/s]. The path is discretized by 3580 mesh points. The discretized BVP resulting into a non linear system of 71510 non linear equations. A continuation approach on parameter w_T was used to ease the convergence. First solution was searched for $w_T = 0.01$ and progressively increased to 1 in order to have the original minimum time problem. The overall computational time is less than 6 seconds for a total of 73 iterations. It is worth it to mention that no particular initial guess for states and lagrange multipliers has been used for this problem. Lagrange multiplier guess is set to zero and the same for all the states excepts longitudinal speed and yaw rate that are set equal to $u = V_0$ and $\Omega = V_0 C(s)$ respectively.

7. Conclusions

Differently than what emerges from literature on optimal control methods and applications, we have shown that indirect method is also a competitive solution approach from all the points of view and was successfully applied to complex engineering problem since the end of 90s. However, according to our experience, to obtain numerical performance excellence with indirect method, some practical measures have to be followed. First of all the resulting BVP has to be approximated with finite difference (or collocation methods) to get a large non linear system which is much less sensible to the initial guess of Lagrange multipliers. Actually, in all our applications the Lagrange multiplier guess is set to zero. Secondly a robust non linear solver must be adopted in combination with continuations on penalty parameters or on minimum time term. This is very similar to what is done by state of art NLP solvers such as IPOPT. Finally, the indirect method allows to sub-solve the optimal control as a function of states and Lagrange multipliers—as formal solution or with iterative scheme—and formally get rid of the controls from the non linear system. This greatly improves the convergence rate as it is shown in the demonstration example Sect. 6.1.

The considerations put forward in this work are supported by the numerical tests that have been reported in Sect. 6.1 for the adopted demonstration example—the source code for all test being available for download at https://github.com/mechatronix/TS-OCS. Clearly, the optimisation solver or equivalently the solver of the non linear system plays a fundamental role in the numerical performance achieved.

Further work is still necessary to better understand the similarities and differences among direct and indirect approaches and compare the results on more complex problems including in set of the direct solution methods the multiple shooting which was not discussed here for sake of space.

References

(1) V. Rao Anil: "A survey of numerical methods for optimal control", *Advances in the Astronautical Sciences*, Vol.135, No.1, pp.497–528 (2009)
(2) H.S. Rodrigues, M. Teresa T. Monteiro, and D.F.M. Torres: Optimal control and numerical software: An overview. In Francisco Miranda, editor, Systems Theory: Perspectives, Applications and Developments, Systems Science Series. Nova Science (2014)
(3) R. Bellman: "The theory of dynamic programming", Bull. Amer. Math. Soc., Vol.60, No.6, pp.503–515 (1954)
(4) R. Luus: Iterative Dynamic Programming, CRC Press (2000)
(5) F.B. Hanson: Techniques in computational stochastic dynamic programming, In Control and Dynamic Systems, pp.103–162. Academic Press (1996)
(6) P. Elbert, S. Ebbesen, and L. Guzzella: "Implementation of dynamic programming for n-dimensional optimal control problems with final state constraints", *IEEE Trans. on Control Systems Technology*, Vol.21, No.3, pp.924–931 (2013)

(7) O. Sundstrom and L. Guzzella: “A generic dynamic programming matlab function”, In Proceedings of the 18th IEEE International Conference on Control Applications, pp.1625–1630, Saint Petersburg, Russia (2009)

(8) L.S. Pontryagin, V.G. Boltyanskii, R.V. Gamkrelidze, and E. Mishchenko: “The mathematical theory of optimal processes (International series of monographs in pure and applied mathematics)”, Interscience Publishers (1962)

(9) D. Liberzon: “Calculus of Variations and Optimal Control Theory: A Concise Introduction”, Princeton University Press, Princeton, NJ, USA (2011)

(10) E. Bertolazzi, F. Biral, and M. Da Lio: “Real-time motion planning for multibody systems: Real life application examples”, *Multibody System Dynamics*, Vol.17, No.2-3, pp.119–139 (2007)

(11) J.T. Betts: Practical Methods for Optimal Control and Estimation Using Nonlinear Programming, Cambridge University Press, 2nd edition (2009)

(12) A. Wächter and L.T. Biegler: “On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming”, *Mathematical Programming*, Vol.106, No.1, pp.25–57 (2006)

(13) J. Nocedal and S. Wright: Numerical Optimization, Springer Series in Operations Research and Financial Engineering, Springer (2006)

(14) D. Garg, M.A. Patterson, W.W. Hager, A.V. Rao, D.A. Benson, and G.T. Huntington: “A unified framework for the numerical solution of optimal control problems using pseudospectral methods”, *Automatica*, Vol.46, No.11, pp.1843–1851 (2010)

(15) Q. Gong, W. Kang, N.S. Bedrossian, F. Fahroo, P. Sekhavat, and K. Bollino: “Pseudospectral optimal control for military and industrial applications”, In 46th IEEE Conference on Decision and Control, pp.4128–4142 (2007)

(16) I. Michael Ross and M. Karpenko: “A review of pseudospectral optimal control: From theory to flight”, *Annual Reviews in Control*, Vol.36, No.2, pp.182–197 (2012)

(17) 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*, Vol.32, No.3, pp.298–312 (2011)

(18) M.A. Patterson and A.V. Rao: “GPOPS-II: a MATLAB software for solving multiple-phase optimal control problems using hp-adaptive gaussian quadrature collocation methods and sparse nonlinear programming”, *ACM Trans. Math. Softw.*, Vol.41, No.1, pp.1–37 (2014)

(19) P.E. Rutquist and M.M. Edvall: PROPT—matlab optimal control software, Technical Report, Tomlab Optimization (2010)

(20) A. Schwartz, E. Polak, and Y.Q. Chen: Riots manual: A matlab toolbox for solving optimal control problems. Technical report (1997)

(21) M. Diehl, D.B. Leineweber, and A.A.S. Schäfer: MUSCOD-II Users’ Manual. Technical Report 2001-25 (2001)

(22) I.M. Ross and F Fahroo: “Pseudospectral knotting methods for solving optimal control problems”, *Journal of Guidance, Control and Dynamics*, Vol.27, No.3, pp,397–405 (2004)

(23) L. Guzzella. DPM function, http://www.idsc.ethz.ch/research-guzzella-onder (2015)

(24) R.H. Byrd, J. Nocedal, and R.A. Waltz. Knitro: “An integrated package for nonlinear optimization”, In Large Scale Nonlinear Optimization, 35–59, 2006, pp35–59, Springer Verlag (2006)

(25) R.J. Vanderbei: “LOQO: an interior point code for quadratic programming”, *Optimization Methods and Software*, Vol.11, No.1-4, pp.451–484 (1999)

(26) C. Büskens and D. Wassel: The ESA NLP solver WORHP, In Giorgio Fasano and János D. Pintér, editors, *Modeling and Optimization in Space Engineering*, volume 73 of *Springer Optimization and Its Applications*, pp.85–110, Springer New York (2013)

(27) S. Bellavia, M. Macconi, and B. Morini: “STRSCNE: a scaled trust-region solver for constrained nonlinear equations”, *Computational Optimization and Applications*, Vol.28, No.1, pp.31–50 (2004)

(28) M. Da Lio, F. Biral, E. Bertolazzi, M. Galvani, P. Bosetti, D. Windridge, A. Saroldi, and F. Tango: “Artificial co-drivers as a universal enabling technology for future intelligent vehicles and transportation systems”, *IEEE Intelligent Transportation Systems Magazine*, Vol.16, No.1, pp.244–263 (2014)

(29) P. Bosetti, M. Da Lio, and A. Saroldi: “On curve negotiation: from driver support to automation”, *IEEE Transactions on Intelligent Transportation Systems* (2015)

(30) E. Bertolazzi, F. Biral, and M. Da Lio: “Symbolic-numeric indirect method for solving optimal control problems for large multibody systems: The time-optimal racing vehicle example”, *Multibody System Dynamics*, Vol.13, No.2, pp.233–252 (2005)

(31) E. Bertolazzi, F. Biral, and M. Da Lio: “Symbolic-numeric efficient solution of optimal control problems for multibody systems”, *Journal of Computational and Applied Mathematics*, Vol.185, No.2, pp.404–421 (2006)

(32) L. Baglivo, N. Biasi, F. Biral, N. Bellomo, E. Bertolazzi, M. Da Lio, and M. De Cecco: “Autonomous pallet localization and picking for industrial forklifts: A robust range and look method”, *Measurement Science and Technology*, Vol.22, No.8 (2011)

(33) F. Biral, E. Bertolazzi, and M. da Lio: The optimal manoeuvre, In M. Tanelli, M. Corno, and S.M. Savaresi, editors, *Modelling, Simulation and Control of Two-Wheeled Vehicles*, chapter 5, pp.119–154, John Wiley & Sons, Ltd (2014)

(34) R. Lot and F. Biral: “A curvilinear abscissa approach for the lap time optimization of racing vehicles”, In 19*th* *IFAC World Congress*, pp.7559–7565, International Federation of Automatic Control (2014)

(35) F. Biral, F. Zendri, E. Bertolazzi, P. Bosetti, M. Galvani, F. Trivellato, and M.Da Lio: A web based “virtual racing car championship” to teach vehicle dynamics and multidisciplinary design, In ASME, editor, *Proceedings of 2011 ASME International Mechanical Engineering Congress and Exposition*, Denver, USA, ASME, ASME (2011)

(36) A. Amditis, E. Bertolazzi, M. Bimpas, F. Biral, P. Bosetti, M. Da Lio, L. Danielsson, A. Gallione, H. Lind, A. Saroldi, and A. Sjögren: A holistic approach to the integration of safety applications: The INSAFES subproject within the european framework programme 6 integrating project PReVENT. *IEEE Transactions on Intelligent Transportation Systems*, Vol.11, No.3, pp.554–566 (2010)

(37) E. Bertolazzi, F. Biral, M. Da Lio, A. Saroldi, and F. Tango: “Supporting drivers in keeping safe speed and safe distance: The SASPENCE subproject within the european framework programme 6 integrating project PReVENT”, *IEEE Transactions on Intelligent Transportation Systems*, Vol.11, No.3, pp.525–538 (2010)

(38) F. Biral, R. Lot, S. Rota, M. Fontana, and V. Huth: “Intersection support system for powered two-wheeled vehicles: Threat assessment based on a receding horizon approach”, *IEEE Transactions on Intelligent Transportation Systems*, Vol.13, No.2, pp.805–816 (2012)

(39) F. Biral, P. Bosetti, and R. Lot: “Experimental evaluation of a system for assisting motorcyclists to safely ride road bends”, *European Transport Research Review*, Vol.6, No.4, pp.411–423 (2014)

(40) V. Huth, R. Lot, F. Biral, and S. Rota: “Intelligent intersection support for powered two-wheeled riders: A human factors perspective”, *IET Intelligent Transport Systems*, Vol.6, No.2, pp.107–114 (2012)

(41) V. Huth, F. Biral, O. Martín, and R. Lot: “Comparison of two warning concepts of an intelligent curve warning system for motorcyclists in a simulator study”, *Accident Analysis and Prevention*, Vol.44, No.1, pp.118–125 (2012)

(42) E. Bertolazzi, F. Biral, P. Bosetti, M. De Cecco, R. Oboe, and F. Zendri: Development of a reduced size unmanned car. In *International Workshop on Advanced Motion Control, AMC*, Vol.1, pp.763–770 (2008)

(43) F. Biral, E. Bertolazzi, D. Bortoluzzi, and P. Bosetti: “Development and testing of an autonomous driving module for critical driving conditions”, In *ASME International Mechanical Engineering Congress and Exposition, Proceedings*, Vol.17, pp.361–370 (2009)

(44) D. Bortoluzzi, F. Biral, E. Bertolazzi, P. Bosetti, and F. Zendri: Influence of vehicle model complexity in autonomous emergency manoeuvre planning. In *ASME International Mechanical Engineering Congress and Exposition, Proceedings*, Vol.17, pp.371–380 (2009)

(45) M. Galvani, F. Biral, B.M. Nguyen, and H. Fujimoto: Four wheel optimal autonomous steering for improving safety in emergency collision avoidance manoeuvres. In *International Workshop on Advanced Motion Control, AMC*, pp.362–367 (2014)

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

(47) P. Bosetti and E. Bertolazzi: “Feed-rate and trajectory optimization for CNC machine tools”, *Robot. Comput.-Integr. Manuf.*, Vol.30, No.6, pp.667–677 (2014)

(48) H.B. Pacejka and Society of Automotive Engineers: *Tire and Vehicle Dynamics*. SAE-R. SAE International (2006)

(49) E. Bertolazzi and M. Frego: “G^1 fitting with clothoids”, *Mathematical Methods in the Applied Sciences*, Vol.38, No.5, pp.881–897 (2015)

Francesco Biral (Non-member) was born in Italy, on February 2nd, 1972. In 1997 he received the Master Degree in Mechanical Engineering at the University of Padova, Italy, and the Ph.D. in Mechanism and Machine Theory from University of Brescia, Italy, in 2000. He is currently Associate Professor at the Department of Industrial Engineering at University of Trento. His research interests include symbolic and numerical multibody dynamics and optimisation, constrained optimal control, mainly in the field of vehicle dynamics with special focus on Intelligent Transportation Systems.

Enrico Bertolazzi (Non-member) received the Master Degree in mathematics from the University of Trento, Italy, in 1990. He is currently Associate Professor with the University of Trento. His research interests are in numerical analysis and include Finite Volumes, discrete maximum principle, constrained optimal control, scientific programming. Prior to degree in mathematics he worked in a small company developing numerical algorithm for computer-aided manufacturing.

Paolo Bosetti (Non-member) was born in Italy, on February 28, 1971. He received Ph.D. in Engineering of Materials and Structures from University of Trento, and he is currently Assistant Professor at the Department of Industrial Engineering at University of Trento. Main research focus is on intelligent manufacturing, following the paradigm of sense, plan, control strategy that is the reference of autonomous robots in other fields.